Design of a mobile prototype system to capture, transfer and analyze microscope images of biological specimens

Tobias Söderlund
Master’s Thesis in Microelectronics
School of Information and Communication Technology
Royal Institute of Technology
September 2009 - January 2010

Jim Jansson
Master’s Thesis in Information and Communication Technology
School of Information and Communication Technology
Royal Institute of Technology
September 2009 - January 2010

Supervisor
Peter Håkansson
Ericsson Research

Co-supervisor
Professor Ewert Linder
Karolinska Institutet

Co-supervisor
Docent Johan Lundin
Karolinska Institutet
Division for global health
and
University of Helsinki
Institute for Molecular Medicine Finland

Examiner
Associate Professor Ingo Sander
Royal Institute of Technology
School for Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007
TRITA-ICT-EX-2010:4
KTH/ICT/ECS-2010-02-17
Abstract

Recent advances in imaging technology and the rapid development of mobile communications have provided the prospect of performing microscopy diagnostics by image transfer via mobile phones.

This study was motivated by the success of virtual microscopy as a method of viewing and processing digital microscopy specimens remotely over computer networks. The use of web-based virtual microscopy both for education and quality assessment in clinical parasitology was recently published as a result of collaboration between University of Helsinki, the Swedish Institute for Infectious Disease Control and Karolinska Institutet.

Also, the study was inspired by recent advances in lens less microscopy imaging and preliminary experiments with on-chip imaging by researchers at University of Helsinki and Karolinska Institutet. This put a focus on the potential of miniaturized "mobile microscopy" as a tool in primary health care, especially in low-resource settings with limited access to microscopy diagnostics expertise.

Mobile microscopy with the use of on chip technologies opens up a whole new world of possibilities for people in rural areas in developing countries. The proposed concept suggests a handset device that can upload high resolution images to a centralized server, containing both automatic image analysis and manual interpretation of specimen samples. The mobile device then fetches an answer from the server and gives the medical personnel assistance in diagnosing the patient. This report explains the steps necessary to create such a concept including both the hardware and software behind it.

A hardware prototype has been built and works. A standard conventional image sensor was used, normally found in mobile phones and digital cameras. Conventional microscope slides can be inserted and images can be apprehended. The result is a small, mobile device with mobile network and internet connection capabilities via GPRS, HSDPA and WLAN. The software prototype system includes a mobile application and a server. These systems handle patient data and images. A mobile application runs on the built prototype with connection capabilities to the centralized server. Picture Meta data containing GPS coordinates and patient information follow the uploaded picture making it possible to track and update the patient’s health record.

The resolution acquired by the handset prototype was 12.4 µm. The theoretical boundary was calculated to be 10.5 µm for the standard CMOS sensor. The main reason why this boundary wasn’t reached is that the object was placed 0.3 mm above the sensor surface and not directly on the surface, which would improve the resolution. A green LED was used together with a lens system to improve light propagation. Green color was used due to the applied Bayer filter on the sensor surface, demanding green for best resolution. To further enhance the resolution a collimated light source should be used.

It has been proved that the concept works and that different disease can be diagnosed with the built prototype however, further clinical studies are needed. Laboratory tests have been done at the Swedish Institute of Infectious Disease Control and the findings are described in this report. The diseases that were investigated were Schistosomiasis, Roundworm and Human whipworm. Some results can be seen below in the table.
<table>
<thead>
<tr>
<th>Disease</th>
<th>On chip microscope</th>
<th>Optical microscope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schistosomiasis mansoni eggs in a sample prepared with the Kato-Katz method.</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>Roundworm parasites</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>Whipworm parasites</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>

The server application was built using the Java Enterprise Edition (Java EE) framework with the commonly known approach of three-tier architecture. A web interface, where the user can access the patient cases and determine if they are infected by some disease has been built and is provided by the server application. The mobile client application was built using the Java Micro Edition (Java ME) framework resulting in a midlet application ready to be executed on Java enabled devices. The mobile application can gather both images and patient information which is uploaded to the server together with GPS data. The mobile application has been built so that it could be used offline; this enables the medical personnel to use the application in remote areas without connection to the mobile network. The collected data can then later be uploaded when entering an area with connection abilities.

When the patient case has been uploaded SMS's containing recent updates can be sent both to the medical personnel and the patient. When the image has been taken, post image processing can improve the picture quality in ways such as contrast and brightness. More advanced image processing techniques such as super resolution can most likely improve the image resolution further.

A field trial of the software applications was made in rural Ghana in November and, concluding that the software works also in the field.
Acknowledgements

The life as a student at the Royal Institute of Technology (KTH) is soon to an end and we would like to start to say thanks to our families and friends for their great support during these years. You made the joyful times more joyful and the stressful and agonizing times workable. Thank you!

Secondly, we would like to thank our supervisor Peter Håkansson at Ericsson for his guidance, support and motivation during the thesis. Ewert Linder and Johan Lundin at the Karolinska Institutet were our co-supervisors and without their support and time, the result of this thesis hadn’t been possible without your help. Thank you both!

The following people made it possible to reach our thesis goals:

Mats Pelltåck-Scharp at Sony Ericsson provided necessary hardware for the thesis that otherwise would have been much harder to get.

Sergei Popov, doctor at the Royal Institute of Technology helped us with knowledge in optics that was needed in the thesis.

Sebastian Elmgren at Ericsson made it possible to build the final prototype with results that otherwise hadn’t been reached.

Tobias Övergaard for his time during the modification of the CMOS sensor chip in the Electrum Laboratory.

Thank you all!
Contents

1 Introduction and background ................................................................. 11
  1.1 Mobile health ........................................................................ 11
  1.2 Situation in developing countries and rural areas ...................... 11
  1.3 The Web Microscope ................................................................ 12
  1.4 Conventional optical vs. On chip microscopy ......................... 12
  1.5 Thesis background, concept and motivation .............................. 14
  1.6 Thesis outline ........................................................................ 15

2 Thesis objective .................................................................................. 17
  2.1 Hardware problem formulation .............................................. 17
  2.2 Software problem formulation ................................................. 17

3 Project structure plan ......................................................................... 19

4 Explorative hardware experiments .................................................... 21
  4.1 Introduction ............................................................................ 21
  4.2 Huygens-Fresnel Principle ..................................................... 21
  4.3 Bayer filter ............................................................................. 23
  4.4 Theoretical digital resolution .................................................. 24
  4.5 Experimental equipment ......................................................... 26
  4.6 Components ........................................................................... 26
  4.7 Hardware experiments ............................................................ 30

5 Hardware results ................................................................................ 35
  5.1 Design of prototype ............................................................... 35
  5.2 Acquired resolution ............................................................... 39
  5.3 Detected parasites ................................................................... 41
  5.4 Ultra wide field of view .......................................................... 44

6 Software system requirements ............................................................ 45
  6.1 System overview ....................................................................... 45
  6.2 Requirements ........................................................................... 46

7 Software system concepts and design ................................................ 51
  7.1 Server Design Concepts ......................................................... 51
  7.2 Server Architecture ............................................................... 52
  7.3 Client Design Concepts ........................................................... 55
  7.4 Client Architecture .................................................................. 56
  7.5 Data structure .......................................................................... 57
  7.6 Interaction ............................................................................... 60

8 Software results .................................................................................. 63
  8.1 Server Application Implementation ....................................... 63
  8.2 Implementation of server requirements .................................. 64
  8.3 Client Application Implementation ......................................... 72
  8.4 Implementation of mobile client requirements ...................... 74
  8.5 User Interfaces ......................................................................... 83

9 Discussion ............................................................................................ 85
  9.1 Hardware discussion ............................................................... 85
  9.2 Software discussion ............................................................... 87

10 Conclusion ........................................................................................... 91
  10.1 Hardware ............................................................................... 91
  10.2 Software ............................................................................... 92
Abbreviations

- AMMS  Advanced Multimedia Supplements
- API    Application Programming Interface
- ASP.net Active Server Pages
- CCD    Charge coupled device
- Cell-ID Cell-Identification
- CLDC   Connected Limited Device Configuration
- CMD    Command Prompt
- CMOS   Complementary metal oxide semiconductor
- CPU    Central Processing Unit
- CSS    Cascading Style Sheet
- CYGM   Cyan, yellow, green and magenta
- DI     Dependency Injection
- GPS    Global Positioning System
- GSM    Global System for Mobile Communication
- HTML   Hyper Text Markup Language
- HTTP   HyperText Transport Protocol
- HTTPS  HyperText Transport Protocol Secure
- IE     Internet Explorer
- IoC    Inversion of Control
- IP     Internet Protocol
- Java EE Java Enterprise Edition
- JDBC   Java Database Connectivity
- JPEG   Joint photographic group
- JRE    Java Runtime Environment
- JSP    Java Server Pages
- JSTL   JavaServer Pages Standard Tag Library
- LAC    Location Area Code
- LCDUI  Liquid Crystal Display User Interface
- LOSMic Lens less On chip Shadow Microscopy
- LWUIT  Lightweight User Interface Toolkit
- MCC    Mobile country Code
- MCD    Millicandela
- mHealth Mobile Health
- MIDP   Mobile Information Device Profile
- MMAPI  Mobile Media Application Programming Interface
- MNC    Mobile Network Code
- MTC    Department of Microbiology, Tumor and Cell Biology at the Karolinska Institutet.
- MVC    Model View Controller
- MVP    Millennium Village Program
- PHP    Hypertext Preprocessor
- RAM    Random Access Memory
- RGB    Red, green and blue
- RGBE   Red, green, blue and emerald
- SE     Sony Ericsson
- SMS    Short Message Service
- SQL    Structured Query Language
- STH    Soil-transmitted helminths
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UV</td>
<td>Ultra violet</td>
</tr>
<tr>
<td>WHO</td>
<td>World Health Organization</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter one

1 Introduction and background

1.1 Mobile health

The mobile health industry is currently exploding and new applications arise frequently. The meaning of mobile health or mhealth is to practice health with help of mobile devices to monitor and transfer information about patients wirelessly. Mobile health is a technology branch of ehealth which is a collection of applications supported by electronics and by electronically communication means.

The possibilities with mhealth seem endless that saves time, money, the environment and lives. Medical personnel can produce a diagnosis quickly without meeting the patient, saving the patient’s travel time and the doctor’s precious working time. This means that patients that live remotely in rural and less populated areas more easily can access the health system however there are risks with mobile health that need to be considered. The information passed on to medical personnel must be sufficiently accurate and easy to interpret so that faulty diagnostics is kept to minimum [1].

1.2 Situation in developing countries and rural areas

The situation for health workers and patients in rural areas in developing countries is very different depending on clinic and country, regarding equipment, staff and patient budget. There are for example less than one physician per 10 000 people in Uganda and two per 10 000 in Ghana [2]. In Figure 1 pictures from a clinic in Mali and the Millennium Village Project (MVP) can be viewed.

Figure 1 Pictures from a clinic in the Ruhira MVP cluster in Uganda (A) and Tiby MVP cluster in Mali (B, C, D, E, F, G).
<table>
<thead>
<tr>
<th>Disease</th>
<th>Infected (million people)</th>
<th>Morbidity (million people)</th>
<th>Mortality (thousand people)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schistosomiasis</td>
<td>&gt;200</td>
<td>120</td>
<td>200</td>
</tr>
<tr>
<td>Roundworm</td>
<td>1450</td>
<td>350</td>
<td>60</td>
</tr>
<tr>
<td>Hookworm</td>
<td>1300</td>
<td>150</td>
<td>65</td>
</tr>
<tr>
<td>Whipworm</td>
<td>1050</td>
<td>220</td>
<td>10</td>
</tr>
</tbody>
</table>

1[3, 4], 2³⁴[4]

Table 1  Global prevalence’s of some parasitical diseases that are evaluated in this report. See appendix for more information about these diseases.

The table above shows the global number of infected people, but the largest prevalence is to be found in rural areas in developing countries. These diseases except from the hookworm disease have been evaluated in this thesis, more information about the diseases can be found in the appendix.

One of the largest parasitical diseases, malaria, infects around 250 million people every year and nearly 1 million of these people die. In Africa 20% of all childhood deaths are related to the disease [5]. The malaria parasite is too small to be identified with the current proposed hardware described in this thesis and is therefore not evaluated in the report.

1.3 The Web Microscope

The Web Microscope is a project making it possible to view highly magnified digitized microscope images of biological specimens. The proposed web interface makes it possible to zoom, pan and create annotations within the image. The system uses computer image algorithms that automatically count or recognize parasites within the sample [6].

This project has from the beginning been collaborating with the Web Microscope project. Together with the system proposed in this thesis, quality assured assessments of biological samples could be made.

1.4 Conventional optical vs. On chip microscopy

Through the years, conventional optical microscopy has been the ruling method for diagnosing parasitical diseases in the field [7, 8, 9, 3]. In this report the concepts of lens less on chip shadow microscopy (LOSMic) is discussed and essential factors of such a system is determined. A brief summary how the conventional optical microscope works is given in this section.

The optical microscope produces magnified images of specimen samples through systems of lenses. An optical microscope consists of two optical systems, the objective and the eyepiece. Within each of these systems a number of lenses create the desired magnification [10].
The total acquired magnification is a combined result of the two lens systems and is calculated by multiplying the individual magnifications of the systems.

There exist a couple of techniques to illuminate the sample. The easiest could be the bright field microscopy where the sample is illuminated from below and observed from above. There is also the dark field microscopy where the aim is to remove the unscattered light that didn’t hit the topical specimen. This can result in a high contrast image where only the interesting objects that are chosen are visible [11].

There are also the phase contrast microscopy, fluorescent microscopy, polarisation microscopy and differential interference microscopy [10].

The resolving power of the microscope is determined by the wavelength used, the material which the lenses are built of and the Numerical Aperture (NA) of the lens. The resolving power is calculated by:

\[ MRD = \frac{\lambda}{2NA} \]

\[ NA = n \sin(\theta) \]

\( MRD \) is the Minimum Resolved Distance which is interpreted as the maximum resolving power. \( n \) is the refractive index and \( \theta \) is the maximum angle in which light can travel into or out of the lens [11].

The on chip shadow microscopy uses no lenses and the light source is placed above the object. The shadow created by this constellation is gathered by a CCD or CMOS sensor. The sampled image is then viewed on a computer.
1.5 Thesis background, concept and motivation

The collaboration between Ericsson and Karolinska Institutet was initiated at a workshop on “e-Health in Low Resource Settings” arranged at Karolinska Institutet and sponsored by Ericsson. Experiments at Ericsson Research with an external lens “add-on” for magnified imaging with a mobile phone camera and the above mentioned microscopy imaging research at Karolinska provided an excellent starting point for this joint project. Also the involvement of Ericsson in mobile health within the millennium village project in Africa was a strong incentive to study these promising new imaging, image transfer and processing methods further.

The general concept of this M.Sc. thesis is to create a price worthy, fast, accurate and mobile system that could be used in rural areas in developing countries. There is a lack of a mobile solution that could give millions of people access to vital help. A disease like malaria, bilharzia, roundworm, hookworm and whipworm demands thousands of lives every year [12, 13, 3, 4]. There are rapid test’s available for some of the diseases but no one can substitute the need of a conventional microscope with its sensitivity and specificity. However, a conventional microscope is relatively expensive, time consuming to use, non mobile, has high power consumption and has service requirements. There is also a learning period before medical personnel can start to use it.

The proposed lens less on chip shadow microscopy concept solves problems with availability, diagnosing time, record keeping, mobility and cost. The conceptual idea implements a mobile device with LOSMic possibilities and connection abilities to a centralized server with automatic image analysis and possibilities for health workers to manually diagnose and report back to the patient (Figure 4). The biological specimens are mounted with well established techniques on conventional microscope slides.

In Figure 5 the conceptual process is illustrated. An image is captured by the LOSMic and together with patient data (1) is uploaded to the centralized server (2). The image is either automatically analyzed by defined algorithms or manually diagnosed by a health worker at the web client interface (3). The diagnosis and information are then attached to the uploaded data (4). The answer is then fetched by the mobile client from the server (5) and the information is transferred to the patient by the health worker (6).
1.6 Thesis outline

This thesis report will give the reader an introduction to on chip microscopy, the software systems supporting it and an insight of the importance of such a system. The many technologies used to implement the concept require a wide introduction where background and theory is given. This gives the reader important background information before going through the results and conclusions.

The report has been written to readers with general knowledge of programming and optics and little or no knowledge in biology and medicine.

The thesis concept deals with new areas of science and therefore the hardware requirements of the system are needed to be determined. These requirements will be described in the section about experiments in this report.

The scope and timeframe of this thesis required two students. Between these persons the work was divided into two separate parts where one worked with the hardware and the other one with the software. Because of the division of the thesis two problem formulations were set up.

Chapter 1 – Introduction and background:
A short introduction to mobile health and the situation in rural areas in developing countries is given. The thesis concept and motivation are also found in this chapter.
*Author: Jim and Tobias*

Chapter 2 – Thesis objective:
The objective and the problem formulation are declared here.
*Author: Jim and Tobias*

Chapter 3 – Project structure plan:
This project was divided into two parts; software and hardware. The project structure plan is summarized here.
*Author: Jim and Tobias*

Chapter 4 – Explorative hardware experiments:
On chip microscopy is a fairly new concept and before designing the final prototype, some exploratory experiments had to be done.
*Author: Tobias*

Chapter 5 – Hardware results:
The results for the hardware part containing the implementation of the final prototype and detected diseases is discussed.
*Author: Tobias*

Chapter 6 – Software requirements:
The requirements for the software part goes through here. Motivation for the choices is also given.
*Author: Jim*
Chapter 7 – Software system concepts and design:
When the software requirements had been specified then the software concept and its design was decided. This chapter goes through the different decisions.
Author: Jim

Chapter 8 – Software results:
The implementation for both the mobile client and the server application is described.
Author: Jim

Chapter 9 – Discussion:
The choices and results are discussed here for both the hardware and the software.
Author: Jim and Tobias

Chapter 10 – Conclusion:
This chapter concludes the project.
Author: Jim and Tobias

Chapter 11 – References:

Appendix:
The appendix explains the selection of desired detectable diseases which was evaluated in this thesis. Theory concerning parts that could be important in future works is described.

Pictures of the LOSMic prototype and the user interfaces can be found here.

The “list of figures” and “list of tables” can be found here.
Author: Jim and Tobias
Chapter two

2 Thesis objective

The aim of this M.Sc. thesis is to develop on-chip microscopy where the specimen are placed directly on a camera sensor chip to produce an image that can be analyzed by a computer, using a specially designed automatic image analysis algorithm and by visual interpretation. This includes finding a method for transfer of high resolution images of biological specimens over wireless mobile networks, where the sender is a mobile device and the recipient is an image web server.

The sent image of the biological sample should contain enough information to be able to make a remote diagnosis. An answer should then be sent back to the health worker so that the information and diagnose reaches the designated patient.

2.1 Hardware problem formulation

- Determine the minimum hardware requirements of the mobile device for transferring high-resolution microscope images via mobile networks. Resolution requirements and image sizes for computer vision algorithms will be determined in a parallel sub-project.

- Develop and build a prototype by either rebuilding a mobile phone or a digital camera. This with the use of as many commercially available components as possible.

- Test the prototype in a lab using biological specimens.

2.2 Software problem formulation

- Determine the software requirements needed to offer diagnosis of tropical diseases over distance using mobile applications and devices.

- Design and implement a mobile and a server application.

- Test and evaluate the system using real devices.
Chapter three

3 Project structure plan

This cross technology thesis demands knowledge in a wide range of disciplines, such as application development, parasitology and optics. To gather knowledge a pre-study phase initiated the project. Previous attempts were identified and evaluated. The information was gathered through scientific articles, books, from key persons and through relevant material on the internet.

The project was conducted as follows:

**Hardware project structure**

- **Pre-study phase and literature study**
  Literature about relevant optics, physical factors, parasitology, microscopy methods and hardware platform characteristics were gathered. A preliminary study of possible and desired detectable diseases were completed. Planning of experiments were conducted when the information gathering was sufficient.

- **Set requirements for the hardware system**
  Decide the requirements for the system that will be feasible to reach during the scope of the thesis.

- **Design of experimental set up and collecting of components**
  After the initial factors had been identified, the design and construction of the experimental set up were conducted. The set up was designed to enable the experiments planned in the preceding phase.

**Software project structure**

- **Pre-study phase and literature study**
  Study literature about server and mobile application development. Evaluate different techniques and programming languages to use. Develop small test applications to evaluate different techniques. Consider the requirements and design of the system and collect as much information as possible to meet these.

- **Set requirements for the software system**
  Decide the requirements for the system that will be feasible to reach during the scope of the thesis.

- **Determine the design of the software system**
  How to implement a server application and a mobile application? Choose how the system will look like. Specify the data model. Determine how the different part of the system will interact.
• **Experimental phase**  
Experiments that answered questions that were raised under the first phase was executed. Control and tests of live specimens were also done at the Swedish Institute of Infectious Disease Control to ensure that the right parameters were evaluated.

• **Design of final prototype**  
When the parameters had been set a more field worthy and durable prototype were designed and constructed at the Ericsson prototype lab.

• **Test and evaluation of prototype, resolution and observed diseases.**  
The final prototype was tested to ensure that the images matched the specifications from the initial experimental set up. Tests of usability were also fulfilled.

• **Implementation of the system**  
Implement the system that will meet the requirements and design that was set up during the previous phases.

• **Test and evaluation of the system.**  
Test the implementation on real devices to see if the system matches the hardware parts of the concept.

• **Report writing**  
This part of the project gathered the knowledge apprehended during the thesis which is described in this report.

• **Presentation and report hand in**  
The report was turned in and the presentation was conducted.
Chapter four

4 Explorative hardware experiments

4.1 Introduction

On-chip microscopy is a new concept and there is only a hand full of articles describing trials using the technique. The basics of on-chip microscopy had to be determined. This resulted in a number of experiments to answer some of the questions.

This section describes these questions and the experiment and calculations done to answer them. Many factors influence the end result to apprehend a quality image. The experiments had to be, to some extent, repeated when a new, parallel factor had been improved. Also, some experiments are directly related to each other and was therefore conducted simultaneously but divided into sections in this report to enhance the readability. To mimic the environment of a built in system, the experiments took place in a dark room with little background light radiation.

4.2 Huygens-Fresnel Principle

To understand how the shadow is created and why the image is larger on the surface of the sensor, a brief introduction to the Huygens-Fresnel Principle is needed.

If a planar light wave hit a small aperture, a diffraction pattern is obtained beyond the aperture. If the light wavelength is small relative to the aperture size the light will be diffracted in small angles beyond the aperture (Figure 6). If the opening is really small or a relative large wavelength is chosen, then the diffraction pattern will resemble a point source sending out spherical waves as seen in Figure 7 [14].

![Figure 6](image6.png) Short light wavelength relative to the aperture size.

![Figure 7](image7.png) Small aperture and relative large light wavelength.

If the aperture is replaced with an opaque object, the shadow of that object will behave in the same way (Figure 8). This means that after the light wave has illuminated the specimens in our glass the shadow of that object will spread out with an angle that is determined by its size and the wavelength of the illuminating light [14].
This implicates that if the distance between the object and the sensor is increased, the size of the shadow created from the object will be enlarged. However, when the topical distance is increased the intensity of the shadow will be decreased, creating a more blurry shadow. This is explained simply by the fact that the shadow is disseminated over a larger area.

This relation can be useful when designing the prototype. The designer has a choice to influence size, intensity and sharpness of the obtained image.

Huygens explains the wave front by implementing an array of point sources in the border where the incident light wave propagates through an aperture or onto an opaque object. Every point source then sends out spherical wave fronts. Superposition of these secondary point sources creates the wave front seen in Figure 10.

Remembering that it’s assumed that a planar light wave would hit the aperture or opaque object, the light source can now be discussed. A perfectly collimated light source would not have any radiation angle when it travels from the source to the object. This means that the angle of incident always is zero if the light source is placed vertically above the object with no angle. If the light source isn’t perfectly collimated, the light will hit the object or objects in different angles creating a larger scattering surface on the sensor. This means, as discussed above, that intensity and sharpness will be decreased.

It is obviously that a perfectly collimated light source would give us a better contrast and resolution of the illuminated object.
4.2.1 Huygens-Fresnel principle applied to the application

When discussing the principle it states that opaque objects in the path of an incident light beam will behave as an aperture with the same size. Applying this to a light beam propagating towards and through a medium with multiple specimen’s gives effects that is needed to be explained.

Imaging Figure 8 and Figure 9 and place these opaque objects randomly in the specimen medium (Figure 11). These secondary point sources resemble the situation in the specimen medium. All objects will be hit by the incident light beam which then send out a wave patterns similar to those shown in the figures.

![Figure 11](image1.jpg)

Figure 11 Multiple opaque objects on a microscope slide sending out spherical waves after that light has hit them.

4.3 Bayer filter

The CMOS or the charge coupled device (CCD) sensor is only sensitive to change in illuminating intensity and not in a difference in color. To be able to see colors with these sensors a mosaic like pattern is applied to the sensor surface. The individual photo detectors, which are observed as pixels on a monitor, all have a colorized filter at its top. This pattern is often called “Bayer pattern” [15] after its inventor Bryce E. Bayer working at Kodak at the time.

White light is a mix of all different wavelengths and the sensation of color is a result of absorption and transmission by the topical object. Red paint for example absorbs all wavelengths except for the color red that is instead transmitted and observed by the eye. A red, partially translucent filter, works in a similar way. It allows the red wavelengths to pass the filter and the remaining colors are reflected. This is the principle of the Bayer filter.

These filters only let the color through of which it’s colorized by. A red filter only let red colors hit the sensor and so on. Knowing how the pattern is configured an output image of various colors can be produced. In section 4.6.2 the chosen image sensor is discussed. The choice fell on the IMX043 sensor from SONY. The standard Bayer filter is seen in Figure 12 and it seems like the IMX043 sensor uses a very similar pattern.

![Figure 12](image2.jpg)

Figure 12 The IMX043 Bayer pattern configuration seen in 100x magnification. Colours enhanced in the upper left corner.
After the invention of the original Bayer pattern a series of different patterns have been developed. Some uses a cyan, yellow, green and magenta (CYGM) or a red, green, blue and emerald (RGBE) configuration instead of the original RGB colors. The mosaic like pattern also has been altered to fit individual requirements and some new patterns can be seen in Figure 14.

4.4 Theoretical digital resolution

The smallest distance between two objects, where a person still can distinguish that they really are separate objects, is the definition of resolution [14].

To be able to calculate the theoretical resolution reachable by the LOSMic, it’s needed to know two parameters. First, the individual pixel size and second how the Bayer pattern looks like and if it’s applied to the sensor surface.

It’s known that the pixel size is 1.75 µm and that the pixels are squares. The Bayer pattern is also known and looks like Figure 13. One unit cell of the Bayer pattern contains a square of four pixels with one red, one blue and two green pixels. This means that there is the double amount of green pixels on the chip. Section 4.2 discussed what happens with the shadow when using different wavelengths, this knowledge “force” the use of a monochromatic light source to illuminate the object. If white light would be used, the light couldn’t be collimated and the obtained picture would be blurry.

Because of the fact that the Bayer pattern is applied and that monochromatic light is needed, green light has to be chosen to be able to use as many pixels as possible.

The geometry of the pixels and how they react to light is important to know to be able to calculate the possible resolution.
In the left part of Figure 15, three rectangular objects are placed above the sensor matrix. The black objects are positioned between two rows of pixels and the third blue object covers a single row. When this arrangement is sampled by the sensor the picture will look like the right part of Figure 15. The sensor only detects intensity of the incoming light and because of the fact that the black object only covers a half of each pixel the result will be a gray image.

To separate two objects there must be a white row between the objects. The minimum number of pixels demanded are therefore three pixels for an object with equal or smaller size than a single pixel. With three pixels a white column is viewed on the sampled picture at some side of the object. If the object is larger than a single pixel more pixels are needed to ensure a white column at the side of the object in the sampled picture.

If two objects that are equal or smaller than a single pixel would be placed beside each other with a distance smaller than three pixels in length the viewer could not tell whether if is one or two objects. The conclusion is therefore that three pixels are needed to see one object.

In Figure 16 two objects can be observed in the right part even if there is no white row in between. Depending how sensitive the sensor is, two objects can be distinguished if the intensity difference is big enough (Figure 17), with two pixels as the minimum requirement. This is explained thoroughly by the Nyquist theorem [17].

\[ \text{Intensity}_1 \]

\[ \text{Intensity}_2 \]

<table>
<thead>
<tr>
<th>Detects two objects</th>
<th>Detects one object</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta I_1 )</td>
<td>( \Delta I_2 )</td>
</tr>
</tbody>
</table>

Figure 17 Sensitivity graph. The intensity is inverted since shadow imaging is discussed.

The theoretical resolution can now be calculated. Three pixels have to be used to differentiate two objects. Every pixel is \( p \) \( \mu \text{m} \), where \( p \) is the individual pixel size. Green light is used and therefore only half of the total pixels are used. The formula becomes:

Equation 3 \[ 3 \times p \times 2 = 6p \text{ } [\mu\text{m}] \]

The system can theoretically differentiate objects a distance \( 6p \) \( \mu \text{m} \) apart from each other. If the Bayer filter is removed, the theoretical resolution would be \( 3p \) \( \mu \text{m} \).
4.5 Experimental equipment

An experimental configuration was created to enable testing of various components and settings. The set up was made of a modified Sony Ericsson C905 where the cameras chip was exposed, a plate of acrylic glass with a slit for the sensor, a gantry which can change the light source height and angle and a plastic bridge which holds the light source (Figure 15). Black unreflecting tubes were created to remove unwanted background lights and mimic the environment of a built in system.

The configuration enables testing of various distances and angles between the light source and microscope slide. The acrylic glass plate holds the microscope slide and sensor in place during taking the picture. A voltage generator is used to supply the light source.

The camera chip module is normally designed with auto focus, optical zoom and a picture stabilization feature. This had to be removed to access the surface of the CMOS sensor. Directly above the sensor surface a protective glass is applied on a ceramic frame. The height of this construction was estimated to be around 1.5 mm. This distance was in theory too long to achieve good resolution and therefore the protective glass had to be removed. The glass was replaced with a modified cover slip which has the thickness of around 0.1 mm.

4.6 Components

4.6.1 Sony Ericsson C905

As the concept is to create a prototype which is cost efficient, durable and mobile, a mobile phone with all its usability such internet connection, low power consumption and easy programmable hardware, was chosen as a platform. The model chosen was the Sony Ericsson C905 which is designed to create high definition pictures with its 8M-Pixel camera. The C905 uses the Sony Ericsson Java Platform 8.4 [18]. This gives possibilities to implement JAVA Mobile applications. Since it’s needed to send pictures over the internet to a server the mobile phone has to have a HSDPA or a GPRS connection.

The model also includes a Global Positioning System (GPS). This gives the possibility to see where the picture is taken. The phone battery standby time is between 360 and 380 hours depending on connection [19].
Other platforms like an ordinary digital camera was also considered. The digital camera would contain the necessary image sensor specifications, but it lacks the flexibility like HSDPA and GPRS connection, that the mobile phone provides. The reason for choosing C905 was because of the characteristics of the CMOS sensor together with other useful product features.

### 4.6.2 Camera chip

The sensor used in this project is the Sony 8M-Pixel, complementary metal oxide semiconductor (CMOS) sensor IMX043 [20]. It’s designed to be used in cell phone cameras and therefore the camera chip is small to create high resolution images. The chip is made for color images and for that reason there is a Bayer pattern [15] applied on its surface. This feature is explained in section: 4.3. This sensor chip suits this project when an important part is to find a sensor with as many pixels as possible where the individual pixels are as small as possibly.

The active pixel sensor (CMOS) is built up of small photo detectors that when illuminated send out a readable current that varies with different illumining intensities. This signal is then processed in a dedicated image processor converting RAW image data to JPEG [21]. This built in process forces us to use a JPEG (Joint Photographic Experts Group) picture instead of RAW which could have been better for enhancing picture information after taking them.

The individual pixels are squares of 1.75 µm and there are $2468 \times 3288$ effective pixels. The total area of a picture is then: $5,754 \times 4,319$ [mm].

![Figure 19](image_url) **Figure 19** Method of measuring light beam incident angle [20].

![Figure 20](image_url) **Figure 20** Pixel Incident angle characteristics. The relative output at 0° is converted to 1.0 [20].

The individual pixels have a pixel incident angle characteristic that follows Figure 20. The pixel incident angle is measured like in Figure 19. It is clear that after approximately 10°, the registered intensity quickly drops. This means that if a light beam is directed towards the sensor at 25°, the output signal will be 70% lower than if directed from 0°. The drop continues down to around 30° when the output is stabilized around 30%.

Mounted on the circuit board on top of the sensor are lenses and auto focus mechanics. The closest component to the sensor is a protective glass which in total builds around 1.5 mm. This glass is glued on a ceramic frame which can be seen in Figure 21.

![Figure 21](image_url) **Figure 21** CMOS chip from above showing the protective glass on a ceramic frame.
4.6.3 CMOS sensor vs. CCD sensor

Neither a CMOS nor a CCD sensor is generally better than the other. Which is chosen depends on the situation and in which application it will be used.

The choice of a CMOS sensor instead of a CCD sensor has in this project several advantages. The CMOS technique is suitable for mobile phones when it uses less power than a CCD sensor [22, 23]. This implies that these sensors have been developed further to serve the mobile industry and it is then probable to find a chip that’s cost efficient and has many pixels on a small surface. The on-chip integration is also easier for a CMOS sensor than CCD and this is also important when it comes to size of the total on-chip circuit.

4.6.4 Glass Collimator

The LOSMic uses a glass collimator which consists of three lenses. The collimator with its lenses is used to enhance the light pattern from a LED. It’s not possible to apprehend perfect collimation with the use of a standard LED and this collimator. The used definition of describing the way of acquiring images explains that there are no lenses involved when magnifying the objects. If another light source could be found that don’t need a collimator then the system is completely lens less.

The collimator used in this prototype is the DLU2A glass collimator from Latronix AB. It is optimized for a wavelength of 670 nm but the purpose is to enhance the beam angle and the illuminated patter created by the chosen LED.

**Characteristics:**

- Focal length: 8.35 mm
- F/#: 1:1.6
- Back focal length: 5.09 mm
- Beam angle: 2.0 mrad
- Design wavelength: 670 nm

Other lenses were evaluated but the DLU-2A seemed to give the best result.

4.6.5 Light emitting diode

Light Emitting Diodes (LED) are cost efficient and very suitable light sources for this project. Fairly monochromatic light can easily be obtained and the power consumption is very low [24].

Different LED’s were tested during the experiments. The large number of tested LED’s depended on the uncertainty how the commercial sensor would react for this application and the ruling factors where initially unknown. In section 4.3 the maximum resolution demanded a green diode. The green dye was therefore extra tested during the experiments. Here is a list of them:
a. Color: 525 nm (green)
   Diameter: 5 mm
   Intensity: 12000 – 18000 mcd
   Beam angle: 10° – 13°
   Product number: 90589 [25]

b. Color: 527 nm (green)
   Diameter: 3 mm
   Intensity: 1500 – 2100 mcd
   Beam angle: 24° – 30°
   Product number: 90623 [25]

c. Color: 528 nm (green)
   Diameter: 1.8 mm
   Intensity: 7000 mcd
   Beam angle: 30°
   Product number: 90618 [25]

d. Color: 470 nm (blue)
   Diameter: 5 mm
   Intensity: 4000 – 8000 mcd
   Beam angle: 10° – 13°
   Product number: 90585 [25]

e. Color: 630 nm (red)
   Diameter: 5 mm
   Intensity: 4000 – 8000 mcd
   Beam angle: 17° – 23°
   Product number: 90587 [25]

f. Color: 627 nm (red)
   Diameter: 3 mm
   Intensity: 1500 – 3000 mcd
   Beam angle: 24° – 30°
   Product number: 90622 [25]

g. Color: White light, 6000 – 12000 K
   Diameter: 5 mm
   Intensity: 12000 – 15000 mcd
   Beam angle: 13° – 17°
   Product number: 90582 [25]

h. Color: White light
   Diameter: 5 mm
   Intensity: 4000 – 6000 mcd.
   Beam angle: 32° – 40°
   Product number: 90583 [25]

i. Color: 370 nm (UV)
   Diameter: 5 mm
   Intensity: 1 mW
   Beam angle: ± 55°
   Product number: 75-003-66 [26]

4.6.6 Microscope slides and cover slip

The microscope slides can vary in thickness and size. The specimen is placed on the surface of the slide which is mounted with the specimen facing the sensor. This positioning eliminates problems that occur if the specimen distance change with the thickness of the slide.

The refractive index ($n_1$) of the slides varies but appears to be around 1.5. This gives us a critical angle using Equation 4, where $n_2$ is the refractive index of the less optically dense medium, in this case air.

$$\theta_c = \arcsin\left(\frac{n_2}{n_1}\right) = \arcsin\left(\frac{1}{1.5}\right) = 41.8^\circ [14]$$

The cover slips that could be used also differ in thickness. The thickness directly affects the resolution of the system. A thin cover slip as possible should be strived for. The glass used for cover slips are assumed to resemble the slides, regarding material, giving them the similar refractive index.

Both the microscope slides and cover slips used for testing were bought at Menzel-Gläser. The slides were the Superfrost®Plus adhesive microscope slides with product number: J1800AMNZ. The cover slips have the product number BB021026A1 [27].
Microscope slide characteristics:
- Width 25 mm
- Length 75 mm
- Thickness 1 mm

Cover glass characteristics:
- Width 21 mm
- Length 26 mm
- Thickness 0.13 - 0.16 mm

4.7 Hardware experiments

4.7.1 Evaluating the light source

The type of light used, highly affects the resolution as seen in section 4.2. The theory speaks about the importance of green monochromatic collimated light to achieve the maximum resolution.

To see if this assumption was correct a series of tests were conducted with different colored LED’s. The construction of the LED with its wired bond from the anode to the reflective cavity of the cathode doesn’t create a perfect point source neither a perfectly collimated light propagation wave.

This forces us to do a compromise. A LED might give sufficient light characteristics for the application but not perfect. For achieving collimated light, lenses and a spatial filter are required. This produce a higher total cost of the product and the thesis aim for using commercial components begins to fade.

To enhance the light pattern from the LED’s the DLU2A collimator (section 4.6.4) was used. It turned out that with use of this component the light characteristics was enhanced so that a visual difference in resolution was acquired. Even though the collimator is designed to work at 670 nm the wave pattern was dramatically improved for lower wavelengths.

To test which light source that worked best, a reference glass of the type NBS USAF 1951 Test chart – R70 [28] was used. The glass has etched line pairs of chrome on its surface with different sizes and lines pairs per millimetre. The resolution is given by the smallest distance between two lines where these two separate lines still can be distinguished.
The light sources were then placed vertically above the reference glass at different distances. At each distance different LED’s was tested and the result was later compared on a computer. Tests showed that monochromatic light was superior multi wavelength light regarding the resolution. Red and blue gave similar results compared to green but the contrast was better with green in all trials. It appears that the sensor is more sensitive to green than other colours. Note that the LED’s in this trial were chosen to match each other when it comes to light intensity and beam angle characteristics. All LED’s were driven on their recommended typical voltage.

When differentiate biological specimens it can’t be assumed that all objects appear opaque. The contrary is more likely and this has to be evaluated. Beginning with a normal blond hair, it can be showed that the green dye brings out more structure than the red and blue LED (Figure 23 and Figure 24).

![Figure 23](image1)

Figure 23 Pictures of a piece of hair showing the difference between a green and blue LED. Not enhanced by computer.

![Figure 24](image2)

Figure 24 3D surface plot of Figure 23. Light intensity represents height but now converted when shadow imaging is discussed.

When comparing a white dye with a green it’s obvious that the white give more realistic colours but not the same resolution. This knowledge could be useful in situations when resolution is not the most significant factor for evaluating the specimen for example in cancer diagnostics and coloured tissues (Figure 25).

![Figure 25](image3)

Figure 25 Coloured tissue sample for cancer diagnostics. A: Green light, B: White light.
Numerous tests show that the green generates brighter and more contrast friendly pictures.

The LED operates between a lower threshold voltage and the higher maximum voltage. The intensity, and to some extent the wavelength of the LED changes when increasing the voltage from the lower to the higher value. The previous test was conducted when the typical voltage was applied, meaning the recommended operating voltage.

### 4.7.2 Distance between object and sensor

The distance between the sensor and the specimen mounted on the surface of the microscope slide is of great importance to the sharpness of the picture. This is explained earlier in section about the Huygens-Fresnel Principle (section 4.2).

Early tests showed that the resolution was very sensitive to the object-sensor distance and it was clear that this distance should be as minimal as possible to maximize resolution. The sensor used had, as mentioned in section 4.5, a protective glass that together with its ceramic frame built approximately 1.5 mm above the sensor. Early pictures showed that this distance was too long and that the glass had to be removed. To protect the highly sensitive surface of the CMOS sensor a protective glass cut out from a cover slip was applied instead. The distance now measured 0.1 mm. This creates a distance between the object and sensor of around 0.3 mm if a cover glass is used on the microscope slide. If this is removed the distance decreases to around 0.2 mm.

It was also clear that the way the microscope slide with specimen and cover slip was mounted on the experimental set up was of importance. Turning the slide upside down so that the cover slip faced the sensor directly, produced much more detailed pictures. In Figure 26 the difference is very clear, showing this relationship.

![Figure 26](image)

**Figure 26** Difference between an objective glass mounted with the pattern up away from the sensor (A) and with the pattern placed with the pattern upside down towards the sensor (B).
The correlation how large an object gets when increasing the distance between the object and sensor seemed to follow the explanation of Huygens-Fresnel. Figure 27 show how large the picture would get if only the spherical wave pattern assumption would dominate. The equation describes how the object size increases on the surface of the sensor with a theoretical 30 degree propagation wave from the object.

As seen in Figure 27 the distance decides how big the cast shadow will be on the sensor surface. The figure concur with the theory of Huygens-Fresnel (4.2.1) and how multiple objects in the specimen medium react to a greater distance between sensor and light source. That’s the reason why the resolution gets better if the distance gets smaller.

To apprehend images that have a resolution very close to what is theoretically possible the objects would have to be placed on the photo detectors' surface beneath the Bayer filter.

4.7.3 Sensor characteristics factors

The characteristics for the sensor regarding the sensibility of light incident angles have great effect when deciding the object-sensor distance. If assumed that an object behaves like a point source the picture would be very big on the sensor surface if the distance was relative big enough. The sensor sensitivity when light propagates from angles counter act this magnification. When light hits the sensor from an angle higher than 10 degrees the intensity drops rapidly (Figure 20). This implies that the edge of the picture resembling the object will have a drop in intensity making it blurry. This is observed in pictures which other wise would have shown objects far bigger than they are.
4.7.4 Distance between object and light source

The distance between the sensor and the light source would not be important if a perfectly collimated light had been successfully implemented. In that case the wave pattern would have been kept intact at all interesting distances. The intensity with a collimated light would also be fairly constant at the relative small distance shifts in this application.

The fact that a LED without a proper collimator forces us into a situation where the distance can affect the end result.

![Figure 28 Decrease of distance between object and light source with a round 525 nm green LED with 3.4 V applied.](image)

The different results have two great reasons. Because of the relative large beam angle from the light source the amount of light hitting the surface of the sensor will increase rather much when the source gets closer to the sensor. This gives the effect of a brighter image but also with the risk of over exposing the sensor. The objects that are partly translucent will also be overexposed removing their shadow. This can be seen in the last images in Figure 28 (C, D, E).

The first two pictures of Figure 28 indicate that the resolution is more constant between 23 cm and 11 cm than between 11 cm and 2 cm. The reason for improved resolution at larger distances could be explained by the wave patterns progression. The light source used, is far from collimated and the wave pattern resembles more a sphere than a planar wave. The spherical wave front will get bigger while propagating and if it gets big enough the sensor will experience the front as a planar incident wave. An example to understand this: When looking at a small sphere resembling the earth it is easy to see that its round. But, in the real scale the earth seems flat in a constrained area. This is the situation with the wave front.

As seen in Figure 28, 11 cm seem to be the perfect distance for the LED used. At this stage the source has been decided to be the LED 90589 [25].
5

Hardware results

5.1 Design of prototype

The design of the final prototype began when sufficient information was collected from the experiments. The image quality and resolution that were obtained during experiments should remain or be improved. It was quickly decided that it should be able to handle commercial microscope slides with and without cover slips.

The Sony Ericsson C905 (SE C905) is used as a hardware platform and a green diode is used to create the light source together with the DL-2A glass collimator [29].

This first section of chapter five describes the work behind the final prototype and its parts.
5.1.1 Camera chip housing component

This component holds the mobile phone (SE C905) and places the sensitive CMOS sensor in position. The yellow highlighted area (Figure 32) is the slit, fitted for the microscope slide.

![Side view of the camera chip housing component](image)

Figure 32 Side view of the camera chip housing component.

The red markers show the thoroughgoing holes that are used to mount the light tube holder component, which is explained in next section. The prototype was built in parts to enable cleaning and replacement of components if other techniques were found during and after the project.

![Top view of the camera chip housing component](image)

Figure 33 Top view of the camera chip housing component.

![Bottom view of the camera chip housing component](image)

Figure 34 Bottom view of the camera chip housing component.

Figure 33 and Figure 34 display the top and bottom view of the camera chip holder (marked blue). The green markers show 5 mm deep holes made for sprinters that later will hold up two lengthways going axles in the purple sections. These axles are fixating the microscope slide when the tube is pushed down onto the axes.

The camera chip was modified in a clean room at the Electrum laboratory [30] to ensure that no dust was trapped under the new thinner protective glass. The glass used was the same as in the experiments [27].

5.1.2 Light tube holder component

To mount the light source at the evaluated distance of 11 cm, a tube had to be implemented. A light tube holder component was therefore developed.
The light tube is pressed down into the light tube hole marked light blue in Figure 35. Axles mentioned in the camera chip housing component (Figure 37) are pressed down by the light tube. A claque is put in the hole to ensure that the tube and axles don’t crush the microscope slide (Figure 36).

This component is then mounted on top of the camera chip housing component by the thoroughgoing holes marked red.

### 5.1.3 Light tube component and tube lid

The light tube contains the green diode and the DL-2A collimator. The LED is powered by a custom made power package with two parallel wired battery packs consisting of three 1.5 V button cell batteries in series. A resistor of 47 ohm is connected before plugging it in to the switcher (Figure 38) and diode.

The tube is made of solid aluminum with a diameter of 40 mm. Inside 11 cm up in to the tube there is a smaller radius equal to the collimator radius (Figure 39).
The system uses a high intensive LED with a wavelength of 525 nm, which represents green color. The LED has a half spherical glass bulb acting as lens.

**Characteristics:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity</td>
<td>12000 - 18000 mcd</td>
</tr>
<tr>
<td>Beam angle</td>
<td>10° to 13°</td>
</tr>
<tr>
<td>Voltage (max)</td>
<td>3.6 V</td>
</tr>
<tr>
<td>Voltage (typical)</td>
<td>3.4 V</td>
</tr>
<tr>
<td>Current</td>
<td>20 mA</td>
</tr>
<tr>
<td>Wavelength</td>
<td>525 nm</td>
</tr>
</tbody>
</table>

The choice of this LED was based on our aim of creating a cost efficient device with commercial components. The LED isn't perfect but sufficient to solve our needs. The LED was bought at Kjell & Company (Product number: 90589) [25].
5.2 Acquired resolution

It was stated that the theoretical resolution would be 10.5 µm where the individual pixel size is 1.75 µm. To measure the acquired resolution the NBS USAF 1951 Test chart – R70 [28] was again used.

To measure the resolution an image of the test chart is taken. The last group of parallel lines where space between the lines can be distinguished is the maximum resolution. The group number is then compared to a chart where the resolution could be found.

![Figure 41 Image of NBS USAF 1951 Test chart – R70](image1)

![Figure 42 Group five is the last readable group of the chart.](image2)

Group 5 in the chart was the last visible group of lines so this section was further evaluated. Visually group 5 and line 3 are the last parallel bars distinguishable. This area was evaluated by a 3D surface plot where the height represents the luminance. The graph has been inverted when shadows are collected from the three lines (Figure 44).

![Figure 44](image3)

It is confirmed in Figure 44 that indeed three lines are seen. There are slopes at the end of the bars which are lower than the highest peaks. This could be explained by the fact that it is more surfaces at the ends that can create superposition when light hits the surface. The conclusion is that there are still enough resolution in group 5, line 3.

If instead group 5 and line 4 are evaluated the result is different (Figure 45). Peaks can be seen at the sides of the outer lines because of lacking interfering surfaces, but the middle peak has been significantly smaller.

![Figure 45](image4)

In Figure 46 no peaks that can be related to three bars can be identified.
Figure 43 3D surface plot of group 5, line 2. Clear visible peaks.

Figure 44 3D surface plot of group 5, line 3. There are still distinguishable peaks.

Figure 45 3D surface plot of group 5, line 4. The middle peak is gone.

Figure 46 3D surface plot of group 5, line 5. The three peaks are replaced by interferes peaks at the line edges.

It’s shown that the resolution is sufficient in group 5, line 3 (Figure 44). The associative table to the test chart [28] show that the resolution is close to 12.41 µm. The configuration has exhibit an 18% decrease of the theoretically possible resolution.
5.3 Detected parasites

Multiple tests of various diseases were conducted at the Swedish Institute of Infectious Disease control and the Department of Microbiology, Tumor and Cell Biology (MTC). Brief theory to these diseases could be found in the appendix.

Images can be modified with use of computer algorithms to enhance contrast and brightness so that they are easier to identify.

5.3.1 Bilharzia

Two species of schistosomiasis were evaluated. First a solution containing purified and concentrated S. mansoni eggs were tested (Figure 47). The solution was put on a microscope slide under a cover slip. The eggs were then put into the prototype and photographed with the green LED. Even though the pictures are far more out of focus than the images from the conventional microscope, the objects can still be identified.

![Figure 47 Comparison between on chip (A, C) and conventional microscope images (B, D).](image)

Figure 47 Comparison between on chip (A, C) and conventional microscope images (B, D).

![Figure 48 Multiple S. mansoni eggs taken with the on chip microscope.](image)

Figure 48 Multiple S. mansoni eggs taken with the on chip microscope.

It’s clear that the shape can be identified in the pictures taken with the LOSMic. The tail of the schistosomiasis eggs can also be distinguished. Figure 48 shows a larger view of multiple S. mansoni eggs.

When diagnosing S. mansoni the Kato-Katz [12, 31] method is widely used to separate eggs from faeces. A trial with frozen faeces from mice’s spiked with S. mansoni was conducted.
It’s still possible to separate eggs in the samples but because of all the faeces left over’s there could be problems with an object covering eggs (Figure 49).

The third trial with S. mansoni involved eggs in liver tissue (Figure 50). The method of extracting a tissue sample is used more in laboratories than in the field but is still interesting to evaluate.

The other species of schistosomiasis was S. haematobium which causes urinary infection (Figure 51). These eggs are slightly different than S. mansoni. The clinical way to diagnose S. haematobium is to search for eggs in urine and this test could show how valid the prototype is when used clinically.
5.3.2 Human whipworm

The football shaped Trichuris trichuria eggs are well defined and easy to spot in the sample (Figure 52). The specimens were mounted wet on a microscope slide.

![Figure 52](image)

Figure 52 Comparison between LOSMic images (A, C) and conventional microscope (B, D). Eggs are clearly separated.

There are eggs that have cracked for different reasons and these are harder to identify and separate from random materials because the shape of these is more random (Figure 53).

![Figure 53](image)

Figure 53 Cracked eggs of Trichuris trichuria.

5.3.3 Roundworm

The roundworm has eggs that are more circular in shape (Figure 54). The resolution is also here sufficient to observe the eggs.

![Figure 54](image)

Figure 54 Ascaris lumbricoides eggs. Comparison between LOSMic images (A, C) and conventional microscope (B, D).
5.4 Ultra wide field of view

When the specimen sample is placed on the sensor surface the total area which is sampled is as large as 24.85 mm$^2$. When comparing this to the area which is covered by a conventional optical microscope, it is clear that the acquired field of view by the hardware prototype is often many times larger. The field of view seen in an optical microscope changes with the used magnification and type of lenses, but to get a feeling how it differs some examples can be found in Table 2. When calculating these numbers, the following assumptions were made: Optovar: 1 and ocular magnification: 20x.

\[
\text{Equation 6} \quad \text{FieldOfView}_{\text{Eyepieces}} = \frac{\text{FOV}^\#}{(M_{\text{Objective}} \times M_{\text{factor}})}
\]

<table>
<thead>
<tr>
<th>Magnification</th>
<th>Field of view</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x</td>
<td>4 mm$^2$</td>
</tr>
<tr>
<td>20x</td>
<td>1 mm$^2$</td>
</tr>
<tr>
<td>40x</td>
<td>0.25 mm$^2$</td>
</tr>
</tbody>
</table>

Table 2 Field of view in optical microscopy.

When the field of view gets larger the number of times that the medical personnel needs to move the sample decreases. Also, if the field of view is large, an image analysis algorithm can work with a few numbers of images. Figure 55 shows the total field of view.

Figure 55 Total field of view. 24.85 mm$^2$. 
Chapter six

6 Software system requirements

This chapter will elaborate on what have to be included in a system in order to be able to offer diagnosis of tropical diseases on distance using mobile technique. How such a system should look like and what parts that are needed to be included will be identified. The parts have different requirements that will be explained and motivated why they are needed. These requirements were founded during this thesis in the pre-study phase according to the thesis aims.

![System topology](image)

6.1 System overview

In Figure 56 two main participators can be identified as the main parts of the system, the client part and the server part. The figure of the system overview clearly states that this is a distributed system and that it is a matter of client/server architecture. As seen in the figure the client could appear in two forms namely as a mobile client or as a web client. These two clients will operate in different ways where the mobile client will be an own application that will be able to interact with the server through a predefined protocol. The web client will just be a thin client whose interface totally will depend on what the server will offer, in this case web pages. The ideas behind the different parts of the system will be discussed below.

6.1.1 Mobile Client

The main objective of the mobile client is to offer an easy way to collect information about a patient for diagnosis purposes. This information will be sent to a server where the data will be stored for further inspection (via the web client). One key aspect is that the mobile client should be able to follow up on the sent data i.e. the diagnosis of the patient should be accessible from the mobile client in an easy way.

The mobile client will run a custom made application that will be able to collect the required data that later will be sent to the server. Because of the use of a locally running application, the mobile client could be considered as a fat client as most of the intended work will be made without connection to the server which means that the client could be used in areas where no connection is available.
6.1.2 Web Client

The main purpose of the web client is to offer a way to inspect the uploaded cases on the server. The server will present the uploaded cases as web pages that can be accessed from a standard web browser. From the web pages the users will have the ability to add diagnoses to the uploaded data by interpret attached information like images of a blood sample, short stories of patient physical state or health questionnaires. These diagnoses will later on be fetched by the medical personnel via mobile clients whom will inform the diagnosis to the patient.

6.1.3 Server

The objective of the server is to store information uploaded by the clients. The server will host a database where the data will be stored using a specially designed data structure. The server will process requests made by both the mobile client and the web client. Depending on which client it operates against it need to have the ability to deliver responses in two different ways. When operate with the web client, a web page should be the response whereas interaction against a mobile client should result in a protocol specified response that could be parsed in the mobile application.

When receiving uploading data the server should have the ability to analyze the attached image automatically by running a specially designed image algorithm. The outcome of the processing will be reported back to the client. Otherwise, a diagnosis will be made by the users by interacting with the web interface.

6.2 Requirements

During the pre-study phase of this project the requirements of the system were defined. These requirements will describe the required abilities of the three parts of the system. For each requirement a note about its presence is provided.

6.2.1 Mobile Client

- **Have the ability to in an easy and intuitive way gather information about a certain patient and attach adequate data so a diagnose can be made remotely**
  This is the main objective of the system. A key aspect is the usability of this process.

- **Authenticate abilities with the server**
  In order to upload data the client needs to authenticate with the server by specifying authentication data to ensure that only a registered user could use the system.

- **Ability to work offline**
  The process of gathering patient data locally in the application doesn’t require connection with the server and could be done offline. When data need to be uploaded a connection should be established with the server using the existing authentication process. This will minimize the data traffic for the client.

- **Attach a snapshot image to the patient data using the built in camera function**
  As this system will be used with the mobile microscope is should be easy to take a snapshot inside the application to ensure good usability.
• **Attach an image that exists locally on the terminal**
  Pictures could have been taken using another camera application. These images should be easy to attach to the patient data.

• **Tag the gathered information with GPS data**
  This will make it possible to view the creation spot of the data which if diagnosed could provide maps showing the geographical distribution of diseases.

• **Store gathered data locally on the phone until uploaded**
  As data could be gathered offline e.g. when no connection abilities are available, the data should be stored locally on the phone until uploaded. If the application is closed and there is unsent data available this should be stored on the phone and then restored when the application is later reused. The unsent data should be located in an outbox interface inside the application to make it easy for the user to manage the gathered information.

• **Fetch activity for uploaded data**
  When data is received by the server a diagnose process will take place. This process is either made automatically or manually, both procedures will arise some sort of activity in form of messages. The mobile will have the ability to fetch this information by using an inbox interface inside the client where the messages will appear.

• **Ability to search after uploaded data**
  To fetch the state of some uploaded data there should be an interface where a search request could be made. This request will include some patient parameters that could be match against the uploaded data e.g. first name, age or residence. The request will result in a list of matched data including information about diagnosis for the patients.

### 6.2.2 Web Client

• **Browse through lists of uploaded data**
  Both personal and other user’s uploaded data should be available for inspection. One way is to introduce the concept of groups and that only the data created inside the group is visible for members of the same group.

• **Upload patient information with attached data**
  In the same way as the mobile client there will also be a possibility for the web client to upload new data to the system. For instance, microscopic images could be located on a computer and by offer this interface, new data could be uploaded to the server in an easy way.

• **View the uploaded data in a user friendly way**
  The uploaded data should be presented to the users in a user friendly way. There should be easy to get a complete overview of existing data like images of blood samples, creation spots and recent activity. The information about which patient it concerns should also be provided.

• **Add images to existing patient data**
  There should be an option to add new images that concern an existing patient that will help to determine the diagnosis of the patient.
• **Diagnose uploaded information**
  When enough information is gathered about a patient, a certain diagnosis could be made and an easy way of reporting that to the system is required. This will generate a message that will be stored along with the data and could be fetched by the mobile client.

• **Add messages to patient data**
  Maybe a user has certain information about the provided patient data. A message could then be feasible to attach. This message will be fetched by mobile clients.

• **Send SMS to patient or health worker**
  When uploading data to the server the phone number to both the patient and the creator (medical personnel) could be provided. The web interface will offer a way of sending an SMS to either of them.

• **Zoom and pan uploaded images**
  The provided images should be presented in a user friendly way and offer zooming and panning abilities.

6.2.3 **Server Application**

• **Store uploaded data from either a mobile client or a web client**
  The uploaded data will be stored in a database hosted by a server. Following a specially designed structure, specified in section 7.5, the data will be stored in different tables.

• **Offer two types of interfaces, web and mobile**
  When interacting with a web client the response from the server should be formatted as web pages when requests arrives from the mobile client the response should be formatted in a way that the mobile application understands.

• **Automatically process received images**
  When receiving images the server should provide processing abilities. In this kind of system several different processing algorithms could be offered like detection of malaria parasites in erythrocytes, parasite recognition algorithms or parasite count algorithms. When uploading data to the server, clients can specify what kind of algorithms that should be processed. This will generate some sort of outcome that will help diagnosing the patient. There could also be the possibility that the process algorithm is secure enough to diagnose the patient which will make the process completely automatic.

• **Show the creation spot of certain uploaded data on a map**
  When uploading the data some sort of position data will be provided which will make it possible to show the creation spot on a map.

• **Retrieve GPS data (longitude/latitude) for Cell-Id and IP-addresses**
  Depending on the uploading client different types of location data will be provided. In case of a mobile client, longitude and latitude could be attached directly if a built in GPS is available at the client terminal. Cell-Id could also be provided which require a look-up. In case of a web client a lookup on the IP-Address could be done.
• **Send SMS to either a patient or a health worker**
  Generate a request at the server which will result in a SMS to either the patient or the health worker, more about this in 8.2.4.

• **Ability to search among cases**
  Search request will be matched against data stored in the database and the database will respond with matched data.

• **Provide admin interface**
  An administrator should have the ability to create and remove users and groups. Uploaded patient data should be easy to browse and remove.
Chapter seven

7 Software system concepts and design

In section 6.1 two parts were identified, the client and the server. When implementing a large server application and a mobile application there are some design concepts that need to be considered. The data that will be involved in this system will be stored at the server using a defined structure that will be presented in this chapter. Finally the protocol describing how the client and server will interact will be described.

7.1 Server Design Concepts

When developing a server application the architecture and design are two parts that need to be elaborated. One decision that needs to be made is how the architecture of the application should be structured. When developing web server applications the natural choice is that the architecture follows a multi-tiered approach [32]. The goal when using multi-tier architecture is to split the application into different parts (tiers) where each part has different assignments. The number of tiers to use depends on the complexity of the application; by increasing the number of tiers, every tier will get more specific assignments. In this system, where a web application will be developed, three tiers are a good approach.

7.1.1 Three-tier architecture

Three-tier layer architecture is a design pattern where the approach is to split the application into three parts where each part has one of the following assignments.

- **Presentation Layer**
  This tier has the responsibility to display the user interface. This tier will display information for the user and also generate actions to the logic layer upon user interaction. In web application the basic choice for the presentation tier is dynamic generated HTML code. The receiving web browser will parse HTML code and display a user interface from which requests could be created. The presentation layer could also be an own application with its own graphical user interface that is partially built up by interacting with the logic tier.

- **Logic Layer**
  The logic tier is the heart of the server application. It will receive requests from the presentation layer that will be processed and result in some sort of response back, the generation of the result may include interaction with the data layer. The logic tier is usually developed using some type of platform such as: Java EE, ASP.net or PHP.

- **Data Layer**
  Stores all the data in the application by using a database. This layer receives queries from the logic layer and delivers answers containing data from the store. The data layer is often realized using a relation based database such as MySQL.
Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and result to something the user can understand.

Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier
Here information is stored and retrieved from database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

Figure 57 Three Tier Architecture.

7.2 Server Architecture

For this server application Java Enterprise Edition [33] is chosen as the platform. Java EE could be seen as an extended version of the Java Standard Edition. The extension of Java EE is for the purpose of using it when developing enterprise applications like web applications. By including several new APIs and tools like Java Server Pages (JSP) [34], Servlets [35] and Java Database Connectivity (JDBC) [36] the three layer architecture could be realized using one platform.

A servlet [35] is a specified class in Java EE which provides an interface for creating server applications that follow a request-response model. The generic use of a servlet is to handle HTTP requests in order to generate dynamic web data. For this purpose there exist a custom made servlet named HttpServlet [37]. By using the HttpServlet class a framework is provided for support of the HTTP protocol. The basic use of HttpServlet is to extend the class and then override methods that the HttpServlet provides. To fully support the HTTP protocol these methods need to be overridden.

- doGet()
  This method will receive all GET requests that are directed to this servlet.

- doPost()
  Handles all incoming POST requests to the servlet.

- doGet() Handles PUT requests

- doDelete() Handles DELETE request
The servlet will be managed by a web container running on a server. A specified servlet code will be loaded into the container who will instantiate the class and hold a reference for incoming requests. Incoming requests from a web browser or other clients will be handled by the server. These requests will be mapped to the requested servlet, a new thread will be created for this client and the request will be passed on to the servlet who will process this request inside the thread. After the request is processed the servlet will generate a response which the server will redirect to the client. This process can be viewed in Figure 58.

![Servlet Process Diagram](image)

Figure 58  Servlet Process.

To avoid having HTML code inside of the servlet Java EE offers a module named Java Server Pages (JSP) [34]. A JSP file contains HTML together with special tags named Java Server Pages Standard Tag Library (JSTL) [38], the usage of tags enables to build web pages with dynamic generated information; the HTML in the JSP file remains static whereas the tags will be exchanged with dynamic data when processed on the server.

A request could be passed to a JSP file containing information that will give a certain outcome of the tags. The chosen JSP file will then be parsed by a Servlet Engine that resides in the web container on the server. When the file is parsed the engine will generate a file containing servlet code. This servlet code will be compiled and the instantiated which will generate a HTML response that will be sent to the originated requester.

![Generate a response using JSP Diagram](image)

Figure 59  Generate a response using JSP.

As seen in Figure 59 a servlet that receives a request from a web client are able to redirect the request to a JSP that will be transformed to a new servlet. The transformed servlet will then deliver the response to the web client.
Java Database Connectivity (JDBC) [36] provides an interface to access databases from Java applications. By loading a correct driver, depending on the database brand, a connection could be established. With this connection the update statements like SQL UPDATE, CREATE, INSERT and DELETE could be invoked, update statement will report the outcome of the update in form of returned integers that indicates how many row the statement affected. Queries like SQL SELECT could also be invoked and which will return a set of data. JDBC connection could be used from the servlet where it can access data upon a request from a client. The data will then be passed to a JSP page that will be transferred back to the client.

In order to store the data model at the server side some sort of persistent storage is needed. Database storage can be realized using different techniques and one of these is SQL based databases [39]. SQL is basically a language that most of the contemporary relation based databases are using. Relation based database structures the data in different tables also called relations. Within a relation the data can be viewed as a set of tuples, where the tuple has a set of attributes. In a table this can be seen as the attributes is represented by the columns where each column has a certain data type e.g. integer. Each row in the table represents a tuple. The data can be modified using different queries defined by the SQL language. In order to modify a special tuple in the relation one of the attributes needs to be uniquely identified, also called a primary key.

![Figure 60 Relation in a relation based database.](image)

To achieve the three layer model the Java EE platform will be used with the following approach:

- As a presentation layer HTML (CSS/JavaScript) will be used for the web interface. The HTML will be generated using JSP and JSTL [38]. Several JSP will be built up where each JSP is responsible for a certain view of the web application. The mobile client will be responsible for its own user interface and will only receive custom generated data.

- The logic layer will be represented by a single Servlet. This Servlet will handle incoming requests from either a web browser or the mobile client. The servlet will include several help functions that will help process the request and deliver a proper response. To interact with the data layer help classes will be implemented and placed in the context of the servlet, the classes will be connected to the data layer by using JDBC.

- The data layer will be represented by a MySQL database.
7.3 Client Design Concepts

Compared to ordinary computers, mobile devices have some limitations like small memory sizes, small display screen, less CPU power and limited battery capacity. These limitations need to be considered when developing an application for such devices. Mobile devices of today are generally based on some sort of platform which often implies a general way of implement applications. For this system a mobile device has been chosen for the hardware part (see section 4.6.1). The chosen mobile device is the Sony Ericsson C905. For this specific device the common choice of implementation platform is Java Micro Edition (Java ME) because of the support of Sony Ericsson Java Platform 8 (8.4) in this device.

Java ME is a platform containing specifications and techniques in order to construct applications for mobile or embedded devices [40]. This platform could be seen as a stripped version of Java SE where the focus is to create a Java Runtime Environment (JRE) that could be executed on a device with limited abilities such as mobile phones. Java ME is built up containing two basic elements namely; a configuration and a profile.

- **Configuration**
  A configuration contains basic libraries that could be used for many different devices. Connected Limited Device Configuration (CLDC) is a configuration in Java ME whose purpose is to be used on small devices.

- **Profiles**
  A profile is a more specific set of libraries with the intention to be used with a certain device for example a mobile phone with limited capabilities. Mobile Information Device Profile (MIDP) is a widely used profile when creating applications for mobile phones.

When developing Java ME mobile applications a combination between CLDC and MIDP is often used. This gives a complete setup of the API’s needed to create an application with the purpose to be executed on a device with limited capabilities. In Figure 62 the different layers of a general MIDP application is shown.
Sony Ericsson Java Platform is a specific version of Java ME that includes both MIDP and CLDC libraries. Together with these libraries there are some extra features that have been added which only support Sony Ericsson devices. This creates a platform which is ready to implement a mobile application using Java ME.

### 7.4 Client Architecture

The basic building block in a Java ME (MIDP/CLDC) application is the midlet. The midlet defines the application and offers methods for interaction with the environment where the application is executed. The current state of the midlet is handled by an application manage software. By invoking methods specified in the midlet the state will be changed, this is described in Figure 63.

A midlet is generally controlled by a user using some sort of user interface. Liquid Crystal Display User Interface (LCDUI) is a part of Java ME that offers an easy way to create user interfaces for midlet applications [41]. Another way of implement user interfaces is to use Light Weight User Interface Toolkit (LWUIT) which is an external toolkit providing more advance graphical methods that LCDUI [42]. Developing user interfaces with LWUIT is similar to the techniques used in Swing, features like pluggable look and feel, layout managers and transitions between forms are all supported by LWUIT.
In Figure 64 an overview of the mobile application is presented. The mobile application will be built up using a midlet that will receive requests from the user interface, LWUIT. A request will result in an invocation of a task class that will process the request and deliver a result to the midlet who informs the new state to the appropriate view.

The task classes will have specific assignments for example, retrieve GPS position or retrieve messages. More information about these classes will be found in section 8.3.

7.5 Data structure

When building a system containing a centralized server that will communicate with different types of clients a common view of the data structure is a key aspect. In this system there will be several data models that both clients and the server needs to handle.

7.5.1 Case

The most important data that will be handled in the system is the patient data with attachments. This data will further on be referred to as a case. A case will include information about which patient it involves as well as data for diagnosis. Example of data for diagnosis could be microscopy images, a short story of the patient’s current state or an answered health questionnaire. The information about which patient the case handles is kept simple and only requires attributes such as first name, last name, age, village and gender. There are also some optional attributes that can be submitted, Table 3 gives a complete overview of the case representation.
### Table 3  Overview of attributes in the case representation.

<table>
<thead>
<tr>
<th>Parameter (* Required)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case ID *</td>
<td>A unique number that represent a specific case</td>
</tr>
<tr>
<td>First Name *</td>
<td>Given name of the patient</td>
</tr>
<tr>
<td>Last Name *</td>
<td>Family name of the patient</td>
</tr>
<tr>
<td>Age *</td>
<td>Age of the patient</td>
</tr>
<tr>
<td>Village *</td>
<td>Current location of patient residence</td>
</tr>
<tr>
<td>Sex *</td>
<td>Patient gender</td>
</tr>
<tr>
<td>Phone Number Patient</td>
<td>Mobile phone number to the patient</td>
</tr>
<tr>
<td>Phone Number Health Worker</td>
<td>Mobile phone number to the health worker</td>
</tr>
<tr>
<td>Creator</td>
<td>The unique key that represent the user that created the case</td>
</tr>
<tr>
<td>Group ID</td>
<td>The unique id of the group which the creator is a member of</td>
</tr>
<tr>
<td>Latitude</td>
<td>The latitude of the creation spot</td>
</tr>
<tr>
<td>Longitude</td>
<td>The longitude of the creation spot</td>
</tr>
<tr>
<td>Confirmed</td>
<td>Boolean value if the patient of this case has a confirmed disease</td>
</tr>
<tr>
<td>Suspicious</td>
<td>Boolean value if a user suspects that the patient has a certain disease</td>
</tr>
<tr>
<td>Healthy</td>
<td>Boolean value describing if the patient is confirmed healthy</td>
</tr>
<tr>
<td>Disease</td>
<td>The name of the confirmed or suspected disease, if the patient is healthy this is empty</td>
</tr>
</tbody>
</table>

One important parameter in the case representation is the Case ID. The Case ID is a unique reference given to the case by the server when the case data is received. In order to update a specific case this ID is needed as reference. It can also be used to reference other data such as images or activities to a specific case.

#### 7.5.2 Images

Images are handled as a separated representation due to the fact that the images are always connected to a particular case. The reason to keep it as an own representation is for the simplicity of adding new images. To refer a newly arrived image to a specific case it is attached with the reference key (Case ID). This will also make it simple to retrieve all images for a certain case; simply ask for images for a given case ID. Table 4 shows the attributes for the image representation.
### 7.5.3 Activity

During the lifetime of a case, events will arise. Examples of different events could be that a new message arrives, a diagnose report is added or a new image is added. These events are stored in a data model named activities. Activities will be bound to a certain case by using the case ID. All activities will be visible for the group members; this is achieved by adding the ID of the group. To describe the activity a message will be added, this could be a system generated message telling that a new image has been added or a message composed by a user regarding the case. Table 5 describes the attributes needed for the activity representation.

<table>
<thead>
<tr>
<th>Parameter (* Required)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Created *</td>
<td>Time when the activity occurred</td>
</tr>
<tr>
<td>Account ID *</td>
<td>The identification key to the creator of the activity</td>
</tr>
<tr>
<td>Case ID *</td>
<td>The case ID of which this activity belongs to</td>
</tr>
<tr>
<td>Group ID*</td>
<td>The Group ID of which this activity belongs to</td>
</tr>
<tr>
<td>Account Name *</td>
<td>The name of the creator of the activity</td>
</tr>
<tr>
<td>Message *</td>
<td>A message representation of the activity</td>
</tr>
</tbody>
</table>

Table 5  Overview of attributes in the activity representation.

### 7.5.4 Users and groups

To have the right to use the system a user account is required. There are two types of users; regular users and administrators. The regular user is a member of a group and does only have the ability to view its own cases and the group cases along with activities and images connected to the cases. The administrator has control over all cases, all users and all groups. It also has the ability to create new groups and users as well as remove these. Attributes for a user is described in Table 6.
<table>
<thead>
<tr>
<th>Parameter (* Required)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account ID*</td>
<td>A unique key assign when created</td>
</tr>
<tr>
<td>Name *</td>
<td>A unique name, used as username for authentication</td>
</tr>
<tr>
<td>Password *</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>Group ID *</td>
<td>The Group ID of which this user is member of</td>
</tr>
<tr>
<td>isAdministrator *</td>
<td>Describe if user is an administrator (True/False)</td>
</tr>
<tr>
<td>isBanned *</td>
<td>Describe if user is banned (True/False)</td>
</tr>
</tbody>
</table>

Table 6 Overview of attributes included in the account representation.

All users need to be a member of a group. Cases created by a certain user will also be visible for all members of the group as the user is member of. This will make it possible to receive feedback on personal cases from other users and also the ability to diagnose other user’s cases. The group representation only includes two attributes, a name and a unique id, described in Table 7.

<table>
<thead>
<tr>
<th>Parameter (* Required)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name *</td>
<td>Time when the activity occurred</td>
</tr>
<tr>
<td>Group ID *</td>
<td>The identification key to the creator of the activity</td>
</tr>
</tbody>
</table>

Table 7 Attributes included in representation of a group.

### 7.6 Interaction

The server application offers two interfaces, one against the mobile application and one against the web client. Both clients use the HTTP protocol but the data within the request and the responses differs.

![General Interaction between mobile and server](image)

Figure 65 General Interaction between mobile and server
In Figure 65 an overview of the general interaction between a mobile client and the server application is shown. The request sent from the mobile client will always be formatted as an HTTP message containing headers and in some cases also a body. The response from the server will be plain text which the mobile client application is able to parse. In Table 8 the different actions that the mobile client will invoked is described.

<table>
<thead>
<tr>
<th>Action</th>
<th>Request Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>HTTP Post</td>
<td>The username and password will be specified in the header of the HTTP request. After validation on the server a positive or negative response will be sent in plain text.</td>
</tr>
<tr>
<td>Upload Case</td>
<td>HTTP Multipart Post</td>
<td>The request will be formatted as a HTTP Multipart [43] Post message including all case parameters. The response from the server will be an acknowledgement that the case was received.</td>
</tr>
<tr>
<td>Get Case</td>
<td>HTTP Post</td>
<td>Will specify in the header of the request the ID of the wanted case. The response will be plain text including the data of the specific case.</td>
</tr>
<tr>
<td>Get Activities</td>
<td>HTTP Post</td>
<td>Will specify the username and password in the header of the request which will result in a response from the server containing all group activities, for the specified user, in plain text separated with a boundary.</td>
</tr>
<tr>
<td>Search Request</td>
<td>HTTP Post</td>
<td>Will specify in the header of the request the following case parameters of a wanted case; case ID, first name, last name, village and age. All parameters are optional and null is accepted as a value for any of the parameters. The response will include matched cases on the server separated with a boundary.</td>
</tr>
</tbody>
</table>

Table 8  Mobile Client Protocol Specification.

The web client will operate in a different way compared to the mobile client. The server will provide web pages from which the web client will create requests. One big difference between the mobile client and the web client is that a web session will be initiated for the web client.
When a web user access the server for the first time, a login view will be presented, where the user can specify the authentication data. This data will be validated on the server and in case the validation succeeds a session will be created including following data: name, account id and group id for the authenticated user.

After a user is authenticated web pages with dynamic generated data depending of the authenticated user will be presented. These pages include clickable objects that redirect the user to expected views. Invocation of a clickable object will result in a HTTP post request containing specific parameters. Requests will be processed on the server and along with the session data result in a dynamic generated HTML view shown in Figure 66.
Chapter eight

8 Software results

8.1 Server Application Implementation

In section 7.2 the motivation for why Java EE was selected as implementation platform was described. This chapter will focus on how the server application was implemented with the model and the concepts from section 7.2 in consideration.

The server side application is developed using three-tier architecture, discussed in 7.1.1. By using this architecture the application will be separated in three layers where each layer are responsible for a certain part namely; presentation, logic and data management. In Figure 67 all classes included in the server application is shown as well as the relationship between them.
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHServer</td>
<td>The main servlet class, handles all incoming requests and processes them in own methods which could imply access to the data layer by JDBC help classes. Sends an appropriate response depending on client type.</td>
</tr>
<tr>
<td>ServletListener</td>
<td>A listener class to the main servlet. This class is responsible for creating a session for incoming request and provides references to the help classes for the main servlet. When starting the server application help classes are instantiated and put into the context of the servlet by this class.</td>
</tr>
<tr>
<td>AccountManager</td>
<td>Is responsible for both the account table and the group table in the database. Provides methods for creating new accounts and users as well as deleting existing ones. Offers a method for the main servlet for login request which are checked against the accounts table in the database by this manager. Can retrieve account information, complete account lists, group information and group lists.</td>
</tr>
<tr>
<td>ActivityManager</td>
<td>Handles all activities in the database. Stores new activities and provides different lists of existing ones upon request from the main servlet.</td>
</tr>
<tr>
<td>CaseHandler</td>
<td>Handles all cases in the system and stores them to an appropriate database table. Provides methods such as; search requests, add diagnoses, list of personal cases, list of group cases etc.</td>
</tr>
<tr>
<td>ImageHandler</td>
<td>Stores images in a database table for a specific case. Provides methods that can respond with a list of existing images for a requested case.</td>
</tr>
<tr>
<td>JSP Files</td>
<td>Each JSP file represents a view for the web interface. The main servlet will set parameters upon a request from the user and will redirect those to a selected JSP file (view). The JSP file will process the parameters and generate a HTML page that is sent to the user.</td>
</tr>
<tr>
<td>Account</td>
<td>A representation of an account on the server holds account data for a specific account.</td>
</tr>
<tr>
<td>Group</td>
<td>A representation of a group on the server holds group data for a specific group.</td>
</tr>
<tr>
<td>Image</td>
<td>A representation of an image on the server holds image data for a specific image.</td>
</tr>
<tr>
<td>Case</td>
<td>A representation of a case on the server holds case data for a specific case.</td>
</tr>
<tr>
<td>Activity</td>
<td>A representation of an activity on the server holds activity data for a specific activity.</td>
</tr>
</tbody>
</table>

Table 9 Description of server classes.

8.2 Implementation of server requirements

In chapter 5 the requirements for the server was defined. How those requirements where implemented will be described in this section as well as the web client requirements due to the fact that the server has full responsibility for those.
8.2.1 Store uploaded data

The server need to store the data and give a response that it has been received.

As described earlier the main servlet has different help classes that provide methods for storing different types of data; AccountManager, ActivityManager, CaseHandler and ImageHandler. When requests like; new case, add message or add diagnose arrives at the server the uploaded data is parsed and validated by the main servlet. The retrieved data is then sent to the appropriate help class who stores it in the database.

Some of the uploaded data could only be sent from the web client, referring to requirements such as; Add images to existing patient data, Diagnose uploaded information, Add message to patient data

These different actions will be responded with a new web page view whereas the mobile client who only collects patient data will be responded with an acknowledgement that the data has arrived.

8.2.2 Interfaces

The server provides two different interfaces, one for the web client and the other for the mobile client.

This requirement is achieved by the protocol that is defined for this application, see section 7.6. When an incoming request arrives at the main servlet it will first check the headers and look for a specific parameter telling that it is a mobile request. If the parameter is found the requested action is processed and a response is sent back to the mobile client according to the protocol. If the parameter isn’t found it’s a web request and according to the protocol the response back will be a web page.

Provide an admin interface
This requirement is achieved by storing an admin parameter for the account in the database. If a user enters the system via the web interface and it is shown that it got admin rights then special web pages will be provided.

Browse through lists of uploaded data
This requirement is achieved by the web pages. The server stores session data for all authenticated web clients. The provided web pages like account view and group view will only show cases that belong to the specific user or group.

8.2.3 Search Requests

Depending on type of client this request will be handled in different ways. Both the web client and the mobile client will specify search parameters that will be matched among stored cases at the server. When a search request arrives it will be sent to the search method in the caseHandler class that will return a list of matched cases. The list will either be sent back as a web page in the account view or group view depending on where the request where arisen. If it was a mobile client it will be sent back according to the protocol, see section 7.6.
8.2.4 Send SMS

The server needs to be able to send an SMS to a specific person.

Ericsson Labs SMS Send & Receive API [44] is used for sending and receiving SMS by interacting with a web interface. When creating a case the mobile phone number for both the patient and the health worker can be attached. The server then has the ability to send SMS to these persons in various situations. Below are some situations where an SMS could be sent:

- Telling the patient to go visit their local health care center.
- Respond with case ID to the patient when uploading a new case to the server. This can be used for easy access of the case for further visits at the health center.
- Remind the patient to take a medicine.
- Report a changed state of a case to the health worker.

In order to send an SMS the server application needs to interact with a web interface hosted by Ericsson Labs. A request is sent to the address http://sms.labs.ericsson.net/send with some parameters added, these parameters are described in Table 10.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key (required)</td>
<td>The API key generated by Ericsson Labs needed for authentication of the requester</td>
</tr>
<tr>
<td>To (required)</td>
<td>The phone number to the phone who will receive the SMS (MSISDN)</td>
</tr>
<tr>
<td>Message (required)</td>
<td>The message part of the SMS, should be encoded as UTF-8</td>
</tr>
<tr>
<td>Type (optional)</td>
<td>The SMS could be sent as a flash SMS that is only showed once on the recipient’s phone. The SMS will not be stored on the phone, simply add the value flash to the parameter to send a flash SMS. To send an ordinary SMS simply skip this parameter.</td>
</tr>
</tbody>
</table>

Table 10 Parameters included in the SMS Send & Receive Request.

Below is an example of a request (using HTTP GET):

- http://sms.labs.ericsson.net/send?key=12345&to=46705123445&message=hello

8.2.5 Retrieve longitude and latitude

The server needs to be able to retrieve position data for different situations.

One way of retrieving the position of a case is to attach the Cell-Id information to the case. A cell is a radio base station that is available for mobile communication. Every radio base station has a uniquely identification number attached to it known as Cell-Id. When a mobile phone is connected to a specific cell this ID could be fetched.

To get longitude and latitude coordinates of the cell some sort of lookup needs to be made; this is what Ericsson Labs Mobile Location API [45] offers. By interact with a web interface by sending a properly formatted HTTP GET requests the subsequently respond will include the longitude and latitude of the cell, if the requested cell exist in the database. At the moment the database hold 3.9 million cells [45] worldwide.
The API offers two types of interfaces depending on what type of format the response should follow. The request should be sent to one of the two following addresses:

- http://cellid.labs.ericsson.net/json/lookup
- http://cellid.labs.ericsson.net/xml/lookup

To create a properly formatted request the following parameters need to be specified:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cellid</td>
<td>The id of the cell</td>
</tr>
<tr>
<td>mnc</td>
<td>The mobile network code</td>
</tr>
<tr>
<td>mcc</td>
<td>The mobile country code</td>
</tr>
<tr>
<td>lac</td>
<td>The location area code</td>
</tr>
<tr>
<td>Key</td>
<td>The API key generated by Ericsson Labs needed for authentication of the requester</td>
</tr>
</tbody>
</table>

Table 11 Parameters included in the mobile location request.

To perform the request these parameters are added to the request URL, an example of this is showed below:

- http://cellid.labs.ericsson.net/xml/lookup?cellid=0007a082&mnc=07&mcc=241&lac=01&8&key=12345

The server that receives the request and performs the lookup will then respond with an XML formatted body.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<position>
  <accuracy>859</accuracy>
  <latitude>35.122155</latitude>
  <longitude>-101.846008</longitude>
  <name>Hollywood Road</name>
</position>
```

The response includes four parameters: accuracy, latitude, longitude and name. The accuracy parameter is the radius in meter of the coverage surface that the base station operates in [46]. The latitude and longitude is the given coordinates for the cell. The name parameter is the given name for the cell. This XML data can be parsed by the server and then stored for the specific case.

The server will provide interfaces for uploading cases from both a mobile client and through a web interface. In order to determine the creation position when uploading a case via the web interface location, a lookup for IP Addresses is needed.

Geobytes Inc offers a service called IP Address Locator tool that for a given IP will respond with a nearby position (longitude/latitude) [47]. The request is a standard HTTP GET request sent to the following address:

The request is attached with some parameters that are then used for running against the lookup database at Geobytes to determine the position.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLocation</td>
<td>Point out that the position of the IP address is of interest</td>
</tr>
<tr>
<td>template</td>
<td>The response could be formatted in different way by specify which template to use. By Using &quot;LonLatCity.txt&quot; the response will include the longitude, latitude and the name of the city in the body of the response.</td>
</tr>
<tr>
<td>IpAddress</td>
<td>The IP Address that should be positioned</td>
</tr>
</tbody>
</table>

Table 12 Geobytes Request.

Below is an example of a request where the parameters are added to the URL:

- http://www.geobytes.com/IpLocator.htm?GetLocation&template=LonLatCity.txt&IpAddress="192.237.142.7"

When choosing “LonLatCity.txt” as a template the respond body will have the following structure:

```html
<body>
  [Longitude], [Latitude], [City]
</body>
```

The longitude, latitude and the name of the city can then be parsed by splitting the body on the comma character.

### 8.2.6 Show creation spot

The server should be able to create and show a map for the provided position data for each case.

For all cases stored at the server there exists location data in forms of longitude and latitude. When a web client user request to look at a certain case a web page called “caseView” is returned. One of the modules embedded on this web page is a map showing the creation spot of this case. This map is realized using Google Maps API [48].

The Google Maps API provides an interface for creation of a map representation. By combining the web application with use of JavaScripts the API offers several tools for manipulate maps.

In this server application the API is used for creating a map that is embedded into the case view page. On this map the longitude and latitude that represent the creation spot of the case is added as a marker on the map.

To be able to use the API the source need to be loaded in the web application. This is achieved by the following code snippet.

```html
<script
  src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=[KEY]"
  type="text/javascript"></script>
```
This will load a JavaScript file containing all functions that is needed for using the API. The KEY parameter is used for authenticate the using application, this key is provided by Google when signing up for usage of the API.

To show the map on the case view page a method specified in a JavaScript file developed for this application is needed. The code below represents the code that is located in that JavaScript file that is invoked from the using web page.

```javascript
function init(arr) {
    if (GBrowserIsCompatible()) {
        var markers = arr;
        var map = new GMap2(document.getElementById("map"));
        for(var i=0; i<markers.length; i+=3) {
            map.setCenter(new GLatLng(markers[i], markers[i+1]), 6);
            map.addControl(new GSmallMapControl());
            map.addControl(new GOverviewMapControl());
            var point = new GLatLng(markers[i], markers[i+1]);
            marker = new GMarker(point);
            map.addOverlay(marker);
        }
    }
}
```

This function will take an array with longitude and latitude of the creation spot as input “arr”. A map will be created using GMap2 function, a style element called “map” will be used for accessing the map in the web page. The position data will be conducted as a marker that will be placed on the map. The map will be positioned with the marker in center. Some built in modules are added to the map, SmallMapControl is a tool that the user can use to zoom and pan on the map, OverViewMapControl adds a small overview map to the map implementation.

This init function is called using the following code from the web page.

```html
<body onload="init(${arr})" onunload="GUnload()">
```

When the parameter “arr” is set from the server application it will call the init function located in the JavaScript file when the web page is loaded. When the page is unloaded the function GUnload is called and the map representation is destroyed.

8.2.7 View Images

The server should provide a user interface where the image could be viewed easily.

Due to the high resolution on images uploaded to the server an image viewer is needed to avoid displaying the whole images at once. Zoomify [49] offers a Flash module as a plug-in on webpage’s that makes it possible to zoom and pan on JPEG images with high resolution.

The product used in this application is the Zoomify Express. Zoomify Express is a freely available edition of Zoomify that offers a converter for converting images into a applicable format and also the flash module needed for displaying the images on a webpage.

Images need to be converted into a format that the Zoomify module can handle. The converter that is provided with the application will convert the image into pieces of tiles displaying different areas of the pictures.
By converting the images into tiles the clients who views the image at the webpage doesn’t need to download the entire image instead only the current view is downloaded.

The Zoomify converter is delivered as a droplet. To use the droplet just drag and drop an image on top of the converter icon, this will create a directory with the same name as the picture stored beside the picture at the same path. The new directory will include the tiles needed for display the image with the Zoomify viewer. This is feasible for manually conversion of pictures but for this server application this needs to be done automatically.

An alternative way to run the droplet is by using the command prompt. The command that is executed in the command prompt will perform the same as the drag and drop method. To make the command prompt method automated this can be translated into a batch script that will perform the same thing.

A batch script [50] consists of one or more commands that are to be executed. The script is an unformatted text file with one command per line. These commands are performed in sequence when the script is executed.

```bash
cd\cd C:\ZoomifyExpress4-Win\"Zoomify Converter.exe" "C:\Pictures\Hasse.jpg"exit
```

This script will first move to the path containing the droplet and then start the droplet with the specified picture. When the converter is done processing the script will simply exit.

To be able to run a batch script from a Java application a process needs to be spawn. This can be achieved by using the Runtime class in java [51]. An instance of the Runtime class exist in every java application, this instance provide methods for interact with the environment it is executed in.

To retrieve the Runtime instance the static method getRuntime() is used. With this instance a process can be spawned by calling the exec(String command) method. This will create a new process and run the specified command within the process.

```java
Runtime r = Runtime.getRuntime();Process p = r.exec("cmd /c start /w C:\myBat.bat");p.waitFor();
```

The command “cmd /c start /w C:\myBat.bat” that is provided to the process in the above code snippet will execute the myBat.bat script in the command prompt. The flag “/c” will ensure that all specified commands will be carried out [52]. The /w flag ensures that programs will be executed sequentially; one program will be executed at the time [52].
To make this process fully automated the batch file that should be executed needs to be generated from the server application. There is only one parameter in the batch script that needs to be dynamically added and that is the path to the file. As a new image arrives to the server it will be stored in an assigned directory. The path to this file will be passed to a function that will generate a batch script that will convert this newly arrived image into a Zoomify image.

8.2.8 Image processing

ImageJ is a java based image processing application with a freely available source code released under the public domain [53]. The program features standard tools that could be expected from a modern image application such as: display, select, edit and save images. The open architecture of ImageJ makes it possible for developers to create specialized plug-ins to extend the performance of ImageJ. With these extensions in form of plug-ins customized image processing problem can be solved e.g. resolution improvement or image recognition. Other features that makes ImageJ useful is the ability to create custom made macros. Macros are script that executes specified commands in a sequence. The commands could either be provided by extended plug-ins or standard tools in ImageJ. To generate macros the built-in command recorder could be used, another way is to write the desired commands in a text-editor.

In the server application different types of processing algorithms could be interesting for example: detection of malaria parasites in erythrocytes, parasite recognition algorithm or parasite count algorithms.

For this application one type of algorithm was prototyped namely, detection of areas with a specific size and light intensity. This could be seen as an early prototype for a processing algorithm that could be used for detecting malaria parasites in a blood sample image. The algorithm is built up using a specially designed macro that is executed together with ImageJ. Below is the body of the macro:

```java
open("uploaded image path");
run("Unsharp Mask...", "radius=7 mask=0.90"); run("8-bit");
setAutoThreshold();
setThreshold(168, 253); run("Convert to Mask");
run("Analyze Particles...", "size=15-50 circularity=0.00-1.00 show=Ellipses display clear include summarize");
saveAs("Jpeg", "save path");
close();
saveAs("Measurements", "result save path");
```

For every image received at the server a macro with the path to the image is generated. This macro is the started with ImageJ using a batch script (in the same was as the generation of zoomable images for the web application).

```java
cd\cd C:\Program Files\ImageJ\java -jar -Xmx200m ij.jar -batch microPic1262849795240_mask
@echo Result created
exit
```
This batch script is executed with the help of Java Process class in the same way as the zoomable images.

8.3 Client Application Implementation

The server application implementation is now described in the previous chapter whereas the implementation of the client application will be described in this chapter.

The mobile client consists of a main midlet, MobileMicroscope, which handle incoming requests from the user interface. Requests are forwarded to a proper task class that runs the task and produces data that is passed to the next view. The application has several different task classes for different purposes and provides data to a specific view which is handled by the main midlet. A full view of the implementation can be viewed in Figure 69.

Figure 69 Class overview of final mobile application implementation
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileMicroscope</td>
<td>The main midlet class that handles input from users and delegates it to an appropriate task class.</td>
</tr>
<tr>
<td>ConnectionInit</td>
<td>Handles the login procedure, composes a request that is sent to the server and reports to the main midlet with an outcome depending on response from the server.</td>
</tr>
<tr>
<td>GPSFinder</td>
<td>Retrieve position data for a specific case.</td>
</tr>
<tr>
<td>PicTaker</td>
<td>Take a picture or chooses an existing picture on the phone and attaches the picture to a specific case. Have responsibility for its own user interface by showing live stream from the camera.</td>
</tr>
<tr>
<td>Sender</td>
<td>Sends a collection of created cases to the server.</td>
</tr>
<tr>
<td>CollectorCase</td>
<td>Collects information about a certain case by sending a request to the server.</td>
</tr>
<tr>
<td>CollectorMessage</td>
<td>Sends a request to the server and wait for a response containing all group messages.</td>
</tr>
<tr>
<td>CollectorPicture</td>
<td>Searches on the phone for pictures and provide references to them.</td>
</tr>
<tr>
<td>CollectorSearch</td>
<td>Sends a search request to the server and waits for a response containing matched cases.</td>
</tr>
<tr>
<td>LWUIT Classes</td>
<td>Each object is responsible for a certain view in the application for instance mainMenuForm is responsible for the interface of the main menu. All invocations made in the views are handled by the main midlet, MobileMicroscope. There are also two modules responsible for dialogs; DialogShow is a module for indicating progress whereas DialogShowButton prompts a message and wait for a button hit.</td>
</tr>
</tbody>
</table>

Table 13  Client application classes
8.4 Implementation of mobile client requirements

This section will describe how the requirements for the mobile application specified in section 6.2 were implemented.

8.4.1 Collect cases

The process of collecting data for a case is done using several task classes. Which task class to invoke and which view to show is regimented by the main midlet MobileMicroscope.

Figure 70 Case creation process

The user will then choose whether to attach an existing image on the phone or taking a new image using the snapshot abilities provided by the application. This is described in section 8.4.4.
When an image has been attached to the case the last step is to submit patient data. The main midlet will display an interface containing a form where the user can fill in data about the patient, this interface is called infoForm. The user will then submit the data which will be validated by the main midlet to check if all required fields was gathered. If the validation succeeds the submitted data will be displayed so the user can check the information one last time and either edit the data or submit the case.

**Ability to work offline**
The process described by Figure 70 is developed with no interaction against a mobile network which implicates that it could be done offline. The collected case will be stored locally on the phone until it’s uploaded; this is described in section 8.4.2.

### 8.4.2 Upload Cases

Created cases will be stored locally on the phone until it’s uploaded. When the user requests to upload the existing cases the Sender task class is invoked.

The main midlet will hold a vector of unsent cases, when the user requests to upload the cases this vector is sent to the Sender task class.

The first validation is the check if there are any cases to send. This is done by checking the size of the vector and proceed if the size exceeds 0.

The second validation is to determine if there is any network connection available and this is done using the System [54] class in Java ME. In the System class the getProperty method is used, this method gives property information for the device it currently runs the application on. Sony Ericsson has developed a property for checking the network status using this class and method. By providing the key "com.sonyericsson.net.status" a status message is given [55].

```java
String net = System.getProperty("com.sonyericsson.net.status");
```

This code snippet above uses the provided method, the string “net” will contain one of the following answers; “Home PLMN”, “Available”, “Preferred”, “Forbidden”, “No Network” and “Unknown”. The first three answers denote that it exist a network connection and the other three denotes that it isn’t any connection available.

The last validation checks if the user has been authenticated with the server. This is done by invoke the instance of ConnectionInit which is created on startup and check whether it has processed a login request or not.

For each case in the given vector a send thread is created, this makes it possible to handle and then send the cases concurrently. The thread starts with generating a HTTP multipart message body containing all parameters in the case including the images file. This message is stored as a StringBuffer in the application until it is sent. The request type “add case” is added to the headers of the HTTP message together with the username and password for the uploading user.

When the Multipart message is generated a connection is created to the server. This is made by using the HttpConnection [56] class in Java ME. A connection is established to the server by using this class. To send the message an OutputStream is provided by the connection, through this stream the message is sent byte by byte.
When the message is sent an InputStream is opened against the same connection. The Sender class then waits for a response from the server telling if the case was accepted or rejected. The thread then either removes the case from the local storage or keeps it depending on the outcome of the response.

### 8.4.3 Authenticate users

The authentication process has been implemented into a task class named ConnectionInit. When the user starts the application, the first view that appears is the login view. The user then decides whether to work offline or authenticate with the server. If the user chooses to log in to the system this task class is invoked.

![Flow chart describing authentication process](image)

The username and password are fetched by the main midlet (MobileMicroscope) which creates an instance of the ConnectionInit class and pass these parameters further to the instance. The ConnectionInit class will then create a HTTP Post request including username and password in the header. This request is then sent over to the server application that verifies the data against the account database. If the username and password are valid an acknowledgement is sent to the client otherwise a decline message is sent.

The connection to the server is created using the HttpConnection class. The requestType “login” along with the username and password is sent in the headers leaving the body empty.

The HTTP message is sent using an OutputStream and the response is received using an InputStream from the connection.
The instance of the ConnectionInit class will be kept alive during the whole time while the application is running, when the application needs to check whether the user is authenticated this instance is invoked.

If the user decides to use the application in offline mode no instance of ConnectionInit will be created which implicates that the user isn’t authenticated.

### 8.4.4 Image handling

The application should provide two functions to attach images; attach local phone images and attach images taken by a built in camera function.

Java ME provides two types of tools for image handling namely MMAPI [57] (Mobile Media API) and AMMS [58] (Advance Mobile Media Supplements).

MMAPI is the basic API for handling multimedia tools in midlet applications. This API provides several features concerning audio, video and camera abilities.

The AMMS API extends the MMAPI with more features and tools, for example controlling the created camera instance and image processing tools.

In Figure 73 the process of attaching an image using the camera function in the application is described. The process starts with initializing some parameters and then the camera view is showed. In MMAPI there exists a class named “Player”, this class is used to get access to the built in camera.

```
Manager.createPlayer("capture://video");
```

The command line above creates a instance of the Player, by giving the key “capture://video” the camera will be accessed.

To be able to acquire snapshots the Player needs to be controlled, this is done using Control interfaces. In this application two interfaces is used, SnapShotControl which is used for setting properties of the image to be taken and control the snapshot process, CameraControl which is used for setting the resolution of the image.

The Player object is started and then placed on a LWUIT form so that live video stream from the camera can be viewed by the user. LWUIT has a certain component named MediaComponent that is used for displaying the video stream.

The application will wait for the user to hit the snapshot button that will start a separated thread which will acquire and process the image. The acquirement of the image will start by first setting the name and path where the image should be stored by using the SnapShotControl.

```
snapControl.setFilePrefix(picturePrefix);
snapControl.setFileSuffix(pictureSuffix);
snapControl.setDirectory(pictureDir);
```

The resolution of the image is set to the highest available on the current phone by using the CameraControl interface.

```
int[] resolutions = cameraControl.getSupportedStillResolutions();
int maxValue = (resolutions.length / 2) - 1;
cameraControl.setStillResolution(maxValue);
```
After this the actual process of taking the image is started.

```java
snapControl.start(1);
```

When the image is stored on the phone an event will arise and notify this. Then the application will create a thumbnail to display the image. A thumbnail is created because the real image size is ~2.5 MB which is too large for the show because of small application memory size.

The thumbnail is created by using the MediaProcessor class and the ImageTransferControl interface, both provided by the AMMS API. A MediaProcessor instance is created and the ImageTransferControl interface is connected to the MediaProcessor instance. An InputStream is opened for the original image and then given to the MediaProcessor. To create a thumbnail the new parameters for the smaller image is given using the ImageTransferControl. The thumbnail can then be read into the application from an OutputStream connected to the MediaProcessor. This image is sent to the imageView form where the user will decide whether to attach it to the case or take a new image.

![Flow chart](image.png)
The user also has the ability to choose an image stored locally on phone. The process of choosing an already existing picture is shown in Figure 74. This request starts with creating an empty result vector which will contain references to the found pictures at the end. CollectorPicture is a threaded class that will take a directory as input, in this directory it will go through all entries and add all found images to the shared result vector created earlier. The process will start with sending all top directories on the phone to a CollectorPicture thread. If the CollectorPicture thread found a subdirectory a new CollectorPicture thread will be created for that directory. This process will go on until all CollectorPicture threads are finished.

```java
FileConnection fc = (FileConnection) Connector.open(path);
Enumeration filelist = fc.list();
```

The code snippet above shows how all entries for a directory could be listed, using FileConnection class provided by Java ME.

When all CollectorPicture threads are finished the result vector will include references to all images located on the phone. This vector will be presented for the user. The user will then decide which picture to be attached to the case, the only thing that is displayed is the name of the picture due to the time it would take to create thumbnails for every picture. When the user chooses a picture a thumbnail will be created and shown, this creation of a thumbnail follow the same procedure as for a picture taken by the built in camera function.

### 8.4.5 Retrieve position data

The application need to be able to attach GPS data to uploading cases.

---

![Flow chart describing retrieving GPS data process](image-url)

Figure 75 Flow chart describing retrieving GPS data process
When a new case is created the GPSFinder class is invoked first. This task class will try to get a GPS fix using the built-in GPS receiver in the phone. The process of retrieving a GPS position is described in Figure 75.

Java ME offers an API for this kind of requests namely the Java ME Location API [59]. The main class used in this API is the LocationProvider class, this class will for a given criteria, specified by the Criteria class, retrieve a position using the built in GPS.

The process starts with setting up a LocationProvider instance. The criteria are specified with a preferred response time set to 10 seconds and an accuracy of 500 meter. This specification is given to the LocationProvider.

```
Criteria cr = new Criteria();
    cr.setPreferredResponseTime(10000);
    cr.setHorizontalAccuracy(500);
    LocationProvider lp = LocationProvider.getInstance(cr);
```

The LocationProvider will then try to get a position using the getLocation() method. This method will take a timeout time as input, if no GPS data is received within this timeout limit the process will stop. The timeout limit is set to 30 seconds.

```
Location l = lp.getLocation(30);
Coordinates c = l.getQualifiedCoordinates();
```

If a location were found the process will stop, otherwise it will try to retrieve the Cell-ID of the current cell that the phone is connected to. This is achieved by using the System class, and the special Sony Ericsson key “com.sonyericsson.net” from which the Cell-ID can be found. The application will retrieve all data needed for determine the position using Ericsson Labs Mobile Location API, described in section 8.2.5. This method only works on Sony Ericsson devices.

```
System.getProperty("com.sonyericsson.net.cellid");
System.getProperty("com.sonyericsson.net.lac");
System.getProperty("com.sonyericsson.net.cmcc");
System.getProperty("com.sonyericsson.net.cmnc");
```

The position data will then be fetched at the end of the case creation process by the main midlet.
8.4.6 Persistent Storage

The case data need to be stored locally on the phone until it’s uploaded in case of no network connectivity.

Figure 76 Storage process of case data

When a case has been created and acknowledged by the user it is stored at a local storage on the phone, this process is described in Figure 76.

Java ME offers persistent storage by the RecordStore class. The RecordStore class offers an interface from which the application can store data locally on the phone, the storage on the phone is handled by the platform. The application will create its own storage, this storage place is private and no other application will be able to access it.

```java
RecordStore rs = RecordStore.openRecordStore("MomCases", true);
byte[] caseBytes = caseToStore.getStoreBytes();
int id = rs.addRecord(caseBytes, 0, caseBytes.length);
caseToStore.setRecordStoreId(id);
rs.closeRecordStore();
```

The code above shows how the RecordStore class is used in the application. The case needs to be transformed into a byte sequence which can be parsed when the application need to fetch data from the storage.

When the application is started the first thing it does is to check if there is any unset cases available in the storage. This process is described in Figure 77.

```java
RecordStore rs = RecordStore.openRecordStore("MomCases", true);
RecordEnumeration recEnum = rs.enumerateRecords(null, null, false);
```
The application will fetch all entries in the private storage; this is achieved by the code snippet above. Each entry will then be parsed into a case and handled by the running application until it is sent to the server.

8.4.7 Search and message request

Messages and activity created in connection to the cases needs to be fetched by the mobile application.

The process of fetching the messages stored is described in Figure 78. This process will first check network connectivity and user authentication. If this validation passes, an empty inboxForm is created which will display messages to the user. A HTTP request is generated and sent to the server using the HttpConnection class. The HTTP request will specify in the headers the requestType (Get Messages) and the user for which the messages should be gathered for.
The application waits for a certain amount of time for a response from the server. When a response arrives it will be parsed so it could be handled and displayed by the inboxForm.

**Ability to search after uploaded data**

The request above states that the application needs to be able to search for cases from the application.

The implemented search function is described in Figure 79. This process will first gather search parameters from the user by displaying a searchForm. The user will then submit parameters that will be sent to the server using the HttpConnection class. The parameters will be located in the header of the request.

The application waits for a certain amount of time until a response arrives. The response will be parsed into a result list which will be displayed for the user. The user then has the ability to request a certain case in the list to get more information about that case.

### 8.5 User Interfaces

The web interface was implemented using standard HTML and CSS together with JavaScript.

The mobile client user interface was implemented using LWUIT. LWUIT is more extensive than the standard user interface API offered by Java ME and makes is easy to create nice and intuitively user interfaces.

Both interfaces can be viewed in Appendix E – User interfaces.
Chapter nine

9 Discussion

The project was divided into two parts with strong connections and dependencies. This chapter discusses the results and alternative solutions that can lead to future works. A through going conclusion is that the components and specifications depend on the application. Different diseases demand specific software, such as image analysis algorithms and sometimes specific hardware. This project proves the main concept instead of targeting specific diseases.

9.1 Hardware discussion

This thesis has showed that on chip microscopy is possible to implement as a complement and sometimes as a substitution to conventional optical microscopy. It is also clear that many improvements are possible and that the development of a lens less on chip microscope is far from finished.

There are numerous possibilities with a mobile microscope that can digitalize specimens. Automatic cell counting and diagnosis makes it possible for more people to get a correct diagnose. It also lowers the demands for education of laboratory technicians making it faster to be implemented and used in the field. The ultra large viewing field of 24.85 mm² makes it possible to take only one picture instead of moving the sample which is being made today.

The project had as main requirement that the prototype should use conventional techniques of mounting the specimens. The prototype can therefore handle conventional standard microscope slides and cover slips.

The resolution of 12.4 µm is sufficient for diagnosis of various diseases. The highest goal was to spot malaria and tuberculosis, but it seems that this initial goal was set too high. Malaria parasites are around 1 µm in size and tuberculosis exhibit the same thickness but can vary in length.

The aim to determine the minimal hardware requirements slightly shifted during the project. The minimal hardware requirements are very connected to what resolution that’s needed to see different specimens, and not a question that provides an answer which is valid in all situations. This project aim instead became what’s needed to create the best resolution possible with cost efficient and commercial means.

9.1.1 IMX043 sensor

The specifications of the used IMX043 sensor could fulfill the requirements set up for this thesis project. It provided a price worthy alternative and was commercially available. However, there are studies, even if they are scarce, where a custom sensor has been built to improve the resolution in different ways.
The most important factor affecting the obtained resolution is the pixel size of the sensor. Substituting the sensor with a custom built sensor with pixels smaller than 1.75 µm would in this case increase the resolution. In a recent project [62] a scanning technique to enhance the resolution has been used. A matrix of holes smaller than the actual pixel is put on the sensor surface. The specimen is then moved over the matrix, relatively tilted so that the smaller holes instead determine the resolution rather than the individual pixel size. More about this can be read in the mentioned article. Another promising project [63] uses the unique diffraction pattern created by objects and then identifies these with help of a computer.

The Bayer filter applied to the IMX043 creates some opportunities but it also directly decreases the resolution. If a sensor would be chosen that lack the Bayer filter, the resolution would double if monochromatic light is used.

### 9.1.2 Light source

A perfectly collimated light source would greatly improve the resolution of the system but. The designing such system would require more time, knowledge and equipment than could be included in this project. This part of the project could easily have been a separate thesis of its own. The light source therefore became the component where the most compromises had to be done.

It has been shown that the light source plays a vital role creating the image but it also affects the design of the system. A collimated light source would make it possible to minimize the length of the light tube.

The choice of light source also depends on the clinical application. Some applications only demand a difference in color from the scattered light instead of high resolution. An example of such an application is mentioned briefly in section 4.7.1 where a sample of ovarian cancer is showed.

### 9.1.3 Immersion oil and mounting medium

In the appendix the importance of mounting and immersion oil is discussed. It’s probable that different mounting and immersion mediums can change the light path down towards the sensor. In theory the blurriness of the picture depend on the refractive index of the mediums. It seems like a mounting medium with a refractive index similar to air would create the sharpest image. This theory hasn’t been evaluated in this project and should therefore be evaluated in future studies.

### 9.1.4 Hardware limitations

The hardware limitations are:

- **Resolution**
  The acquired resolution of 12.41 µm limits the possibilities to diagnose diseases like malaria and tuberculosis that requires higher resolution (~1µm).

- **Sensor chip circuit geometry**
  The sensor circuit configuration prohibits that the microscope slide can be photographed at some places. There are obstructing resistors that are higher than the sensor and protective glass. If these components could be moved, the whole microscope slide could be examined.
• **Light source**
  Compromises has been made when designing the light source. These compromises resulted in a fairly clumsy prototype with its long light tube. If the light source would be substituted with a collimated source the resolution would increase and the tube could be much shorter.

• **Battery**
  The battery time while using the camera and internet connection can be quite short. It is therefore recommended to include extra batteries and a solar charger.

### 9.1.5 Hardware, further work

Possible improvements for the hardware part are summarized below:

- Change the camera sensor chip to obtain higher resolution
- Replace light source to a collimated one
- Evaluate mounting and immersion oil techniques
- Find new techniques to extract and mount specimens
- Evaluate the concept in a clinical trial

### 9.1.6 Sony Ericsson TakeBack program

In order to ensure low cost of a final product including the modified mobile phone the take back programs of the handset vendors should be considered.

At Sony Ericsson a “TakeBack” program [60, 61] takes care of used mobile phones and recycles them. The recycling program includes re-use of parts of the mobile phones. These parts could later be found in other products like toys and dishwashers. Using such program to ensure low-cost components should be further considered.

![Figure 80](image_url) Mobile phone re-used in new products, for example dishwasher displays.

### 9.2 Software discussion

#### 9.2.1 Server design and implementation

As the server implementation increased, the management of the code became difficult, especially the main servlet that include all the application logic. This is one of the main problems when using three-tier architecture with a single servlet, also known as Model 2 architecture. All incoming requests are handled by the servlet and the responses are directed to the correct view through the servlet, this makes the code quite large and also repetitive.
A possibility to improve further development and make the code more manageable is to use a framework. When developing a Java EE application two major frameworks exists, Spring and Struts [64, 65], both are commonly used for bigger enterprise applications. The frameworks extends the Java EE web API to make it easier to separate logic and remove unnecessary code that otherwise needs to be added when normal Servlets are used.

Instead of using a Model 2 approach of the three tier architecture both Spring and Struts encourage to use the classic design concept of the MVC (Model-View-Controller) pattern. The MVC pattern focuses on trying to get a strong connection between a specified model and the appropriate view leaving the controller to only receive incoming requests and pass that into an appropriate model. The model will then have a responsibility to inform the correct view instead of passing the information back to the controller.

This differs from the implementation in this report where the controller (Servlet) receives incoming requests, processes the requests using the JDBC help classes and the internal help methods, this is then packed into a response which is sent to a selected view that parses the responses and generates a user interface. The purposed pattern of the MVC will remove the interaction between the controller and the view by leaving it up to the model to inform the view when data needs to be updated. This will decrease this size of the servlet code but instead increase the model code as more logic is needed to be processed.

Another aspect that both Spring and Struts encourages is the usage of a dependency injection (DI). When a server application has developed dependencies between modules the DI is created. Usually these dependencies are managed by the programmer but by using a framework this sort of management could be operated by external parties. By specifying the dependencies in a manageable file a container will make sure that these modules are wired together in the code; this is also called inversion of control (IoC). The IoC container will inject the dependencies in different modules when necessary, this is called dependency injection and the programmer wont have to worry about it when implementing the application.

By using the Spring or Struts framework the MVC pattern could be realized and a DI could be used fairly easy.

Another question to answer is: when is it necessary to use a framework like Spring or Struts? The answer is that it depends of the size of the application, but for this project it could have been used for sure. It is often used when there are more people involved with the project when it is easy to split up the work among the people. The framework has built in support for management of implemented server modules.

As the software, in this project, was implemented by one person the selected design is feasible. The implementer had a full view over implemented parts and how they interact with each other. If the project had been bigger in the beginning the selection of architecture would have been different. Also more time had been spent on deciding the architecture and diving work tasks among the people involved with the project, this would certainly have included a framework like Spring or Struts.
9.2.2 Client design and implementation

The mobile application was implemented with general Java ME classes, using a pattern of task classes. The correct task class is initiated from the controlling midlet which also updates the user interface. The implemented application initiates a lot of coupling between different classes, one way of improve this is to use a specially designed framework for Java ME, FallME [66]. The FallME framework introduces an IoC container, described earlier for the Spring framework. The FallME framework will help inject specific displaying units into task classes which reduces coupling among classes and the management of the application modules is made easier because of the external management part, the IoC container.

9.2.3 Data collection

There exist many applications that offer collection of data such as: JavaRosa, EpiSurveyor and OpenMRS [67, 68, 69]. These applications offer a simple solution to set up a system to gather data which is highly used for mHealth purposes in developing countries for example creating health questionnaires or gather birth data.

These applications could be integrated with the system created in this project. One thing that isn’t implemented in this application is a way to create health questionnaires on the server application using a web interface; these questionnaires should then be fetched by the mobile application and submitted together with the patient data. EpiSurveyor offers this kind of feature where a questionnaire form is created on a server through a web interface and later on is downloaded by a mobile client. The mobile client will then submit the form containing filled in data which later could be watched using the web interface.

The integration between this application and EpiSurveyor would include both server applications as they are. The implemented server application, in this project, will handle all patient data and medical images. EpiSurveyor will be used for creating health questionnaires and watch statistics over gathered data. Through EpiSurveyor an XForm representation of the questionnaire will be created, which will be fetched by the mobile application by using the API offered by EpiSurveyor. The mobile application in this project needs to be extended with the ability to parse the received health questionnaires and present a valid user interface. The submission of data from the new mobile application will be split into two parts where the medical data will be sent to the original server application and a health questionnaire to the EpiSurveyor server application. These two data representations need to be linked together by using the case ID from the original server application. Links to the EpiSurveyor web interface could be provided by the web interface in the original server application so that the web user only has to log in once.

9.2.4 Security

Focus has not been put on the security aspects of the systems and the protocol used between the server and the mobile application can be improved from a security perspective. The header in the HTTP packet that is sent between the mobile application and the server application always include the username and password. This is a security aspect that needs to be solved if this system will be used with real patient data. One quick solution is to put these parameters into the body and encrypt the data. The packets should then be sent using secure connections (HTTPS) to ensure privacy.
9.2.5 Software limitations

The software limitations are:

- **Security**
  The interaction between the server and the mobile client uses connections where the data is sent unencrypted. If this application will be used with real patients the data needs to be sent encrypted using secure connection to ensure privacy.

- **Image processing**
  The image processing algorithm provided by the server application is to show that such a concept works. It needs to be custom made for a specific disease.

- **Web interface**
  The web interface renders badly in Internet Explorer (IE) due to the bad support of CSS in IE.

9.2.6 Software, further work

Possible further work described below:

- Port the server application to a framework like Struts or Spring,

- Implement support for data collection applications like EpiSurveyor or MST. These applications could be used for collecting health questionnaires.

- Improve and add image processing algorithms.

- Use secure connection between the mobile client and the server

- Improve the protocol; remove critical data such as password and username from headers and put them encrypted in message body.
Chapter ten

10 Conclusion

The main aims of the project have been met and the concept has been successfully implemented combining the hardware and software parts of the project. A working mobile application runs on the final on chip microscope prototype communicating with a centralized server. Patient information is collected by the mobile application and image analysis capabilities has been implemented on the server side.

10.1 Hardware

A modern mobile phone with a good camera was modified to meet the requirements of the project. It can send images using the mobile 2G/3G network and still use commercially available components.

Factors that affect the resolution has been evaluated and the hardware and theory behind it have been discussed. The diagnosis that is interesting rules the hardware decisions.

A prototype was built using a Sony Ericsson C905 as hardware platform. The phone was chosen for its state of art camera module and image sensor. Experimental evaluations concluded that a feasible modification for this project was to remove the lens package and cover glass from the images sensor and replace it with a thin protective glass. These modifications were done at the Electrum clean room laboratory with good results. Initial attempts indicated that the modifications needed to be done in a clean room to avoid dust between the protective glass and sensor. The housing of the prototype is designed as part of the thesis project and custom built at the prototype lab at Ericsson. The components for the housing are made of plastic and aluminum and the components became relative expensive (~7.000SEK). This cost could be lowered if the housing becomes mass produced. All other components are standard components commercially available.

The resolution acquired by the handset prototype was 12.4 µm. The theoretical boundary was calculated to be 10.5 µm for the standard CMOS sensor. The main reason why this boundary wasn’t reached is that the object was placed 0.3 mm above the sensor surface and not directly on the surface, which would improve the resolution. A green LED was used together with a lens system to improve light propagation. Green color was used due to the applied Bayer filter on the sensor surface, demanding green for best resolution. To further enhance the resolution a collimated light source should be used.

Both the prototype and the experimental set up were extensively tested at the Swedish Institute of Infectious Disease Control and MTC. Markers for various diseases were confirmed visible with use of the LOSMic. The prototype is tested to work with the WHO recommended technique Kato-Katz to extrude eggs from the four STH diseases mentioned with positive results. S. haematobium which causes urinary schistosomiasis infection has been observed with standardized techniques. Future clinical studies are needed to determine if these findings cohere with conventional techniques used in the field.
10.2 Software

In the pre-study phase of this project the requirements for the system was defined with inspiration of similar existing system and what was found reasonable to adopt.

The proposed software solution includes a mobile client application and a server application. The most crucial data needed for diagnose is the microscope images which demands an application which can acquire images.

Attached along with the images is information about the patient such as; name, age, village, gender etc. is gathered. Another requirement is to gather GPS data, by adding GPS data to all uploaded cases; it will be possible to track down the creation spot which could determine geographical outburst patterns of different diseases.

The server application will receive incoming data from the mobile client. This data will be presented through a web interface where diagnosis can be made and reported back to the mobile client. The server will also offer an image processing algorithm which could, with further work, make it possible to diagnose different diseases automatically.

The mobile client application was implemented using Java ME as platform.

Because of the selection of Sony Ericsson C905 as hardware platform Java ME was a natural choose as implementing platform due to the good support on that device. The design of the application was conducted to be as separated as possible by implement specific task class for the different requirements and a controlling class that connects these task classes with the user interface.

The server was implemented using Java EE as platform. The selection of Java EE depended on the fact that it’s widely used a platform and it is possible to implement a server solution following the commonly adopted three-tier architecture. This resulted in a main Servlet that handles all logic, database help classes to handle the data model and Java Server Pages for presenting a web interface.

The mobile application was tested on two different devices, Sony Ericsson C905 and Sony Ericsson C702. It was proven that the application works and that case data can be uploaded to the server using a mobile network.

The server application was tested on an average personal computer (Intel P4 2.66 GHz, 1 GB RAM). Windows XP was used as operating system and GlassFish as application server, hosting the implemented application.

The server is able to interact against both the mobile client and the web client.
Chapter eleven

11 References


[17] Dr. Rainer Wegerhoff, Dr. Olaf Weidlich, and Dr. Manfred Kässens. Basics of light microscopy & imaging. GIT VERLAG GmbH & Co. KG.


[28] Pyser-SGI, Fircroft Way, Edenbridge, Kent, United Kingdom, TN8 6HA. OPTICAL RESOLUTION CHARTS.


[30] The Electrum Laboratory is operated by KTH (Royal Institute of Technology) in collaboration with Acreo the leading Swedish research institute in microelectronics and optics. Electrum laboratory.,


Appendix A – Targeted parasitical diseases

The prevalence of parasitical diseases is vast. It is estimated that billions of people get an infection every year and the number of unknown cases are enormous. Most of the infected people live in rural areas in Africa, Asia and South America [4].

The immensity of these diseases is fundamental to understand the magnitude of the type of concept proposed in this project. This section briefly explains those diseases that where topical in this project. The selection depended on accessibility of samples, parasitical geometry and size and how widely the diseases are spread.

The technique of diagnosing the diseases in this section is all about quantifying the parasites. This is made today by hand and this forces the operator into non ergonomic working positions which can result in strains in muscles, causing injuries. The time spent on a single microscope slide is also relative long.

Bilharzia

Also known as schistosomiasis has an estimated prevalence of at least 200 million people globally. The infection is mostly spread out in rural areas in tropical and sub-tropical areas [3].

A study in sub-Saharan Africa alone, revealed that there where 166 million people that where infected with schistosomiasis, out of these around 200 000 people died every year. Schistosomiasis has a fairly low mortality but the people infected suffer from chronic morbidity. The sickness can be treated with medicine but it’s hard to eradicate all eggs when they can hide within tissue. People get infected with schistosomiasis when they are drinking, swimming, fishing and other activities in and with infested water [4].

There are five species of schistosomiasis that are spread out differently throughout the world. Schistosoma haematobium causes urinary schistosomiasis and the other S. mansoni, S. japonicum, S. intercalatum and S. mekongi causes intestinal schistosomiasis [3].

Schistosomiasis is diagnosed by observing eggs in tissue, stool samples or in urine. The eggs measure between 100-180 µm wide and 40-70 µm in height (Figure 81).

The adult schistosome worm is viewed in Figure 82.
Human Whipworm

The human whipworm or Trichuris trichuria infects about 1050 million people worldwide and about 10 000 of these die every year [4].

The infection is strongly connected with poverty and places with deficient sanitation. It spreads from human to human via infested soil. The lack of latrines continues to spread the disease into soil and water supplies. After the soil has been infested, the eggs of T. trichuria need to mature for about three weeks before it can infect the next person. This happens by vegetables that has not been properly washed, peeled or cooked, drinking infested water and when kids are playing around in the dirt and put there fingers in their mouth [13].

When diagnosing T. trichuria, eggs are searched for in stool samples and quantified with for example the Kato-Katz technique. The eggs look like a football in shape and measure about 70 µm long and 40 µm wide (Figure 83) [9].

The worms which represent the adult form of the T. trichuria measures 30 – 45 millimeters long. They appear in the intestines, cecum and the appendix of the human [9].

Roundworm

The roundworm or Ascaris lumbricoides, has the largest prevalence among the soil-transmitted helminths (STH). It is estimated that 1450 million people get infected and among these 60 000 die every year [4].

The way the infection spreads are the same as the one for the whipworm, described in the section above. Also in this case, evidence of infection is given by searching for eggs in stool samples and then quantified them so that a relation of eggs per gram faeces can be established.

The eggs are elliptic in shape when fertilized and almost tube shaped when unfertilized (Figure Figure 84).
The adult worm measures astonishing 15 – 35 centimetres and are found in the intestines (Figure 85) [7].

**Hookworm**

The prevalence globally is estimated to 1300 million infected people. 65 000 people die every year [4].

There are a number of species of hookworm, where two infects humans. Those two are the *Ancylostoma duodenale* and *Necator americanus*. Both infections can affect the learning and working capabilities. Conditions like diarrhea, abdominal pain, malaise and weakness can appear [13].
Appendix B – Basic handling of specimens

To have understanding of some of the factors that need to be though of when detecting parasitical diseases we have to look briefly into some techniques used to mount and extract the specimens on microscope slides.

General precautions for all parasitological tests are to keep the microscope slides and cover glasses clean. The microscope used must be kept clean when it easily can be contaminated when wet samples are examined. Lab personnel must use means to protect them self from highly contagious samples and wash hands before and after all handling [70].

The techniques mentioned below are selected by there status of recommendation by the World health organization and competing techniques are only referred to.

Kato-Katz

The technique is used to distinguish eggs of different parasitical diseases in stool samples. Some of these are the Soil Transmitted Helminths (STH), especially hookworm, human whipworm and the roundworm. The technique also works for intestinal Schistosoma eggs which disease is known as bilharzia or schistosomiasis [12].

The samples are prepared by taking a bit of faeces and then press the substance through a filter to remove large particles. A template with a hole is then mounted on top of microscope slide. The volume within the hole is known and work as a means of knowing the amount of faeces investigated. The filtered faeces are then put into the hole so it’s completely filled. A cover glass or scrape is used to sweep over the template so the faeces only fill the hole.

A pre soaked plastic cellophane with glycerol and malachite green solution, is used as a cover slip. The template is then removed leaving the known amount of faeces on the slide. The cellophane is then placed on the faeces. The slide is then turned so that the faeces are facing down, towards the table. The slide is then pressed so that the sample spread out [31].

This slide is then examined under a microscope counting the eggs and then use mathematical formulas to determine the amount of eggs in the faeces.

This technique has been questioned to lack sensitivity, alter the morphology of the cells and be unsuitable for fluid and hard specimens. The safety and hygiene for the health worker involved with the technique are also questioned [70]. Even with this knowledge the World health organization still recommends this technique for use in the field when its cost efficient, easy to learn and give a standardized reading [12].

There are other tests that could compete with the Kato-Kats technique and they could be the Sodium hydroxide digest technique, the Stoll helminth egg counting technique and the formol detergent gravity technique [70].

Figure 86 Schistosome egg extracted with the Kato-Katz technique.
Examination of urine for schistosomiasis

One of the species of schistosomiasis, S. haematobium causes urinary schistosomiasis. This type can be observed in urine where eggs are present and sometimes also blood cells.

A known amount of urine is collected in a container, usually around 10 – 15 ml. A solution of example saponin is added to cause disintegration of red blood cells so that the eggs could be observed more easily. The solution is then centrifugalized to separate the sediment containing eggs from unnecessary liquid. This process could be replaced by letting the solution rest about one hour. The liquid lying above the sediment is dispersed and the sediment collected and transferred onto a microscope slide.

It is important to be aware of that the distribution of eggs into the urine is not constant and varies over time. Therefore should the health worker test the patient in different days if an infection is suspected and the first test was negative. The excretion of eggs can also vary over a single day. The best time to find the eggs in urine is between 10.00 and 14.00.

This slide is then investigated under a microscope where the detected eggs are counted. The degree of infection is determined by the amount of eggs per ml urine [70].

Alternate methods of extracting eggs exist and one of them is to filter the sediment with a dedicated filter made for schistosomiasis [12].
Appendix C – Mounting medium and immersion oil

Mounting medium

When the specimen is mounted on the microscope slide the mounting medium can alter the amount of light hitting the sensor. Different mediums have different refractive indexes and if the chosen one differs from the refractive index of the cover glass, the light angle will change towards the sensor.

If the refractive index of the mounting medium is less than within the cover glass the angle of refraction will be less than the angle of incident [71].

![Figure 87 Mounted specimen with a mounting medium with a refractive index $n_1$ less than $n_2$.](image)

![Figure 88 Mounted specimen with a mounting medium with a refractive index $n_3$ equal to $n_2$.](image)

Figure 87 and Figure 88 are examples how the angle is changed due to difference in refractive index. It is obvious that if $n_2$ is larger than $n_1$ the light will scatter in larger angles beyond the cover glass than if $n_1$ is less than $n_2$. This implicates that less light is registered by the sensor and the picture becomes more faded [71].

Immersion oil

When the light travels from the cover glass to the space between the cover glass and the sensor, on more angle shift occurs. The angle of incident will be less if $n_2 > n_1$. Section 4.6.2 discussed earlier that if the light angle towards the sensor is larger than 10° the registered intensity will drop fast.
This means that the picture will have a better resolution if immersion oil is used given that the oil has a suitable reflection index (Figure 89 and Figure 90). The distance between the cover glass and the sensor is in this case very important when a smaller distance generates relative less scatter light wave [71].

To illustrate how the light is scattered after travelling through the cover glass and into the air which fills the void between the cover glass and the sensor, a table has been derived from Snell’s law (Equation 7) [14].

\[
\text{Equation 7 } \quad n_1 \sin \theta_1 = n_2 \sin \theta_2
\]

Where \( n_1 \) is the refractive index of in which the incident light travels in, and \( n_2 \) the second medium. \( n_1 \) is assumed to be air and have the refractive index around 1, and \( n_2 \), assumed glass and a refractive index of 1.5.

<table>
<thead>
<tr>
<th>Incident light angle (in cover glass)</th>
<th>Refractive light angle (air)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.65°</td>
<td>10°</td>
</tr>
<tr>
<td>16.36°</td>
<td>25°</td>
</tr>
<tr>
<td>28.1255° (exceeds critical angle)</td>
<td>45°</td>
</tr>
</tbody>
</table>

Table 14 Difference in angles when light travels from the cover glass into air.
Snell’s law describes what happens to light when passing different mediums. Figure 91 shows what happens to a point source which sends out spherical wave patterns. Here the upper medium exhibits a lower reflective index than the medium below.

Figure 91  Snell’s law describing a point source sending out a spherical wave pattern.
Appendix D – The prototype

Figure 92   Side view of the LOSMic

Figure 93   Top view of the LOSMic

Figure 94   Night picture of the LOSMic with the green LED turned on.
Figure 95  Protective briefcase for the LOSMic including tripod, charger and data cable.
Appendix E – User interfaces

Web user interface

Pictures of the web interface:

![Login welcome screen](image1)

Figure 96  Login welcome screen.

![Account view](image2)

Figure 97  Account view for the health worker after login screen.
Figure 98  Case view for an uploaded patient case.

Figure 99  Add case view.
Mobile user interface

Figure 100  Login screen.

Figure 101  Main menu.

Figure 102  Choose image mode.

Figure 103  Patient data form.

Figure 104  View submitted patient data.

Figure 105  Outbox with one case.
Figure 106 Empty inbox.

Figure 107 Search case form.
List of Figures

Figure 1 Pictures from a clinic in the Ruhira MVP cluster in Uganda (A) and Tiby MVP cluster in Mali (B, C, D, E, F, G)................................................................. 11
Figure 2 Schematic view of an optical microscope.................................................. 13
Figure 3 On chip microscopy. The shadow dissipation on the sensor surface ................ 13
Figure 4 System topology.................................................................................... 14
Figure 5 The conceptual process overview............................................................ 15
Figure 6 Short light wavelength relative to the aperture size.................................. 21
Figure 7 Small aperture and relative large light wavelength................................. 21
Figure 8 Aperture replaced by an opaque object................................................... 22
Figure 9 Small opaque object............................................................................. 22
Figure 10 A wave front composed by point sources............................................. 22
Figure 11 Multiple opaque objects on a microscope slide sending out spherical waves after that light has hit them................................................................. 23
Figure 12 The IMX043 Bayer pattern configuration seen in 100x magnification. Colours enhanced in the upper left corner.................................................... 23
Figure 13 Standard red, blue and green (RBG) Bayer pattern............................... 24
Figure 14 Three new RGB Kodak patterns [16].................................................... 24
Figure 15 Left: Objects covering sensor matrix. Right: Obtained picture by sensor. .......................................................... 24
Figure 16 The sampling of two circles close to each other..................................... 24
Figure 17 Sensitivity graph. The intensity is inverted since shadow imaging is discussed. 25
Figure 19 Method of measuring light beam incident angle [20]................................. 27
Figure 20 Pixel Incident angle characteristics. The relative output at 0° is converted to 1.0 [20] .. 27
Figure 21 CMOS chip from above showing the protective glass on a ceramic frame.......................................................... 27
Figure 22 Green, red and blue LED with internal structure. .................................... 30
Figure 23 Pictures of a piece of hair showing the difference between a green and blue LED. Not enhanced by computer............................................................. 31
Figure 24 3D surface plot of Figure 23. Light intensity represents height but now converted when shadow imaging is discussed......................................................... 31
Figure 25 Coloured tissue sample for cancer diagnostics. A: Green light, B: White light...... 31
Figure 26 Difference between an objective glass mounted with the pattern up away from the sensor (A) and with the pattern placed with the pattern upside down towards the sensor (B)......... 32
Figure 27 The magnification of an object when raising the object above the sensor........ 33
Figure 28 Decrease of distance between object and light source with a round 525 nm green LED with 3.4 V applied ............................................................... 34
Figure 29 Image of the final prototype mounted on a camera tripod............................ 35
Figure 30 Final model of the prototype............................................................... 35
Figure 31 Top view of the final prototype............................................................. 35
Figure 32 Side view of the camera chip housing component.................................... 36
Figure 33 Top view of the camera chip housing component .................................... 36
Figure 34 Bottom view of the camera chip housing component............................ 36
Figure 35 Side view of the light tube holder component....................................... 37
Figure 36 Top view of light tube holder component.............................................. 37
Figure 37 Bottom view of light tube holder component........................................ 37
Figure 38 Light tube with switch hole marked as orange....................................... 37
Figure 39 Bottom view of the light tube. The smaller radius shows the mounting hole for the collimator................................................................. 38
Figure 40 The tube lid is place on top of the tube to seal it..................................... 38
Figure 41 Image of NBS USAF 1951 Test chart – R70 [28]...................................... 39
Figure 42 Group five is the last readable group of the chart.................................... 39
Figure 43 3D surface plot of group 5, line 2. Clear visible peaks.............................. 40
Figure 44 3D surface plot of group 5, line 3. There are still distinguishable peaks ......................... 40
Figure 45 3D surface plot of group 5, line 4. The middle peak is gone ........................................... 40
Figure 46 3D surface plot of group 5, line 5. The three peaks are replaced by interferences peaks at the
line edges ................................................................................................................................. 40
Figure 47 Comparison between on chip (A, C) and conventional microscope images (B, D) .......... 41
Figure 48 Multiple S. mansoni eggs taken with the on chip microscope ........................................ 41
Figure 49 Kato-Katz extracted eggs. A comparison between on chip (A, C) and a conventional
microscope images (B). ............................................................................................................. 42
Figure 50 S. mansoni eggs in liver tissue. A and C are LOSMic images and B and D are taken with a
conventional microscope .......................................................................................................... 42
Figure 51 S. haematobium eggs taken with LOSMic (A, B, C) and conventional microscope (D, E).
The eggs appear clearly and could be identified ...................................................................... 42
Figure 52 Comparison between LOSMic images (A, C) and conventional microscope (B, D). Eggs
are clearly separated ................................................................................................................. 43
Figure 53 Cracked eggs of Trichuris trichuris ................................................................................. 43
Figure 54 Ascaris lumbricoides eggs. Comparison between LOSMic images (A, C) and conventional
microscope (B, D) ..................................................................................................................... 43
Figure 55 Total field of view. 24.85 mm² ............................................................................................ 44
Figure 56 System topology ............................................................................................................. 45
Figure 57 Three Tier Architecture ............................................................................................... 52
Figure 58 Servlet Process .............................................................................................................. 53
Figure 59 Generate a response using JSP ..................................................................................... 53
Figure 60 Relation in a relation based database ......................................................................... 54
Figure 61 Complete Overview of server application .................................................................... 55
Figure 62 MIDP architecture ...................................................................................................... 56
Figure 63 Life Cycle of midlet applications .................................................................................. 56
Figure 64 Overview of mobile application .................................................................................... 57
Figure 65 General Interaction between mobile and server ............................................................ 60
Figure 66 General interaction between web client and server ........................................................................ 61
Figure 67 Class overview of final server application implementation ...................................... 63
Figure 68 Tile-groups in different zoom levels .............................................................................. 70
Figure 69 Class overview of final mobile application implementation ....................................... 72
Figure 70 Case creation process .................................................................................................. 74
Figure 71 Flow chart describing the send process ........................................................................ 74
Figure 72 Flow chart describing authentication process ............................................................... 76
Figure 73 Flow chart describing the snapshot process .................................................................. 78
Figure 74 Flow chart describing choose of picture process .......................................................... 78
Figure 75 Flow chart describing retrieving GPS data process ...................................................... 78
Figure 76 Storage process of case data ........................................................................................ 81
Figure 77 Fetch stored cases process .......................................................................................... 81
Figure 78 Get messages process ................................................................................................ 82
Figure 79 Flow chart describing search process .......................................................................... 82
Figure 80 Mobile phone re-used in new products, for example dishwasher displays .................. 87
Figure 81 Picture of a schistosomiasis egg of the type S. mansoni taken with conventional
microscope .................................................................................................................................. 98
Figure 82 Adult worm of schistosomiasis [20]. ............................................................................. 98
Figure 83 Picture of a T. trichuria egg taken with a conventional microscope ................................ 98
Figure 84 Picture of an fertilized Ascaris lumbricoides egg taken with a conventional microscope ........................................................................................................................................ 99
Figure 85 Picture of an adult Ascaris lumbricoides [7] .................................................................... 99
Figure 86 Schistosome egg extracted with the Kato-Katz technique ............................................ 101
Figure 87 Mounted specimen with a mounting medium with a refractive index \( n_1 \) less then \( n_2 \) ......... 103
Figure 88 Mounted specimen with a mounting medium with a refractive index \( n_3 \) equal to \( n_2 \) ......... 103
Figure 89 Specimen mounted under cover glass with no immersion oil between cover glass and
sensor ............................................................................................................................................ 104
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Global prevalence’s of some parasitical diseases that are evaluated in this report. See appendix for more information about these diseases.</td>
<td>12</td>
</tr>
<tr>
<td>Table 2</td>
<td>Field of view in optical microscopy.</td>
<td>44</td>
</tr>
<tr>
<td>Table 3</td>
<td>Overview of attributes in the case representation.</td>
<td>58</td>
</tr>
<tr>
<td>Table 4</td>
<td>Overview of attributes in the image representation.</td>
<td>59</td>
</tr>
<tr>
<td>Table 5</td>
<td>Overview of attributes in the activity representation.</td>
<td>59</td>
</tr>
<tr>
<td>Table 6</td>
<td>Overview of attributes included in the account representation.</td>
<td>60</td>
</tr>
<tr>
<td>Table 7</td>
<td>Attributes included in representation of a group.</td>
<td>60</td>
</tr>
<tr>
<td>Table 8</td>
<td>Mobile Client Protocol Specification.</td>
<td>61</td>
</tr>
<tr>
<td>Table 9</td>
<td>Description of server classes.</td>
<td>64</td>
</tr>
<tr>
<td>Table 10</td>
<td>Parameters included in the SMS Send &amp; Receive Request.</td>
<td>66</td>
</tr>
<tr>
<td>Table 11</td>
<td>Parameters included in the mobile location request.</td>
<td>67</td>
</tr>
<tr>
<td>Table 12</td>
<td>Geobytes Request</td>
<td>68</td>
</tr>
<tr>
<td>Table 13</td>
<td>Client application classes</td>
<td>73</td>
</tr>
<tr>
<td>Table 14</td>
<td>Difference in angles when light travels from the cover glass into air.</td>
<td>104</td>
</tr>
</tbody>
</table>