

# Systems Engineering

Infotronics

# Thesis 2014

Michael Schmid

*Multispectral photograph for the  
detection of skin cancer*

Expert

Martial Geiser

Sion, 5. July 2014

SI	TV
X	X

<input checked="" type="checkbox"/> FSI <input type="checkbox"/> FTV	Année académique / Studienjahr 2013/14	No TD / Nr. DA it/2014/73
Mandant / Auftraggeber <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input checked="" type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i> Dr Giannada	Etudiant / Student Michael Schmid  Professeur / Dozent Martial Geiser	Lieu d'exécution / Ausführungsort <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>
Travail confidentiel / vertrauliche Arbeit <input type="checkbox"/> oui / ja <sup>1</sup> <input checked="" type="checkbox"/> non / nein	Expert / Experte (données complètes)	

Titre / Titel <p style="text-align: center;"><b>Multispektrale Fotoaufnahme für die Detektion von Hautkrebs</b></p>
Description / Beschreibung <p>Sehen, was weder mit bloßen Auge, noch mit einer Standardkamera sichtbar ist! Durch eine multispektrale Aufnahme eines suspekten Hautbereichs kann die Konzentration von Melanin, als auch andere optische Parameter einer Haut mit Gewebeeränderung bestimmt werden. Dies wird durch eine sequentielle Beleuchtung von 5-9 SMD LEDs gemacht. Mit einem einfachen optischen System und einer mini CMOS werden multispektrale Bilder aufgenommen. Anschließend werden diese analysiert um eine Vordiagnose zu stellen.</p> Objectifs / Ziele <ul style="list-style-type: none"> <li>- Aufbau des optischen Systems mit Beleuchtung wie in der Semesterarbeit vorgeschlagen.</li> <li>- Testbilder Aufnahme und Voranalyse.</li> <li>- Reelle typische Beispielaufnahmen mit Hervorhebung von Melanin und Blutsauerstoffgehalt.</li> </ul>

Signature ou visa / Unterschrift oder Visum  Responsable de l'orientation Leiter der Vertiefungsrichtung: .....  <sup>1</sup> Etudiant / Student : .....	Délais / Termine  Attribution du thème / Ausgabe des Auftrags: 12.05.2014  Remise du rapport / Abgabe des Schlussberichts: 11.07.2014, 12:00  Expositions / Ausstellungen der Diplomarbeiten: 27 – 29.08.2014  Défense orale / Mündliche Verfechtung: Semaine   Woche 36
---	--

<sup>1</sup> Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme. Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

# INDEX

<b>1</b>	<b>PREFACE</b> .....	<b>1</b>
<b>2</b>	<b>INTRODUCTION</b> .....	<b>1</b>
2.1	What is the problem .....	1
2.2	Similar projects .....	1
2.3	Our solution .....	2
2.3.1	<i>Melanin detection</i> .....	2
2.3.2	<i>Blood analysis (hemoglobin detection)</i> .....	3
2.3.3	<i>Deep view</i> .....	3
<b>3</b>	<b>SPECIFICATIONS</b> .....	<b>4</b>
<b>4</b>	<b>HARDWARE</b> .....	<b>4</b>
4.1	Image acquisition - Camera .....	4
4.2	.....	5
4.2	Optical setup .....	6
4.3	Illumination .....	7
4.3.1	<i>LED ring</i> .....	7
4.4	Casing .....	8
4.4.1	<i>Reference regions</i> .....	8
<b>5</b>	<b>SOFTWARE</b> .....	<b>9</b>
5.1	Image acquisition .....	9
5.1.1	<i>Main window</i> .....	9
5.1.2	<i>Analysis window</i> .....	10
5.1.3	<i>Settings window</i> .....	11
5.2	Analysis .....	12
5.2.1	<i>Optimisation</i> .....	12
5.2.2	<i>Measurement</i> .....	17
<b>6</b>	<b>FUTURE EXTENSIONS</b> .....	<b>21</b>
6.1	Optical system .....	21
6.2	Image acquisition .....	21
6.3	Data storage .....	21
6.4	Analysis .....	21
6.5	Casing .....	21
<b>7</b>	<b>CONCLUSION</b> .....	<b>22</b>
<b>8</b>	<b>SOURCES</b> .....	<b>22</b>
8.1	Literature .....	22
8.2	Figures .....	22
<b>9</b>	<b>DATE &amp; SIGNATURE</b> .....	<b>22</b>
<b>10</b>	<b>APPENDIX</b> .....	<b>23</b>
10.1	Hardware .....	23

10.1.1	<i>LED controller</i>	23
10.1.2	<i>LED ring</i>	31
10.1.3	<i>Parts list</i>	35
10.2	Software - general	36
10.2.1	<i>Class diagram</i>	36
10.2.2	<i>MainTimer</i>	37
10.3	Software - code	38
10.3.1	<i>IdsSimpleLiveDlg (main window)</i>	38
10.3.2	<i>Analysis</i>	44
10.4	Tests	53
10.4.1	<i>Melanin and hemoglobin tests</i>	53
10.4.2	<i>Prove of melanin detection</i>	54
10.5	Future extensions	56
10.5.1	<i>Image acquisition</i>	56
10.5.2	<i>Data storage</i>	57
10.5.3	<i>Analysis</i>	57
10.5.4	<i>Development tips</i>	59
10.5.5	<i>Contact informations</i>	59

# 1 PREFACE

The results of this project aren't just my earnings. Therefore I first want to thank everyone who helped me with patience and competence. A special thanks to my expert Martial Geiser, who particularly supported me with my work. He was always there to answer questions, and to acquire solutions together. A thanks to Frederic Truffer and Helene Strese, they gladly answered questions, and gave me convenient tips. I also want to thank Olivier Walpen and Serge Amoos for helping me with electronics and casing. Finally an acknowledgement to Dr. Gianadda Elisabeth, who supported us with valuable ideas and standards.

This project was a great experience doing a step in the direction of the real working world. Finding different solutions, deciding for one, and then handling the ramifications was an exiting experience, which can't be taught in regular class. Due to the vast scope of this project, it was really interesting to participate the different domains of biology, optics, hardware and software development, and combine them to one extensive project.

## 2 INTRODUCTION

### 2.1 What is the problem

Since skin cancer is the most common kind of cancer in Switzerland, its detection and treatment consumes a high amount of time and money. Due to subjective diagnosis by eye, based on experience of the dermatologist, a surgical operation will take place - or not. Here the product can show his strengths. Substantiate the decision of the dermatologist, or maybe indicate another view. It will be possible to avoid unnecessary surgery, sparing a lot of trouble. A meeting with Dr. Gianadda helped to understand the problem, and led the project into the right direction. We got a view on the FotoFinder, which will be described on the next chapter.

### 2.2 Similar projects

A commonly used system is the FotoFinder. It is a highly developed device with a broad spectrum of sophisticated functionalities. The aviable full body scan, and the vast database are just two of its strengths. The body scan will indicate critical points, wich then can be scanned with the additional fullHD camera. Of course those measurements have an outstanding quality, yet it just provides one RGB image with the entire light spectrum.



Figure 1: FotoFinder

Here comes the opportunity of our device, which provides a multispectral imaging. This multispectral imaging not only shows the surface, but also allows a deeper view into the skin. A melanin analysis, and also a oxy/deoxy - hemoglobine determination is possible with those capabilities.

This project is based on a previous solution, which was developed with a similar specification and the same purpose. The initial goal was to make the device smaller, and to develop further software. Yet the result is a completely independent solution. [1]

## 2.3 Our solution

The main method lies in the melanin, but also in the hemoglobin detection. It can be achieved by illumination with particular light spectrums. The camera measures the amount of reflected/scattered light, which can give indications to a possible risk.

### 2.3.1 Melanin detection

The melanin detection becomes possible through the known absorption behaviour of the entire light spectrum. Specific wavelengths can be measured, and then compared to the original absorption function. Following graph shows the chosen wavelengths for the measurement. A point below 390nm can't be implemented because of the sensivity of the camera. Nevertheless, the available wavelengths are perfectly adequate.

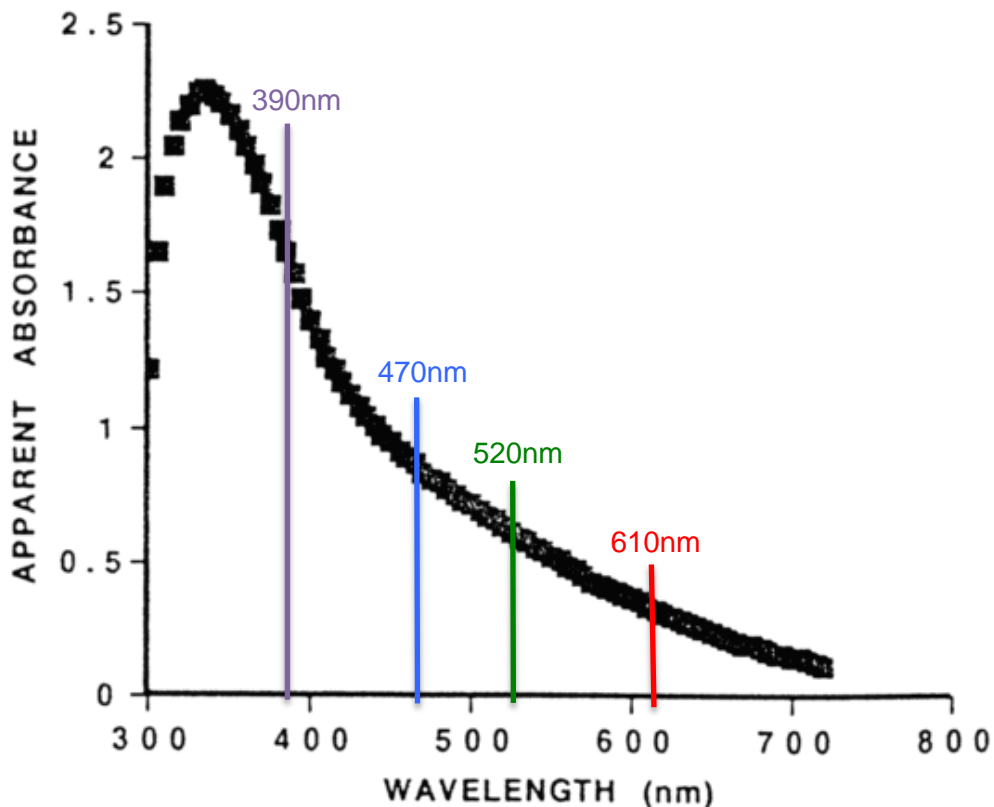


Figure 2: Absorption spectrum of melanin

### 2.3.2 Blood analysis (hemoglobin detection)

The blood analysis uses a procedure quite similar to the melanin detection. The known graph of oxy and deoxy-hemoglobin transmission is essential. A smart choice of two specific wavelengths already enables the possibility to distinguish oxygen rich, and oxygen poor hemoglobin. One point is set where the sensitivities are very close, the other point stands for a maximal difference. The ratio of these two values then indicates the amount of oxy and deoxy-hemoglobin.

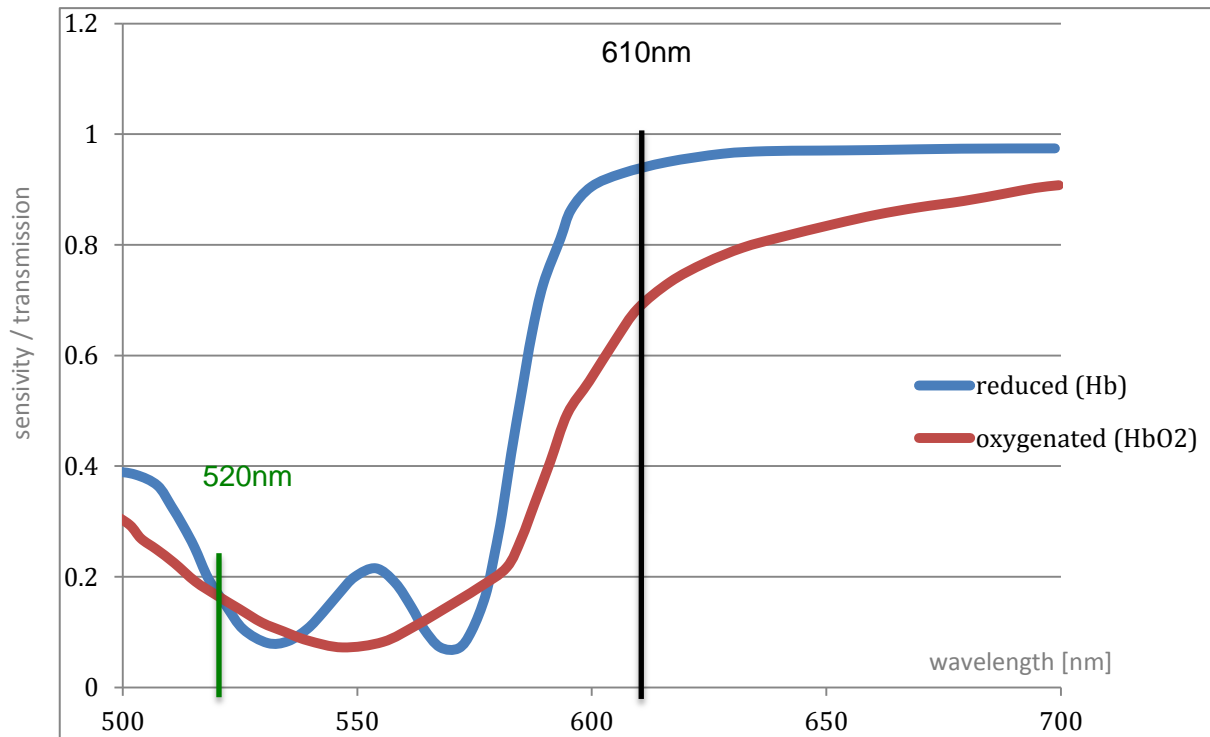


Figure 3: hemoglobin transmission

### 2.3.3 Deep view

Beside the methods shown, the remarkable feature of looking into the different layers of the skin is definitely worth to mention. Especially the infrared LED with 880nm wavelength enables a really deep view beyond the epidermis. This should allow locating aberrations even deep under the skin. Following measurement shows the speciality of every LED. Note the vein, which was found in deeper regions.

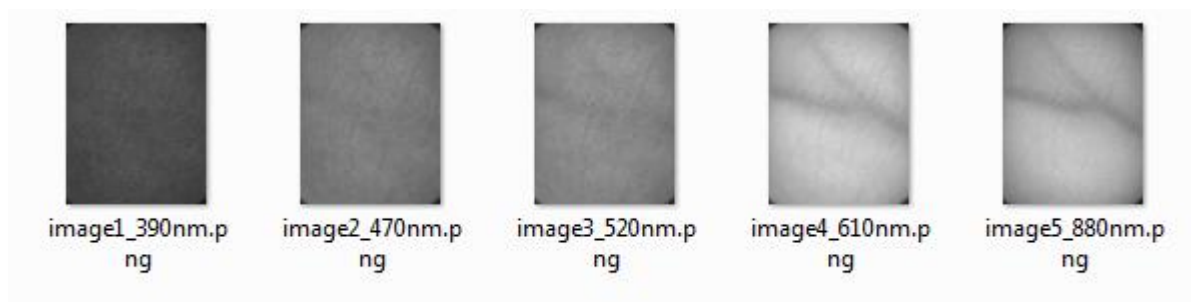


Figure 4: Measurement example

## 3 SPECIFICATIONS

This project attends to the task of taking a multispectral image acquisition of a specific skin region, with a diameter of 1 Inch (about 25mm). The five images will pass through an analysis algorithm, which confirms or disproves the suspicion of skin cancer. The product should provide following functions:

**Calibration:** The device has two calibration functions. One essential main calibration with a reference, but also a small gauging with each measurement.

**Image acquirement:** A measurement should be very simple. A button on the camera will provide maximal usability. Also the process should not take longer than a second.

**Display:** After the measurement a brief overview of the taken images will be shown. Which images exactly, (raw data / analyzed data / combo) will be determined in a later instance.

**Analysis:** The software provides an evaluation with following criteria:

- Determination of melanin amount.
- The ratio of Oxy- and Deoxyhemoglobine.
- Analysis of contours.

**Database:** Every measurement will be stored properly.

**Properties:** The software contains different setting possibilities. (such as contrast, exposure time ect.)

## 4 HARDWARE

### 4.1 Image acquisition - Camera

The recent semester project intensively approached the choice of the right camera. The decision fell to the UI-1222E from the german company "uEye". It is a CMOS camera, also available with mono technology. This is a big advantage for our purposes, due to its improved dynamic range. The goal was to go with the newest and best available technology. The reward was a professional camera with outstanding characteristics and very good documentation, which allowed determined and effective development. Following table shows the most important technical specifications. For more details and a very detailed documentation explore the CD.



Figure 5: UI-1222E CMOS camera



Sensor specification	
Sensor type	CMOS
Shutter system	Electronic global shutter
Characteristic	Linear with knee points
Readout mode	Progressive scan
Resolution class	WVGA
Resolution	752 x 480 pixels (0.36 Megapixel)
Aspect ratio	5:4
<a href="#">Bit depth</a>	10 bits
Optical sensor class	1/3 inch
Exact sensitive area	4.512 mm x 2.880 mm
Exact optical sensor diagonal	5.35 mm (1/2.99 inch)
Pixel size	6.00 $\mu\text{m}$ , square

Figure 6: Camera specifications

The relative sensor sensitivity is definitely worth a special mentioning. This camera can see widely beyond the human eye, or other similar cameras. It allows the usage of a 880nm LED. The detailed graph is shown below.

## 4.2

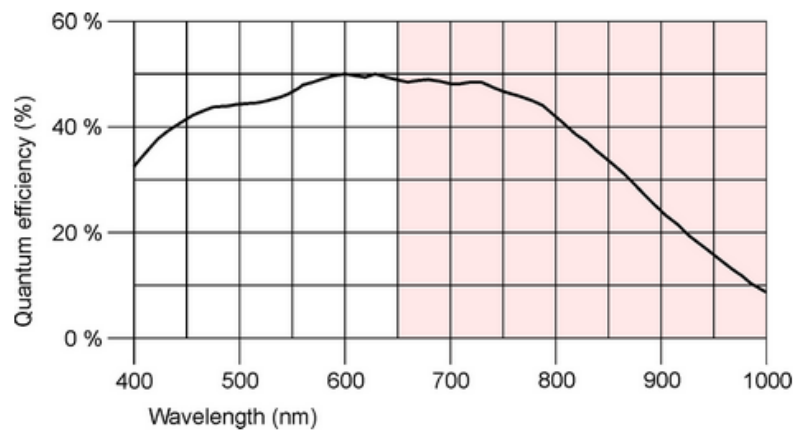


Figure 7: Quantum efficiency

## Optical setup

### 4.2.1.1 Lense setup

The first prototype in the semester project [2] showed that a single lens can't achieve the desired quality demands. Therefore a system with two lenses was calculated, and implemented. This time an achromatic lens was used, due to the error correction of different wavelengths. Following two lenses were used:

- AC 050 - 010 - B
- Bi. +; 50/12,7 gef.

The distance between object and first lens is 83mm, the distance sensor  $\leftrightarrow$  second lens amounts to 2mm. Yet it can't be measured properly, since the montage doesn't allow it. The same for the distance between the two lenses, which is around 10mm. Nevertheless those values are not important, since a new optical system would require a new calculation anyway.

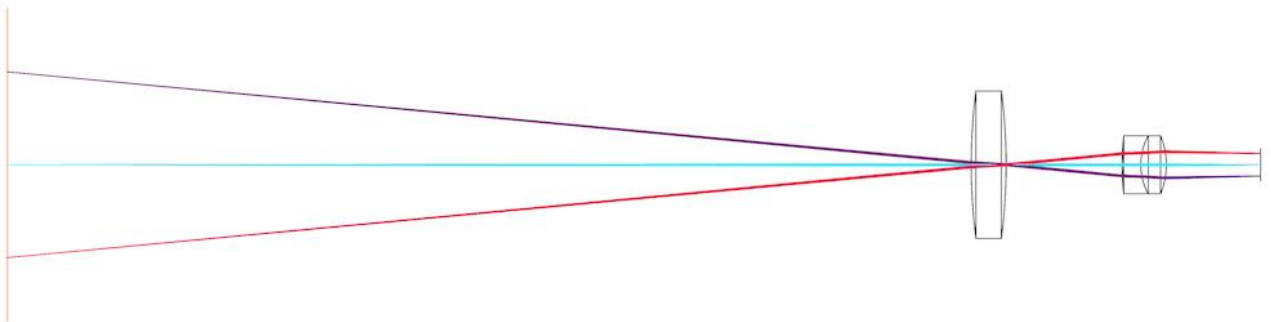


Figure 8: Optical system simulation

The assembly of the optical system was a challenging task, because a small variation in the distance between the optical elements already affects the image quality. The laser cutting machine was used to create the required parts. The plans are available on the CD.

**HINT!** The lenses are clamped with screws, which should hold fine. Still, I recommend a careful usage. Never shake the device! A very small shift of a lens could already have impact on the image quality.

### 4.2.1.2 Pinhole

Already in the semester project we had too much light on the sensor. This problem was solved by a pinhole, which also improved the image quality. Different versions between 0.5mm and 2mm were tried, while the 2mm had the best acuity. The pinhole is directly attached to the inner side of the second, big lens.

## 4.3 Illumination

### 4.3.1 LED ring

The key to a good measurement is the proper lighting of the object. The assembly of the LED's plays a crucial role to a homogenous picture. After some tests it has been shown that a specific angle of the LED's is not absolutely required. This enables the possibility to design the LED ring with a simple PCB. Still, the assembly has to be a perfect circle. It led to some major problems with PCAD (software for PCB design) since it is not intended to work with round objects. Nevertheless the board was created. It is shown in the picture below. For schematic, PCB layout, calculations and partlist see *appendix\_10.1.LED*

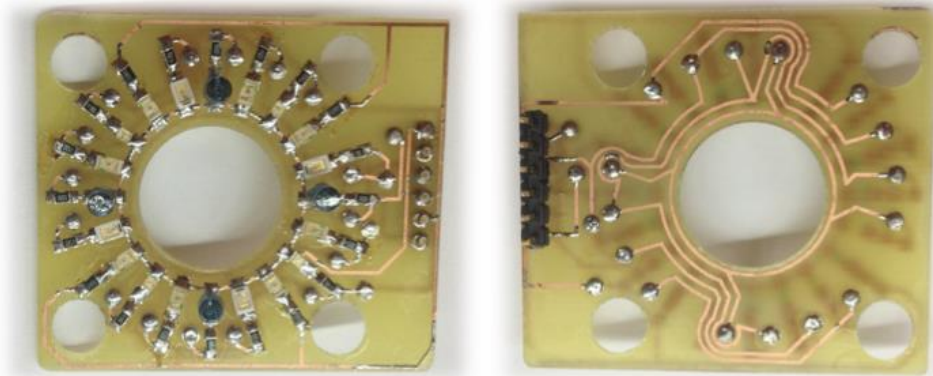


Figure 9: LED ring PCB

The camera provides a very useful feature with its given inputs/outputs. Unfortunately it hasn't enough outputs to actuate every LED separately. Therefore a simple solution with a PIC was found. The trigger button is connected with the camera as well as the PIC. Now the start signal will launch both parallel. The PIC is programmed to power on each LED group for 200ms. This gives the camera enough time to take the picture. The PIC was implemented on a separate PCB, which can be easily connected with the LED-Ring, as shown in the following picture. For schematic, PCB layout, partlist and details about the PIC programming see *appendix\_10.1*. The Led ring and the PIC controller share the same partlist.

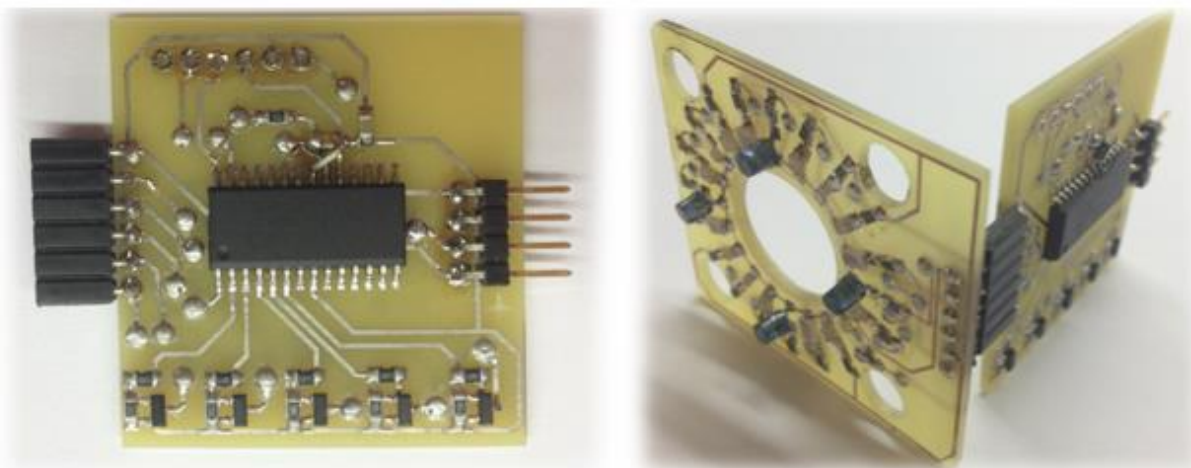


Figure 10: LED controller PCB

## 4.4 Casing

The casing was realised with the 3D printer. Serge Amoos created the plans in Autodesk Inventor. They are available on the CD. Following picture shows a view of the 3D model.

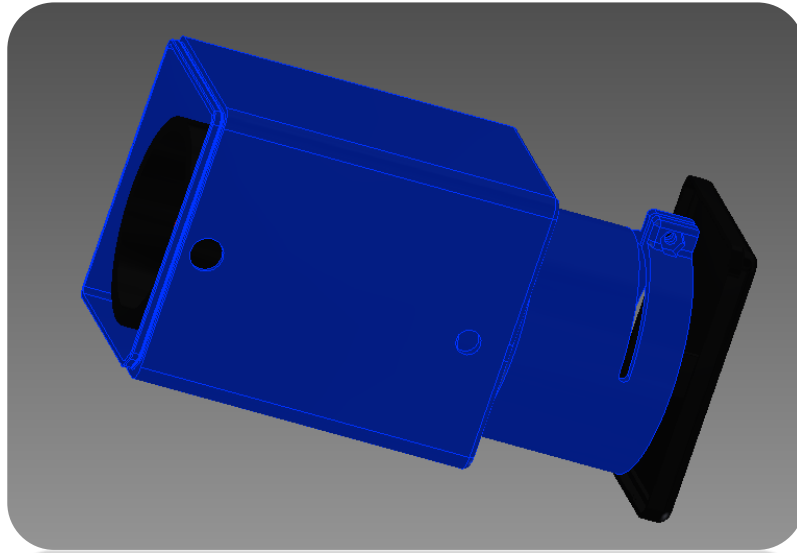


Figure 11: 3D model of casing

### 4.4.1 Reference regions

The white regions are crucial for the gain adjustment of each measurement. This part is basically made out of a simple ring, created with the laser cutting machine (plans on CD). Two stripes of thick paper are glued to this ring. They have a width of 10mm. It is very important to keep the ring on the right angle during the measurement.



Figure 12: Calibration region

**HINT!** The raw data will include the reference regions. A black rectangle indicates the region used for the adjustment. Make sure that the entire rectangle only contains the designated white region.

## 5 SOFTWARE

### 5.1 Image acquisition

#### 5.1.1 Main window

The data acquisition should be as simple as possible. Pushing the button will trigger the camera and the Illumination. A user-friendly and simple GUI will help to take pictures and make an analysis with a minimum effort. Following picture shows the user interface that will appear when the user opens the program. It provides a big display window for a live video stream, an optimal preview for measurements. The window provides following options:



Figure 13: main window

- **Next:** Make a quick measurement, without the need of opening the analysis window. It will make the analysis automatically and save the results. When the "Show quick Result" checkbox is open, it will display the result of the melanin and blood analysis.
- **Analyze:** This button will open the Analysis window for further options.
- **Patient Informations:** Please enter the patient information here.
- **Calibration Mode:** This enables the calibration mode. In this mode an absolutely homogenous and white surface is needed. The images will be saved into images/Calibration, where they can be used as reference for a proper calibration.
- **Settings:** Open the settings for exposure time adjustment, or loading parameters.
- **Exit:** Close the program.

### 5.1.2 Analysis window

This window allows custom options for the measurement. An example body represents the patient, where the location of the measurement can be indicated. The patient informations will be taken automatically from the main window.

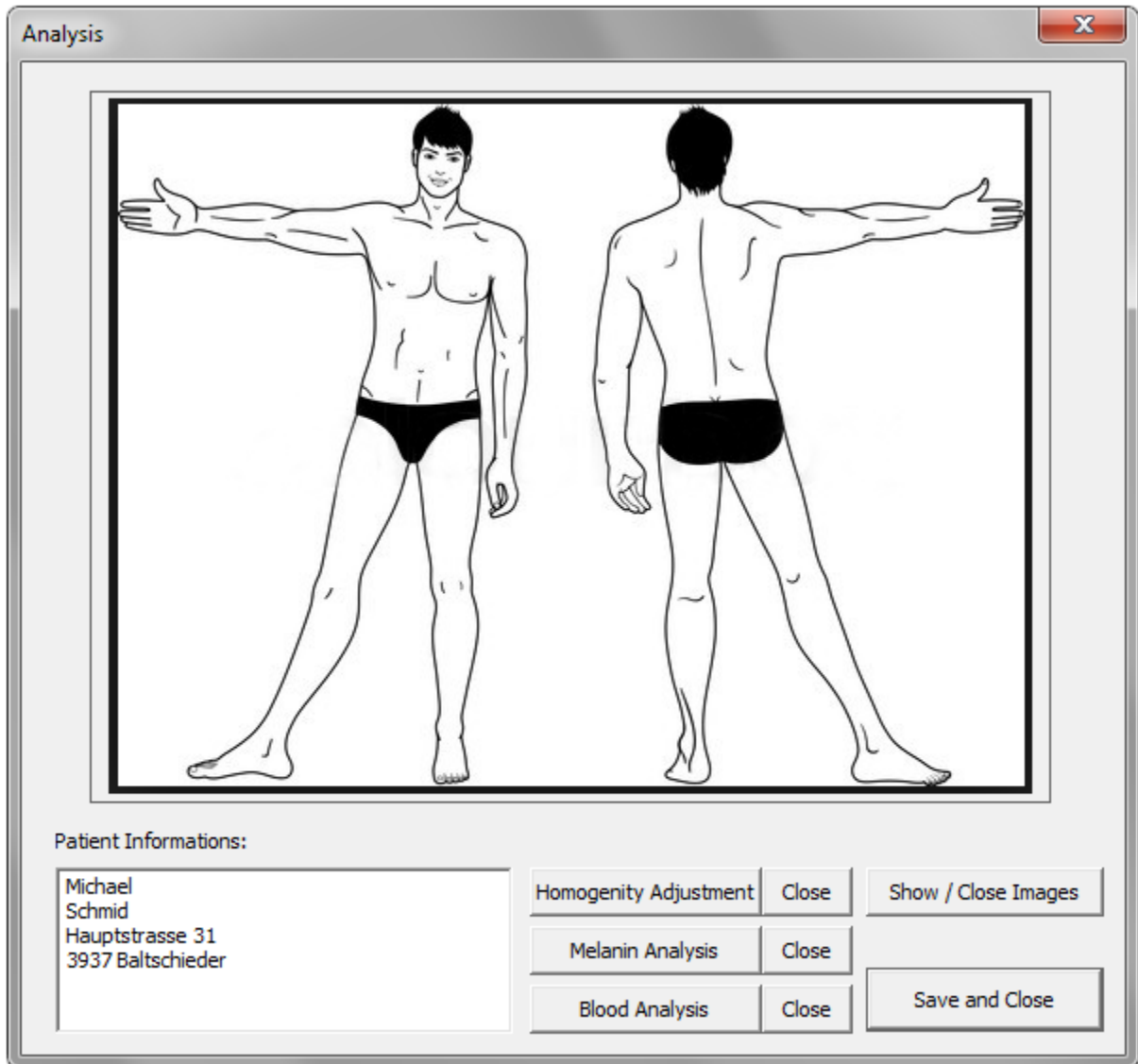


Figure 14: analysis window

The window provides following options:

- **Homogeneity Adjustment:** Since the homogeneity adjustment is the most time consuming procedure, it is hereby available to perform voluntary. After the adjustment procedure the results will be shown. They can all be closed at once with the "Close" button near the initial button.
- **Melanin Analysis:** This button will do the melanin analysis, and show the results. Also these images can all be closed at once with the nearby "Close" button.
- **Blood Analysis:** Same procedure for the oxy/deoxy-hemoglobine determination.
- **Show / Close Images:** This button will show or close all taken pictures.
- **Save and Close:** Be sure to use this button to save your results.

### 5.1.3 Settings window

This window is supposed to perform desired changes to the exposure times. Try not to vary too much from the recommended values since they are well balanced. Of course the gain adaptation will always correct the difference, still it would bring unwanted noise into the system. The exposure times will be saved in settings.txt in the Parameters folder. With the "Load Parameters" button, it is possible to load earlier saved camera parameters from the Software "uEye Cockpit". It is recommended to stop the livestream for that. Note that this option hardly was investigated because it wasn't needed at all. There may be further adjustments needed to properly load given parameters.

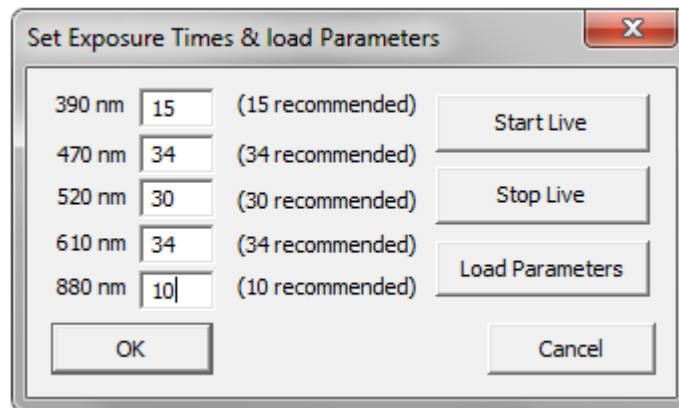


Figure 15: settings window

## 5.2 Analysis

Since important operations are pretty much implemented with fitting methods, almost every capture will be closely linked to an actual function. If so, the name of the method will be mentioned right under the title.

### 5.2.1 Optimisation

Before an actual measurement can take place, the images need to be optimized. Even with good balanced exposure times and proper illumination, there are slight differences between the images, which would decrease the accuracy of the measurement. The optimisation takes place in two steps. First the **gain** will be adjusted, to put all images on the same level. Secondly the **homogeneity** will be evened. The exact procedure is explained in this chapter. An overview is shown in the following exaggerated picture, to emphasize the process.

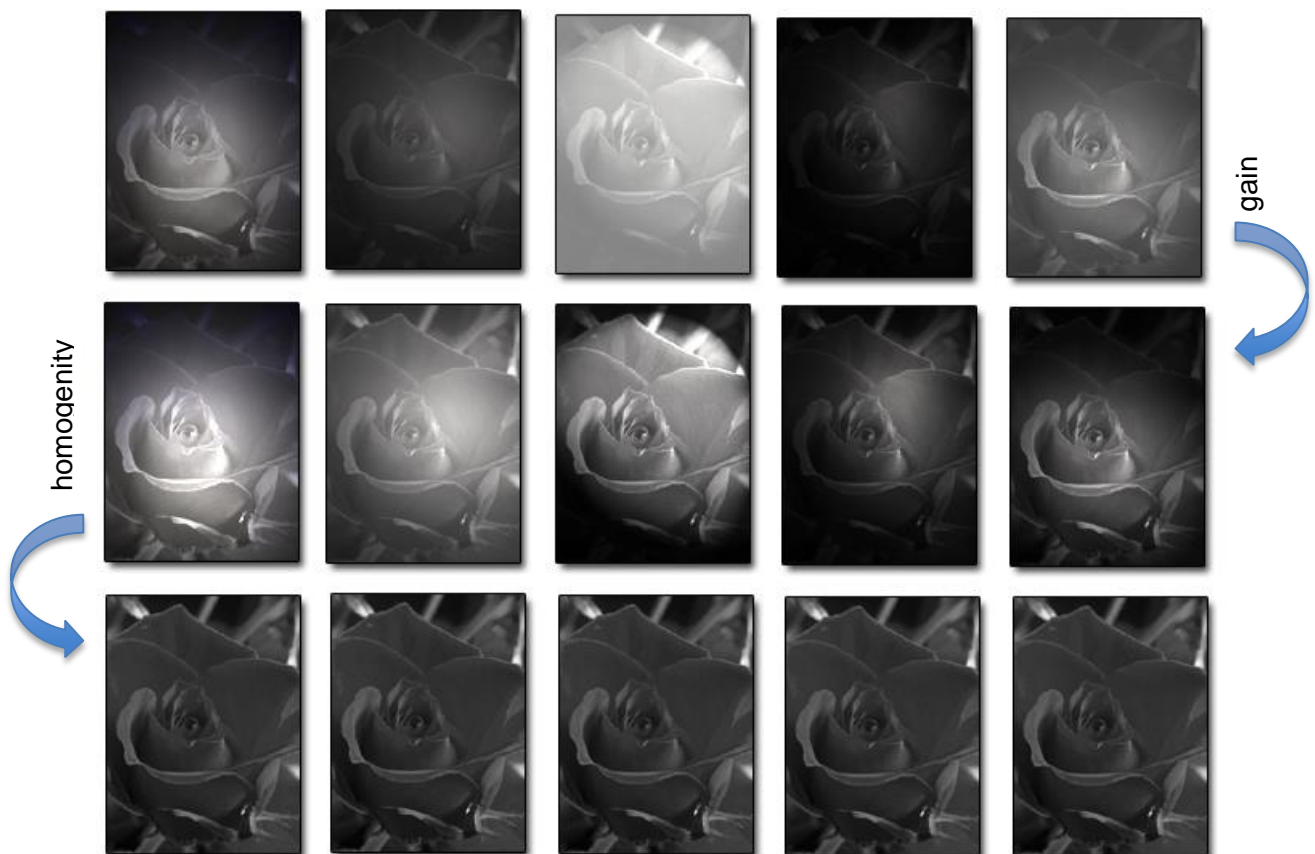


Figure 16: example of gain and homogeneity adjustment

#### 5.2.1.1 Set gain of an image

Software solution in *appendix\_10.3.2.2*

This Method has a really simple task. It takes an image (Mat), adjust its gain, and then returns the newly calculated image. There are two parameters available (alpha, beta). Following formula shows how they work:



$$result_{(i,j)} = value_{(i,j)} * alpha + beta$$

The function will recalculate every single pixel with this formula. Alpha is the gain factor, while beta works as offset.

### 5.2.1.2 Even out the images with gain adaption

Software solution in *appendix\_10.3.2.3*

The goal of this method is to adjust the gain of all images to a common level. Each image has two white calibration regions on its right and its left. The average value of this regions will be calculated. Usually these values should be quite similar. If they vary by an increased amount, there are two possibilities of irregular lighting. Either a LED is not working or illuminating weak, or there is an external lightsource disturbing the lighting. It could also be that the calibration papers are not attached properly or with a different angle, the rectangles will hit a part of skin instead of paper. A warning message will show if one of these problems occurs. If everything runs right the mean of these two average values will provide the proper gain adjustment.

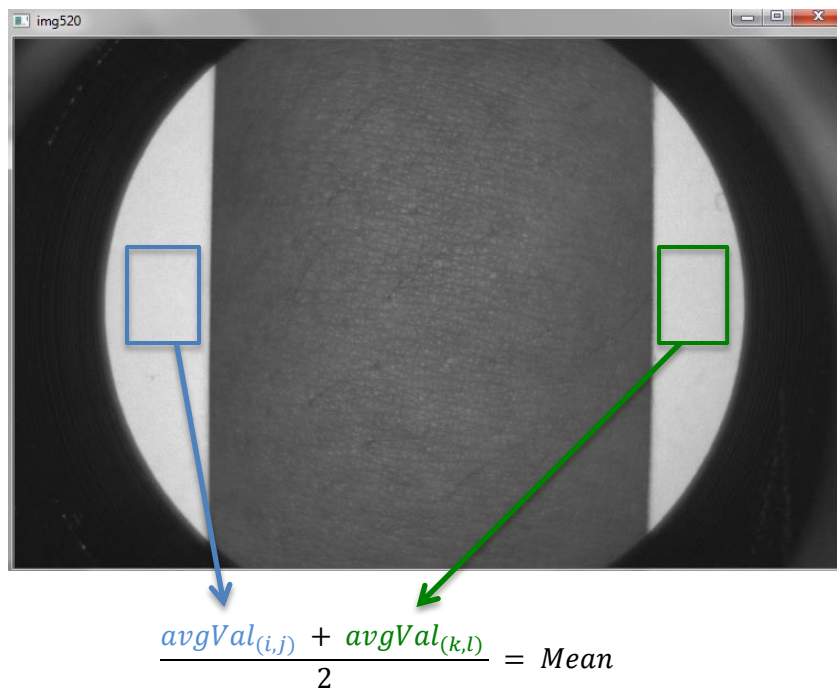


Figure 17: reference regions

The intended gain can now be easily calculated by following formula:

$$Gain = \frac{255}{Mean}$$

Now a call of the setGain() Method will do the actual gain adjustment. The value was carefully chosen to bring the white calibration region of each image into saturation (max value = 255).

### 5.2.1.3 Homogeneity issue

One of the most important requirements for a proper measurement is a good homogeneity of the image. It is almost only influenced by the lighting. Since the solution for the illumination already attempts to achieve an optimal result, there must be an additional software support for an optimized homogeneity of the picture. This will be achieved with the help of a calibration. Therefore an image of a clean white surface will work as a reference. From this reference a pattern will be created which can be utilized to even out further measurements.

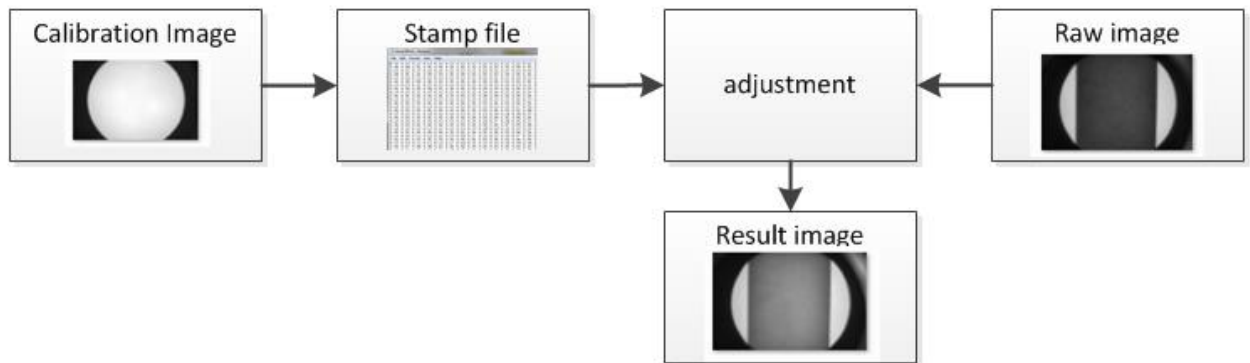


Figure 18: overview of homogeneity adjustment

The actual procedure works like a compressor (in audio domain). There will be a certain threshold, in this case the mean (average) value of the picture. So every value that lies above the threshold, will be attenuated. Every value below it, will be increased. In the end the dynamic range will be much smaller, the values are closer to each other, which means the picture has a better homogeneity.

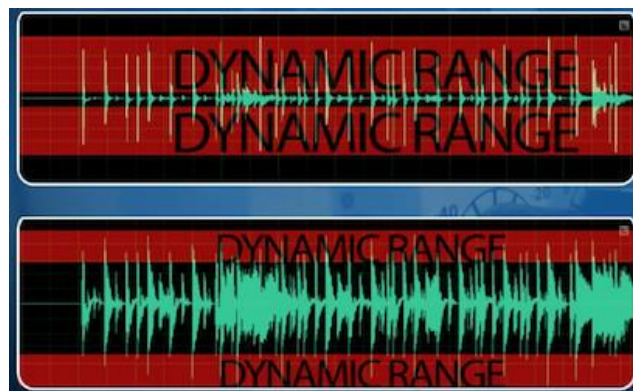


Figure 19: example of reduced dynamic range

### 5.2.1.4 Create the stamps

Software solution in *appendix\_10.3.2.4*

The name of this method was chosen, because it creates a file comparable to a stamp. The stamp then can be used on an image, to alter its appearance (in this case homogeneity). So the goal is having a specific correction-factor for each single pixel. The created textfile will contain all this information.

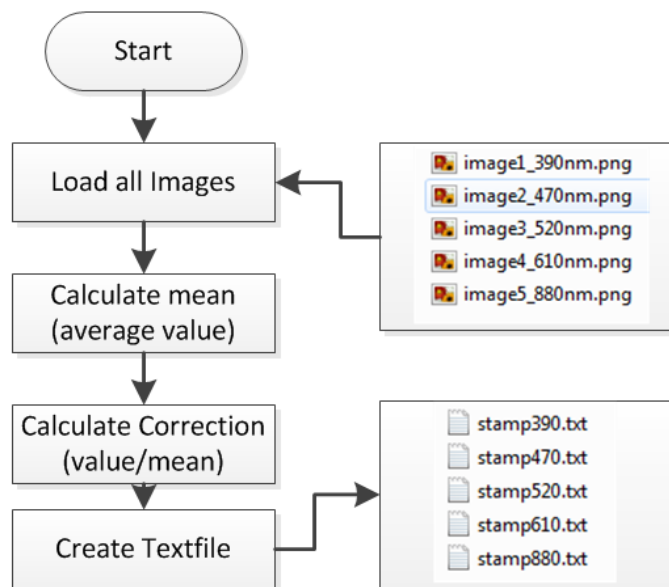


Figure 20: flowchart of stamp creation

First the calibrationImage will be loaded. They are stored in /images/Calibration. Then the average value of all picture will be calculated individually, and stored in the mean[] Array. (Don't confuse it with the mean for melanin detection, which is a mean of all the images together)

$$mean_{(i,j)} = \frac{\sum_{i,j=0}^{i*j} value_{(i,j)}}{i * j}$$

Now a loop will go through the entire image, calculating the correction factor for each pixel with following formula:

$$correction_{(i,j)} = \frac{value_{(i,j)}}{mean}$$

Because of the fraction, the correction now is a value around 1 with several decimal places, which results in a double. Considering that the images just have a Byte per pixel, the output can't be another image. Since there is the need of a permanent accessibility of this data, there has to be another way to store it. Unfortunately the installed Visual Studio does not support many database possibilities. Therefore the data is stored in stamp-textfiles, which will be created in the /Parameters folder.

Please note the need of an absolute exact and unambiguous format of the textfiles. Already the slightest change can result in absolute unreadability. Like the following little example shows, the values are just separated with a space " " .

### 5.2.1.5 Adjust the homogeneity

Software solution in *appendix\_10.3.2.5*

Now that the stamps are ready to use, this method finally takes an image, uses the stamp(correction) on it, and returns a proper homogeneity-adjusted result. It again loops through all the pixels, and makes a division of the pixelValue by the correction.

$$adjustedVal_{(i,j)} = \frac{pixelValue_{(i,j)}}{correction_{(i,j)}} * factor + offset$$

There is a chance that the adjusted value can hit a higher value than the maximal allowed (> 255). This would result in a glitch, where the value becomes zero (black). To avoid this problem, a **factor** and **offset** was added. With a suggested factor of 0.8 the function won't hit any saturation.

**HINT!** If the image shows randomly spread black dots, try to use a factor < 1, and/or a negative offset, to not hit saturation!

Since the adjustment is really time consuming, the parameter "i" allows to take a specific image.

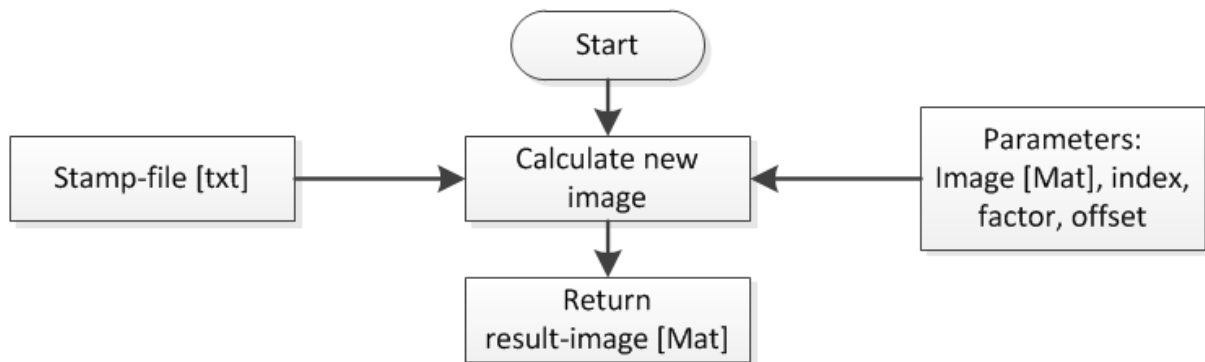


Figure 21: Flowchart of homogeneity adjustment

Following picture shows a successful homogeneity adjustment on a test image.



Figure 22: Homogeneity adjustment test

## 5.2.2 Measurement

In this chapter we will see the actual analysis of the measurement. These functions will implement the investigation of signs for skin-cancer. There are three important symptoms, which will be the focus of this Project:

- high amount of Melanin
- anomaly of oxy-and deoxy-hemoglobin
- irregular shapes of birthmark

Again, every of these procedures is implemented with a fitting method.

### 5.2.2.1 Melanin analysis

Software solution in *appendix\_10.3.2.6*

One of the most important symptoms, a highly increased amount of melanin. The most obvious sign of a mutation of melanocytes. Now the first and most important task is the detection of the melanin in the first place. The chosen solution is to compare the taken pictures with the known light absorption graph of melanin (see 2.3.1). For better access, the Graph was imported to excel with help of WebPlotDigitizer [3]. Since it makes more sense to work with the permeability instead of the absorption, a transformation with following formula took place:

$$permeability(\lambda) = e^{-absorption(\lambda)} * 255$$

The transformation itself would result in a value between zero and one. So the factor of 255 is used to stretch the value over the available range of a Byte. Also the values below 380nm were removed, since the lowest available wavelength is 390nm. The camera probably would not be able to get any lower. Following picture shows the before-after Graph.

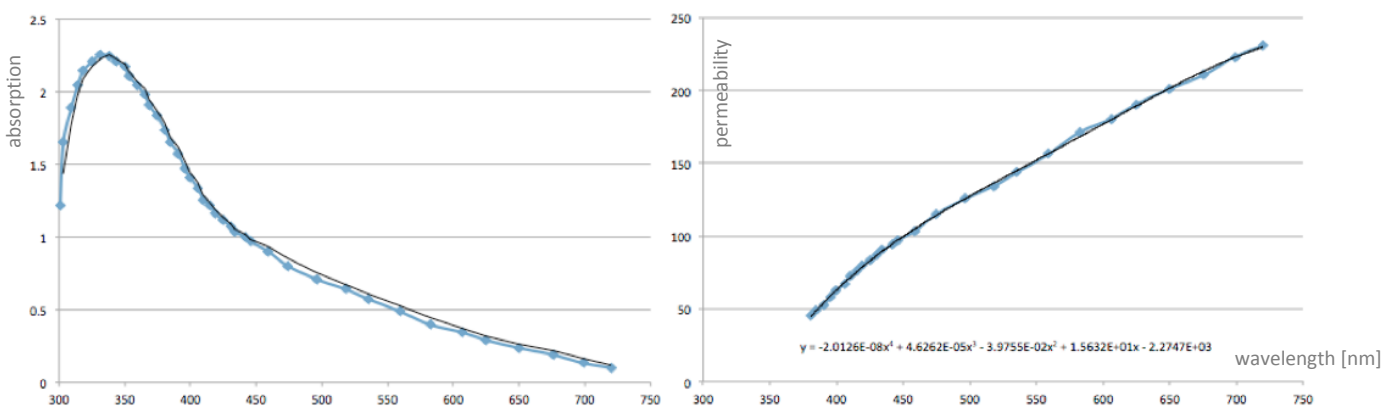


Figure 23: Melanin absorption to permeability transformation

Now the interesting part about the import into Excel is the possibility to create an approached trendline polynomial function. Already a function with 4th order turned out to be quiet accurate. So here is the final function:

$$y = -2.0126E-08x^4 + 4.6262E-05x^3 - 3.9755E-02x^2 + 1.5632E+01x - 2.2747E+03$$

For easy use, there is a method implemented which will return the permeability value in function of the wavelength: `double melaninFunction(double wavelength)`

Now we have everything to get the factor between the desired curve, and the actual measured image. The factor can be calculated with following formula:

$$factor_{(i,j)} = \frac{value_{(i,j)}}{melaninFunction(\lambda)}$$

The value of the factor will roughly vary between 0 and 2, where 1 means that the measured and the intended melanin value are identical. Now for a better visual overview, the function above will be extended, and now saved in an image instead of an array.

$$factorImg_{(i,j)} = factor_{(i,j)} * 128$$

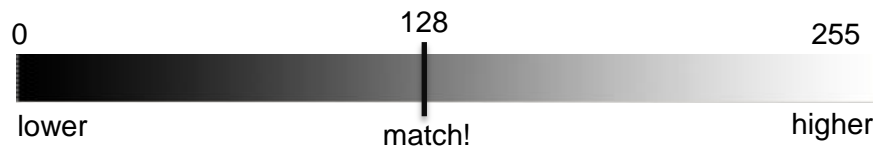


Figure 25: Dial for the factorImage

Assuming a 100% match, the entire factorImage would be grey-> 128. If it goes more in direction of white, (128+) it means that the measured value is higher than the melanin value. If it goes in direction of black (128-), the measurement is lower than the melanin value. Of course this will count for every pixel, and result in an image.

Now the five images will be merged to only one image, to gain some better overview. Following function calculates the mean (average) of all images, and saves it to a single new image.

$$mean_{(i,j,k)} = \frac{\sum_{k=0}^5 value_{(i,j,k)}}{5}$$



Figure 24: example of melaninMean

Last but not least, there has to be a little calibration to adapt the values to real world environment. This is achieved with the calibration factor melaninIndex. This index is based on a dark skin sample.

$$mean_{(i,j)} = mean_{(i,j)} * melaninIndex$$

And there we have a result! The brighter the Image, the more melanin was detected.

Beside the mean of the melanin, there is another result image available. It represents the standard deviation. Every pixel will be calculated with following formula:

$$melaninSD_{(i,j)} = \sqrt{\frac{1}{k} * \sum_{k=0}^k (value_{(i,j,k)} - mean_{(i,j,k)})^2}$$

### 5.2.2.2 Prove of melanin detection

This chapter will prove that the analysis detects actual melanin. Of course there will be always other factors that can influence the measurement, but following tests should show that it really focuses on melanin. Please consider that the last image (610nm) is not very much affected by melanin, so its value is rather negligible. The excel files with the calculations and graphs can be found on the CD. For more details about the tests see *appendix\_10.4.2*.

Following graph shows a summary of the important measurements. It is clearly visible that the two "tricked" examples are way off, where the actual skin scan shows a surprisingly high accuracy.

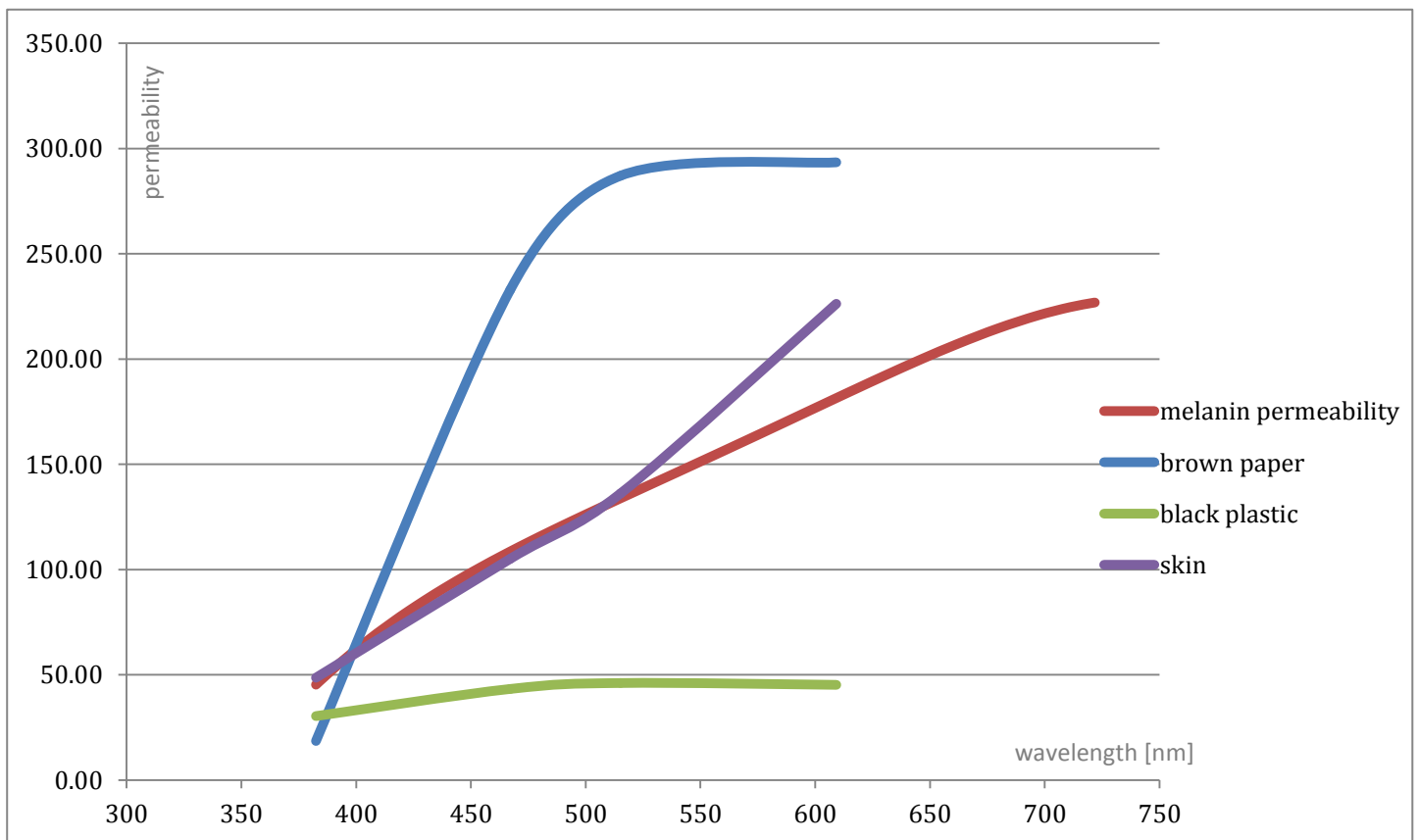


Figure 26: Measurement of different surfaces

### 5.2.2.3 The evaluation

After the validating tests we can assure that the measured graph represents pretty much the melanin permeability. But since it is a relative value, how do we determine the effective amount of melanin? For this procedure a calibration with help of the calibrationFactor is necessary. Following graph shows just two measurements - a brown and a white skin.

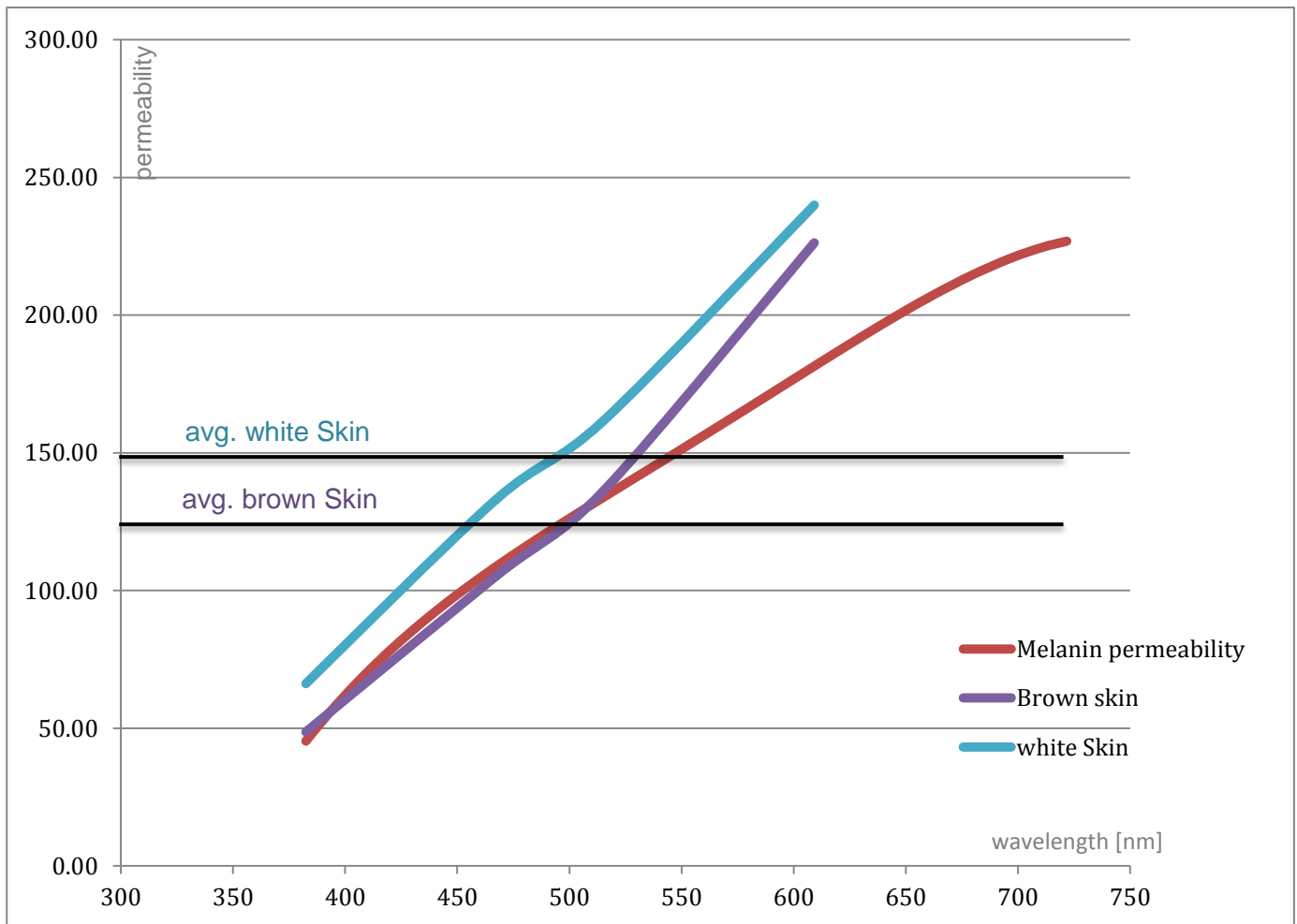


Figure 27: Evaluation of melanin measurement

The very simple solution is to choose a calibrationFactor setting, so that the darkest skin example exactly matches the melanin permeability curve. It now represents the reference. Now every measurement with less melanin will be above this value. Since it is a dark skin color, it is unlikely that a regular measurement will go much deeper than this value. If the value has a distinct low value, it can be considered as potential hazard. To simplify the result, an average value of the 4 wavelengths is sufficient. This procedure will be done for every single pixel, and finally be accessible in the melaninMean result image.

#### 5.2.2.4 Blood analysis

Software solution in appendix\_10.5.3.1

Since the focus of the project was on the melanin detection, the blood analysis is held very simple. One single operation will take the 520nm, and the 610nm image for a ratio calculation (see 2.3.2). Following formula is used:

$$result_{(i,j)} = \frac{val520_{(i,j)}}{val610_{(i,j)}} * 128$$

The factor of 128 has the same purpose like the melanin measurement before, it displays a value between roughly 0 and 2 with grayscales. For proper use there is a need of some tests and a calibration, which is not implemented yet.



## 6 FUTURE EXTENSIONS

Unfortunately the scope of this project often exceeded the available timeframe. Therefore compromises had to be made. Since this project will be continued, this chapter will mention a few points worth of improvement.

### 6.1 Optical system

The lenses are assembled in a construction rather determined for test purposes. Surely it absolutely accomplishes its task, yet the system could be smaller. Also most of the weight of the device comes from the optical system. The camera has additional connection possibilities to support a much smaller system. I would highly recommend a new customised optical system.

### 6.2 Image acquisition

The camera currently needs 200ms per image. However, if a faster acquisition is needed, there would be the possibility to use the camera intern buffer. Also a I<sup>2</sup>C communication between PIC and camera processor would optimize the duration of the illumination. For details see *appendix\_9.5.1*.

### 6.3 Data storage

The current project creates a folder for each patient, and saves the measurements with the actual date and time. A more professional solution would be a SQL database, which would allow a wide range of customisation. Also online access for further analyses, processing or just data exchange would be possible. For an already worked out concept see *appendix\_9.5.2*.

### 6.4 Analysis

Fortunately the melanin analysis could be finished with great results. Yet the hemoglobin and contour methods are just prototypes. They definitely need some further development. This is probably the most important task to begin with. Still, with some tests and a proper calibration, the hemoglobin procedure already may yield some usable results.

### 6.5 Casing

Unfortunately the time run out to improve the casing. An increased diameter of the tube would allow a larger usable image. Also the height should be slightly increased for a better montage. An additional fixture of the reference regions would prevent them from shifting. Nevertheless I would recommend a revised optical system, which would imply an entire new casing.

## 7 CONCLUSION

The choice of the camera turned out to be rewarding, and also the range of wavelengths brought good results. The setup was changed to 390nm, 470nm, 520nm, 610nm and 880nm, where especially the infrared LED enables some interesting opportunities. The final results are very persuasive, a melanin determination with such high accuracy is a solid cornerstone for this device. I see great potential for the future, where targeted development for optimized oxy and deoxy - hemoglobin determination will complement the device to a great product. The outline analysis may be redundant, since the FotoFinder already includes this feature. A redesign of the optical system will allow a much lighter and smaller case, allowing it to be competitive to other products.

## 8 SOURCES

### 8.1 Literature

[1] Documentation of previous project, available on CD in *illustrations* folder

[2] Semester Project available on CD in *illustrations* folder

[3] WebPlotDigitizer: Program for graph transcription <http://arohatgi.info/WebPlotDigitizer/>

### 8.2 Figures

**Figure 1:** <http://www.fotofinder.de>

**Figure 2:** <http://www.cl.cam.ac.uk/~jgd1000/melanin.html>

**Figure 3:** <http://biomedicaloptics.spiedigitallibrary.org/article.aspx?articleid=1102785>

**Figure 6:** uEye Documentation

**Figure 7:** uEye Documentation

**Figure 8:** Simulation

**Figure 16:** Photoshop

**Figure 19:** <http://www.youtube.com/watch?v=91qs3fux5HY> -> Audio Compressor explained

## 9 DATE & SIGNATURE

Sion, 11. July 2014

Michael Schmid

# 10 APPENDIX

## 10.1 Hardware

### 10.1.1 LED controller

#### 10.1.1.1 PIC Programming

##### 10.1.1.1.1 Timer

For further easy to use timing, a timer with a 10ms elapse will be implemented. To achieve this interval, a Timer0 Value of 0x63BF is used. With this timer every exposure time can be achieved with a factor of 10ms.

##### 10.1.1.1.2 Interrupts

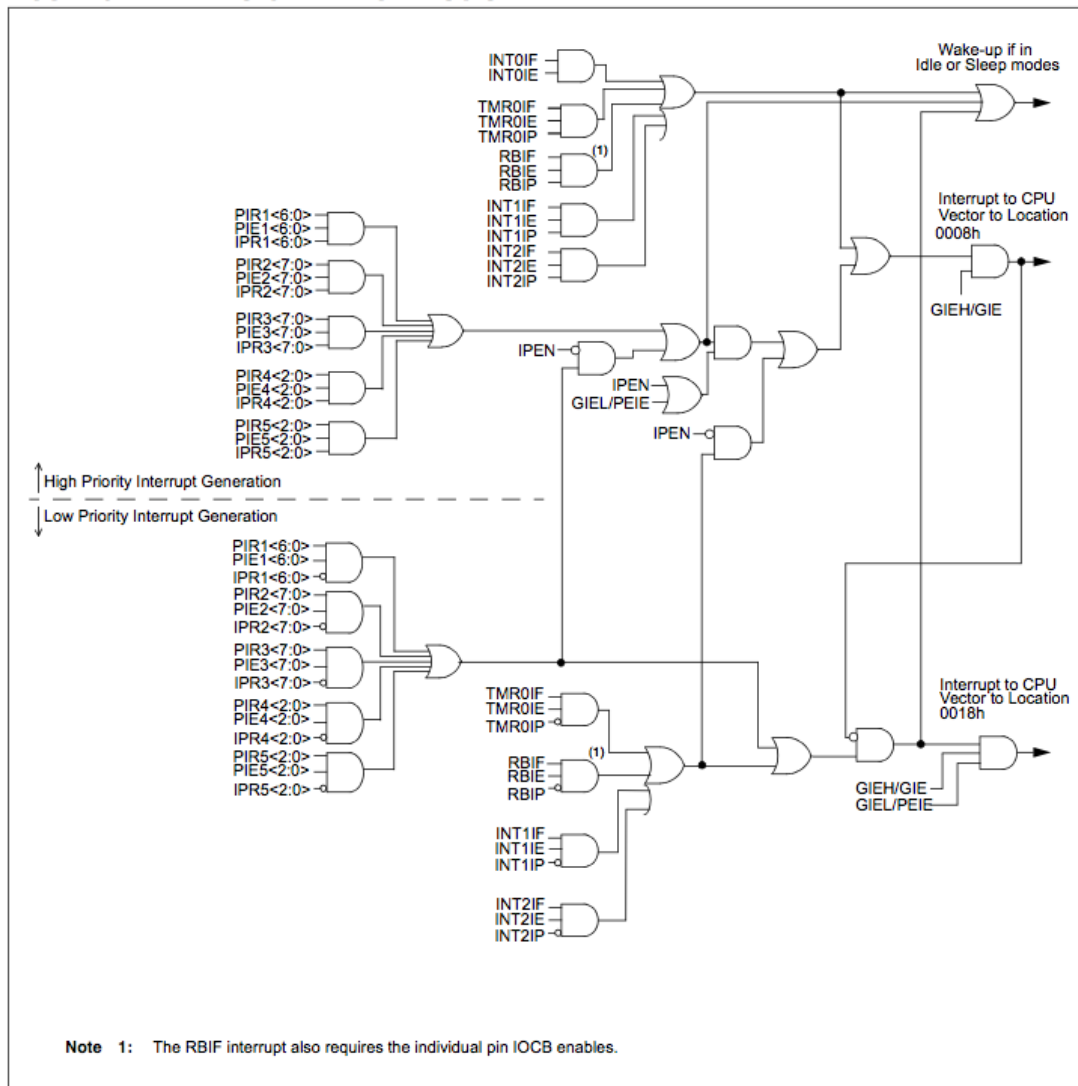
Following picture helps configuring the Interrupt Registers.

GIEH = 1 for global Interrupt enabling.

We use the first Pin for interrupts, so INT0, INTOIE has to be 1.

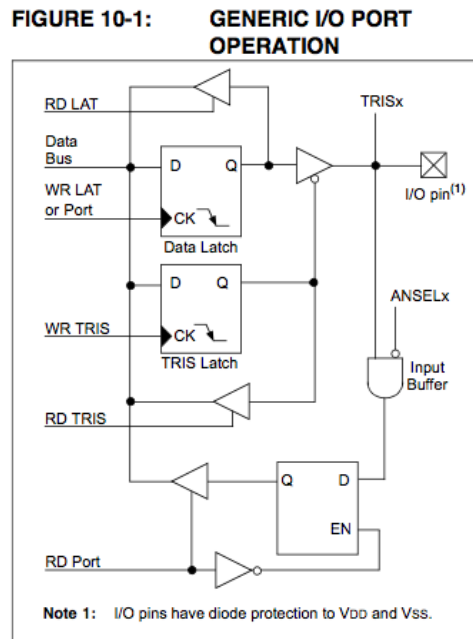
Also for the Timer TMR0IE is 1. For being High Priority with Timer0, TMR0IP also goes 1.

FIGURE 9-1: PIC18 INTERRUPT LOGIC



10.1.1.1.3 Inputs / Outputs

Following Picture helps configure the ANSELx Registers. For getting outputs, ANSELA, will be entirely 0's.



Also since Port B consists of digital inputs, ANSELB will be entirely 0's.

**REGISTER 10-4: ANSELB – PORTB ANALOG SELECT REGISTER**

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
bit 7							bit 0

**Legend:**

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared
		x = Bit is unknown

bit 7-6      **Unimplemented:** Read as '0'

bit 5-0      **ANSB<5:0>:** RB<5:0> Analog Select bit

                  1 = Digital input buffer disabled

                  0 = Digital input buffer enabled

#### 10.1.1.1.4 Code

##### 10.1.1.1.4.1 User.c

This method is basically used to initialize the PIC. IT defines Inputs/Outputs with TRISx and ANSELx, configures the timer with T0CON, and sets the timer elapsing time with TMR0H and TMR0L. Also the Interrupts will be enabled and configured with GIEH/PEIE/INT0IE/INTEDG0. With LATA = 0x00 we make sure that all leds are disabled at the beginning.

**HINT!** Use LATA to define all outputs/inputs at once everytime, instead of PORTA.RAx. Yes you have to reassign all outputs everytime instead of just one, but you will avoid some unwanted behaviour.

```

void InitApp(void)
{
    //Port Declarations
    TRISA = 0b00000000;
    ANSELA = 0b00000000;
    TRISB = 0b11000001;
    ANSELB = 0b00000000;
    // Timer 0
    T0CON = 0b00001000;
    TMR0H = 0x63;
    TMR0L = 0xBF; // 0x63BF = 10ms
    //Interrupts
    GIEH = 1;
    PEIE = 1;
    INT0IE = 1; // Enables the INT0 external interrupt
    INTEDG0 = 1; // Set to interrupt on rising edge
    // Timers & Interrupts
    TMR0IE = 1; // Enables the TMR0 overflow interrupt
    TMR0IP = 1; // Timer0 -> High priority

    LATA = 0x00;
    start = false;
    counter = 0;
}
  
```

##### 10.1.1.1.4.2 System.c

This method is used to set the oscillator frequency. It is common use to just let the oscillator run on his maximal frequency, which in this case is 16 MHz (0b11110010).

```

void ConfigureOscillator(void)
{
    OSCCON = 0b11110010;
    while(!HFIOFS)
    {
        // do nothing until Oscillator is stable
    }
}
  
```

##### 10.1.1.1.4.3 Interrupts.c

This method is essential for the work with timers. Everytime the timer elapses, it will call the interrupt sub routine. This will restart the timer, and also increment the counter wich then can be used to determine the illumination time.

```

void interrupt high_isr(void)
{
    if (INT0IF)
    {
  
```

```

TMR0ON = true; //Enable Timer0
start = true;
INT0IF = false; //clear flag
}

if (TMR0IF)
{
  TMR0H = 0x63;
  TMR0L = 0xBF;
  TMR0IF = false; //clear flag
  counter++;
}
}

```

10.1.1.1.4.4 main.c

This method will actually set the LED states to on or off. Depending on the value of the counter, it will chose the fitting LED. Unfortunately the case parameters have to be static. They are basically  $x \cdot \text{lightTime} + \text{offset}$ . So if a change of the illumination time is desired, every case has to be updated separately. The offset parameter is there to help synchronize the camera and the PIC. It is possible to play around with the offset to get better results. -5 is a good value for an illumination time of 200ms. After 1150ms the cycle is over, and everything will be resetted to be ready for the next measurement.

```

/*****
/* User Global Variable Declaration */
/*****
  static int lightTime = 20; // factor 10ms!
  static int offset = -5; // offset for camera synchronisation

/*****
/* Main Program */
/*****

void main(void)
{
  // init values
  ConfigureOscillator();
  InitApp();
  LATA = 0x04; // let green LED on as default
  //LATA = 0x00; // set all LEDS off
  while(1)
  {
    if(start)
    {
      switch(counter)
      {
        // lightTime + offset
        case(15):
          LATA = 0x01;
          break;
        // 2*lightTime + offset
        case(35):
          LATA = 0x02;
          break;
        // 3*lightTime + offset
        case(55):
          LATA = 0x04;
          break;
        // 4*lightTime + offset
        case(75):

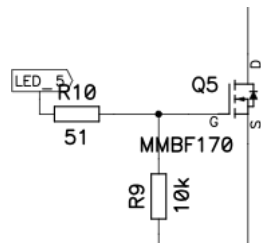
```

```

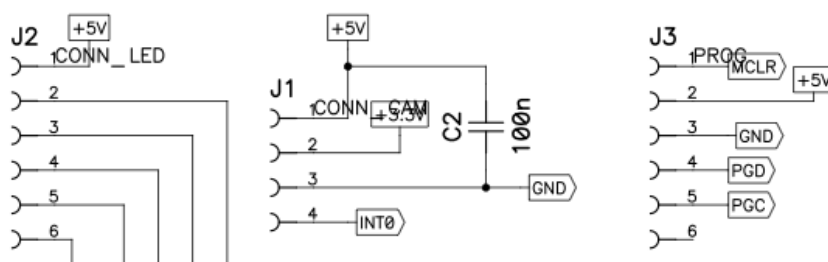
        LATA = 0x08;
        break;
// 5*lightTime + offset
case(95):
    LATA = 0x10;
    break;
// 6*lightTime + offset
case(115):
    //LATA = 0x00;    // turn all LED's off
    LATA = 0x04;    // turn green LED's on
    TMR0ON = false; // stop timer
    start = false;  // reset start flag
    counter = 0;    // reset counter
    break;
    }
    }
}
    
```

### 10.1.1.2 Schematic

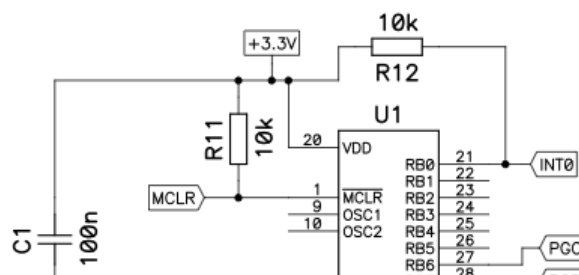
Following page will show the entire schematic of the LED Controller. The core of this PCB clearly is the PIC18 who is controlling the entire illumination process. Since the PIC would not have enough output power, he triggers five MOSFETS, which will then supply the actual LED's. Each of this MOSFETS has two dedicated resistors to enhance their performance:

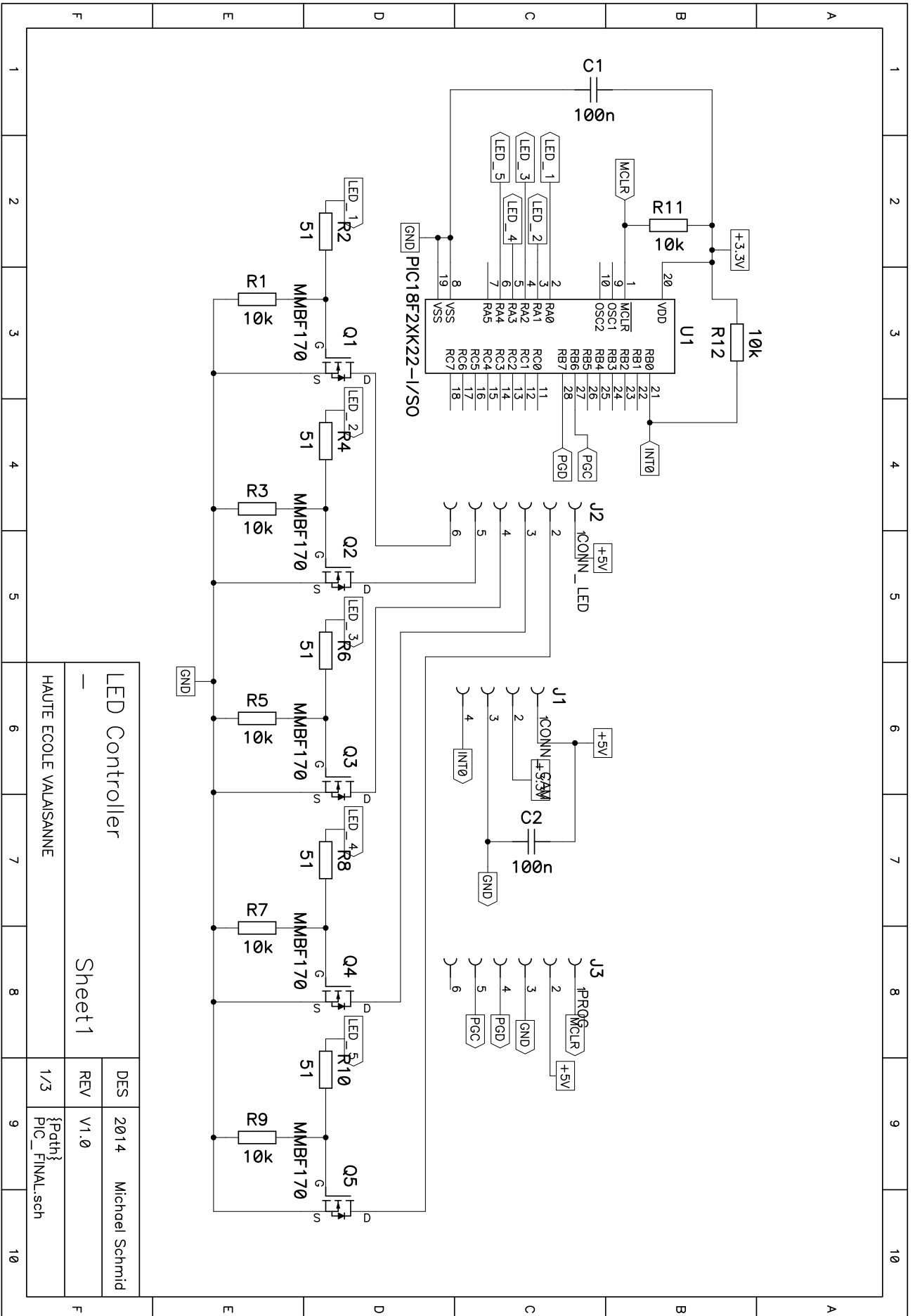


The connectors are from left to right: connection to LEDs, connection to camera, and programming interface.



C1 and R11 are part of the basic circuit needed for the PIC18, while R12 is a Pull-up resistor for the button.



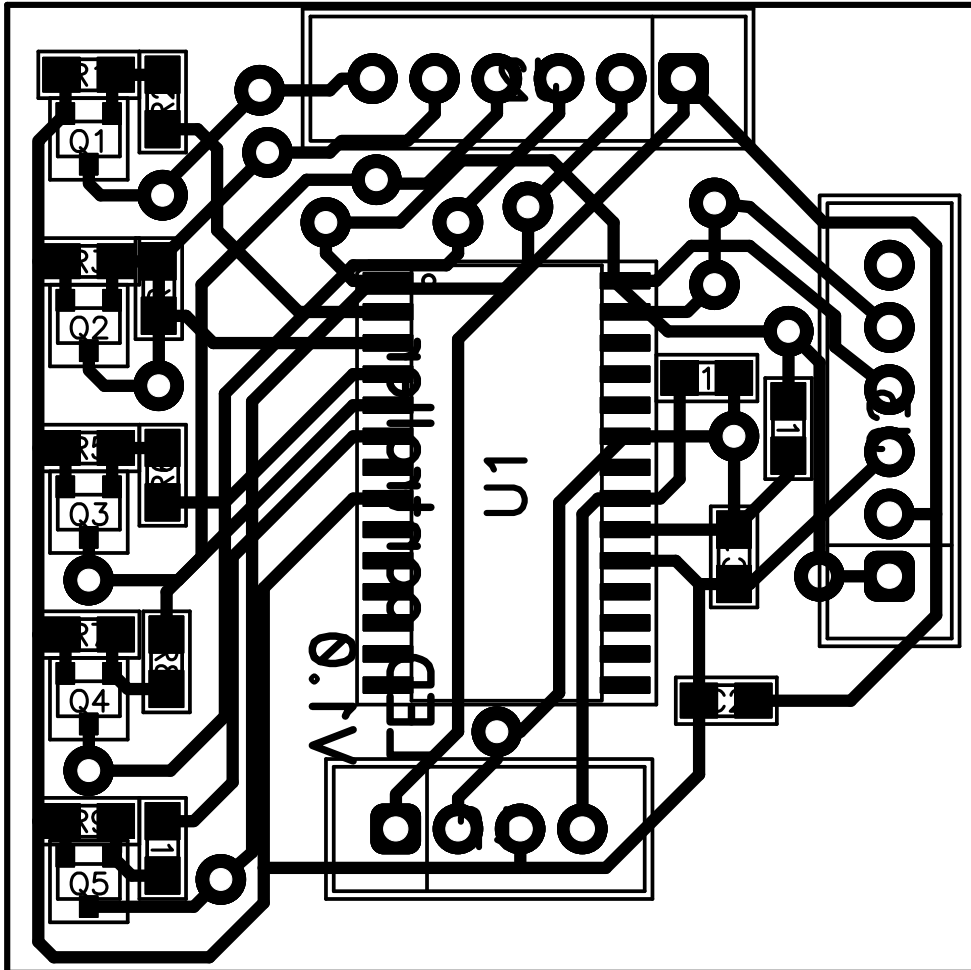


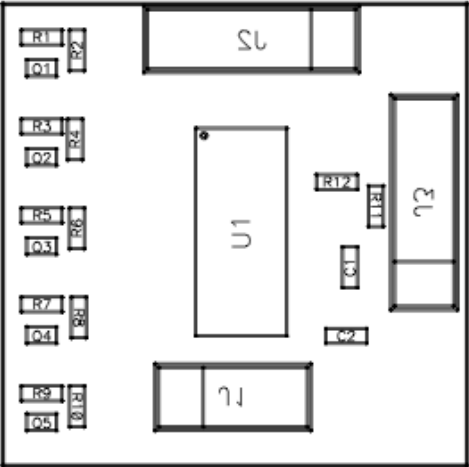
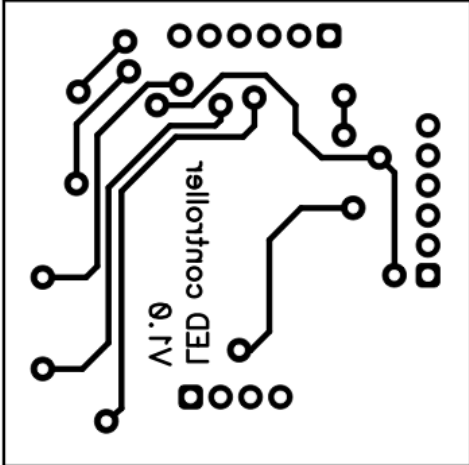
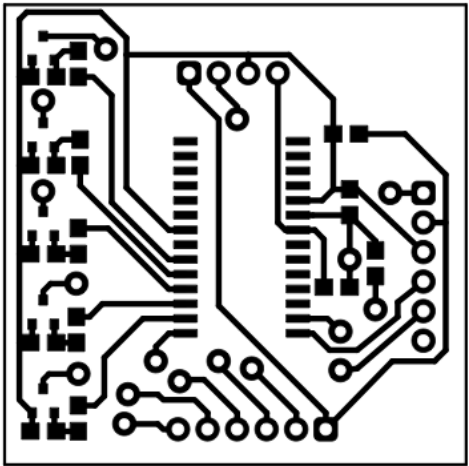

LED Controller		DES	2014	Michael Schmid
—		REV	V1.0	
HAUTE ECOLE VALAISANNE		1/3	{{Path}} PIC_FINAL.sch	
Sheet1				



### 10.1.1.3 Layout

The next page will contain detailed information about the Top and Bottom layer, and also a version with placed components. Following picture shows a overview of all the layers combined.



	0	1	2	3	4	5	6	7	8	9																																		
F	<table border="1"> <tr> <td>1.000</td> <td>X</td> <td>10</td> <td>Yes</td> </tr> <tr> <td>0.000</td> <td>+</td> <td>11</td> <td>Yes</td> </tr> <tr> <td>Hole Dia (mm)</td> <td colspan="2">2mm</td> <td>3mm</td> </tr> <tr> <td colspan="4">D:111_Taple</td> </tr> </table>											1.000	X	10	Yes	0.000	+	11	Yes	Hole Dia (mm)	2mm		3mm	D:111_Taple																				
1.000	X	10	Yes																																									
0.000	+	11	Yes																																									
Hole Dia (mm)	2mm		3mm																																									
D:111_Taple																																												
F	<table border="1"> <tr> <td colspan="11">HAUTE ECOLE VALAIS</td> </tr> <tr> <td colspan="11">FED Coufrollel</td> </tr> </table>											HAUTE ECOLE VALAIS											FED Coufrollel																					
HAUTE ECOLE VALAIS																																												
FED Coufrollel																																												
E	<table border="1"> <tr> <td colspan="11">BIC LIYUΓ bcp BCB\</td> </tr> <tr> <td colspan="11">BEV A1'0</td> </tr> <tr> <td colspan="11">DE2 0\,00'S0,14</td> </tr> </table>											BIC LIYUΓ bcp BCB\											BEV A1'0											DE2 0\,00'S0,14										
BIC LIYUΓ bcp BCB\																																												
BEV A1'0																																												
DE2 0\,00'S0,14																																												
D																																												
C																																												
B																																												
A																																												

## 10.1.2 LED ring

### 10.1.2.1 Schematic

Following page will show the entire LED ring schematic. Each domain consists 4 parallel LED's with their pre-resistors. The calculation of the resistors is shown in following Table.

$$V_{cc} = 5V \qquad R = (V_{cc} - U_{Dx}) / I$$

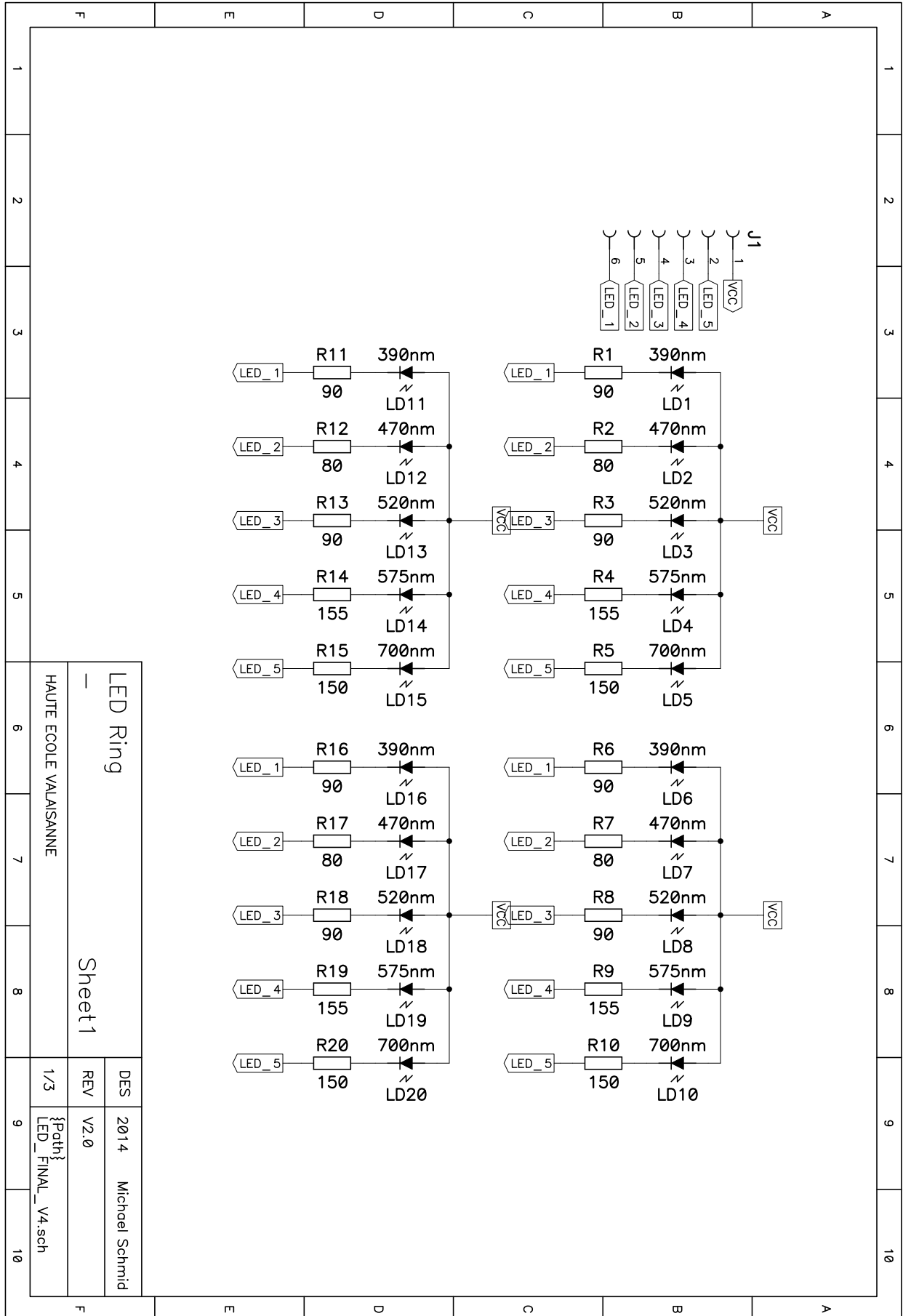
LED1: 390nm  
 UD1 = 3.2V  
 ID1 = 20mA  
 RV1 = 90Ω  
 Appropriate resistor: **91Ω**

LED1: 470nm  
 UD1 = 3.7V (3.4V typ.)  
 ID1 = 20mA  
 RV1 = 65Ω  
 Appropriate resistor: **68Ω**

LED1: 520nm  
 ID1 = 20mA  
 UD1 = 3.2V  
 RV1 = 90Ω  
 Appropriate resistor: **91Ω**

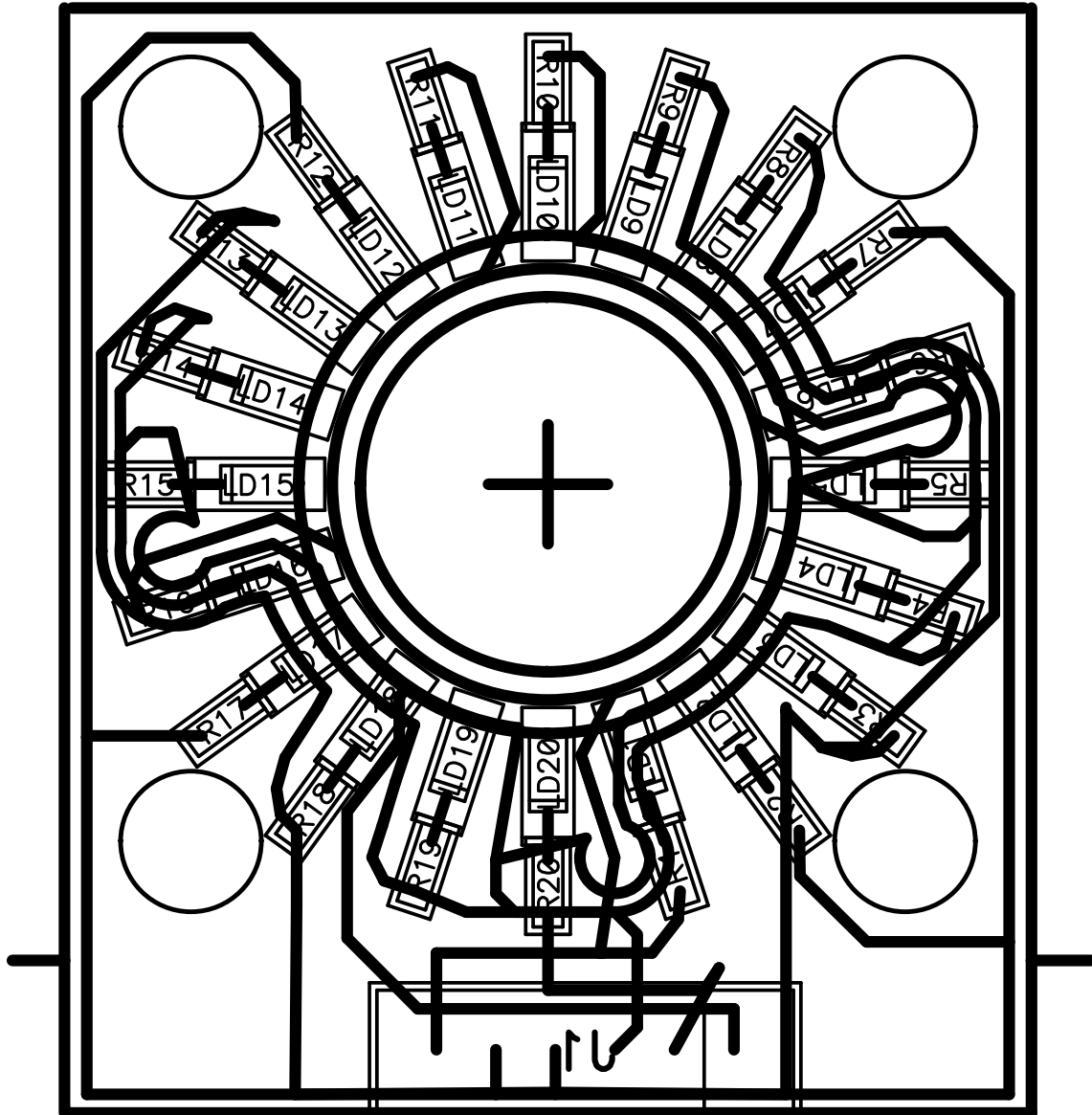
LED1: 610nm  
 ID1 = 20mA  
 UD1 = 2V  
 RV1 = 150Ω  
 Appropriate resistor: **150Ω**

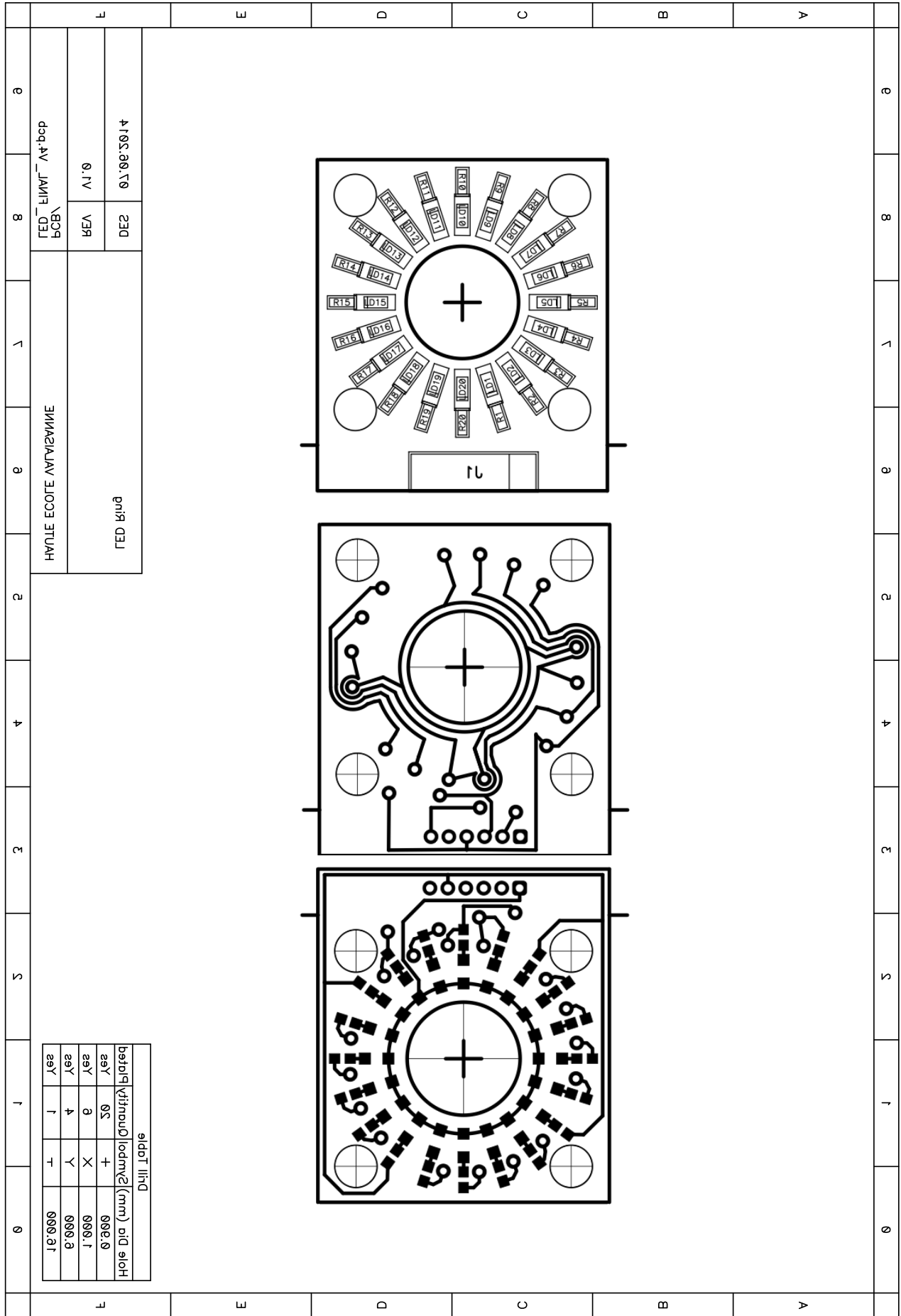
LED1: 880nm  
 ID1 = 20mA  
 UD1 = 1.32V  
 RV1 = 184Ω  
 Appropriate resistor: **180Ω**



### 10.1.2.2 Layout

The next page will contain detailed information about the Top and Bottom layer, and also a version with placed components. Following picture shows a overview of all the layers combined.





### 10.1.3 Parts list

Following Table contains the Part List of both components, the LED ring and the LED Controller.

#### LED's

Quantity	Description	Value	Art-Nr.	vendor	Price
4	SMD LED	390nm	1890331	Farnell	SFr. 12.15
4	SMD LED	470nm	8529965RL	Farnell	SFr. 3.67
4	SMD LED	520nm	1716766RL	Farnell	SFr. 0.42
4	SMD LED	610nm	8530009	Farnell	SFr. 0.30
4	3mm LED	880nm	L-7104SF4BT	Distrelec	SFr. 0.81

#### Resistors

Quantity	Description	Value	Art-Nr.	vendor	Price
4	pre - resistor for 390nm	91Ω			STOCK
4	pre - resistor for 470nm	82Ω			STOCK
4	pre - resistor for 520nm	91Ω			STOCK
4	pre - resistor for 610nm	150Ω			STOCK
4	pre - resistor for 880nm	180Ω			STOCK
7	resistor for PIC/MOSFET	10kΩ			STOCK
5	resistor for MOSFET	51Ω			STOCK

#### Capacitors

Quantity	Description	Value	Art-Nr.	vendor	Price
2	decoupling	100nF			STOCK

#### Transistors

Quantity	Description	Value	Art-Nr.	vendor	Price
5	Mosfet: supply for LED's	MMBF170			STOCK

#### PIC

Quantity	Description	Value	Art-Nr.	vendor	Price
1	PIC18(L)F2X/4XK22	PIC18F2XK22-/SO			STOCK

#### Connectors

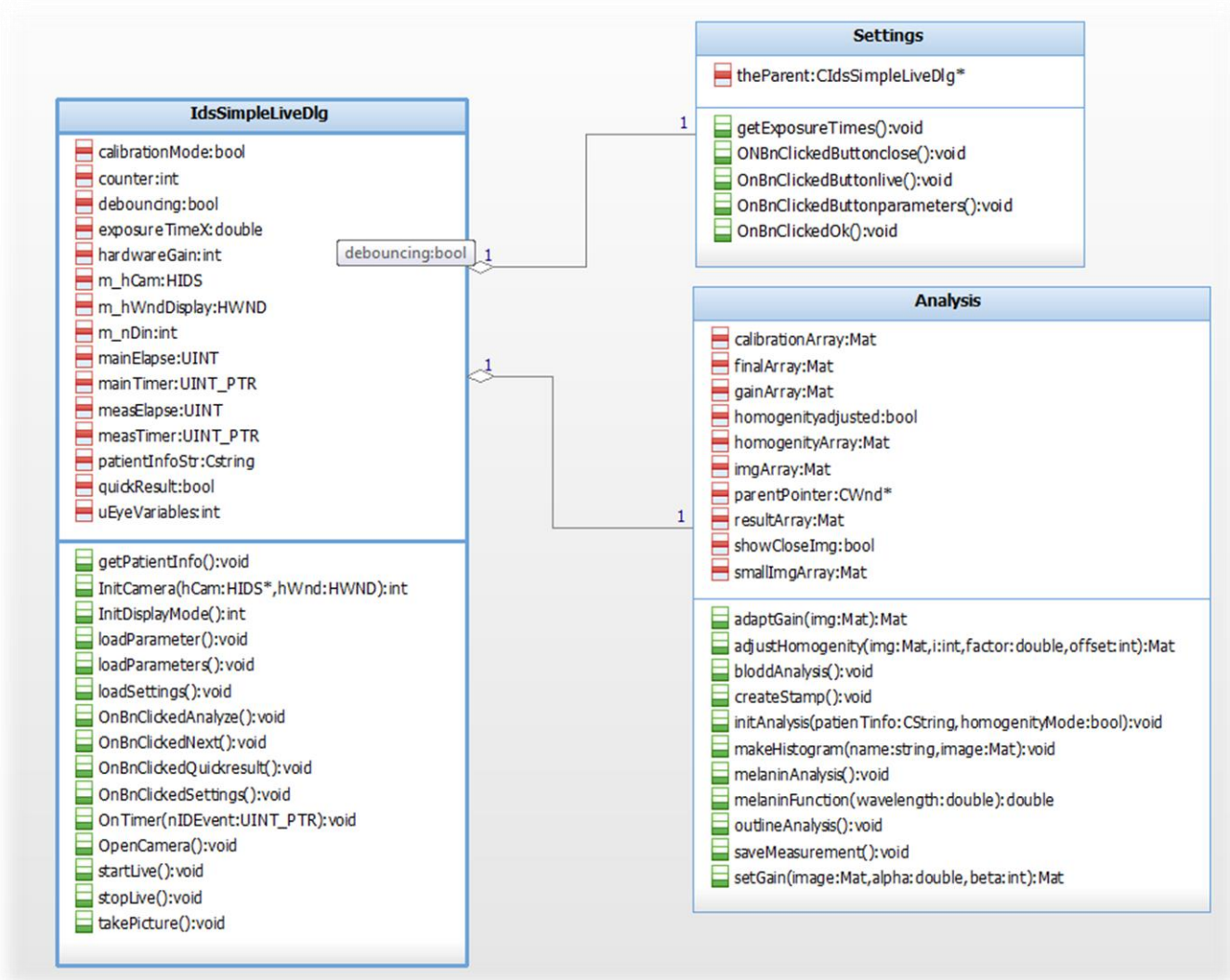
Quantity	Description	Value	Art-Nr.	vendor	Price
1	straight	6 Pins			STOCK
1	90° angle	6 Pins			STOCK
1	90° angle	4 Pins			STOCK

## 10.2 Software - general

This chapter will cover some issues considered to specific for the main document.

### 10.2.1 Class diagram

Fortunately it was possible to implement the software with a simple class constellation. The `IdsSimpleLiveDlg` includes all GUI informations, and also the entire connection to the camera, which makes the image acquirement possible. It includes the huge `uEye.h`, which provides all the comfortable methods from `uEye`. Both other classes are also windows, which contain their own variables and methods. They will be instantiated in the `IdsSimpleLiveDlg` class. Please note that just the most important entries are listed.

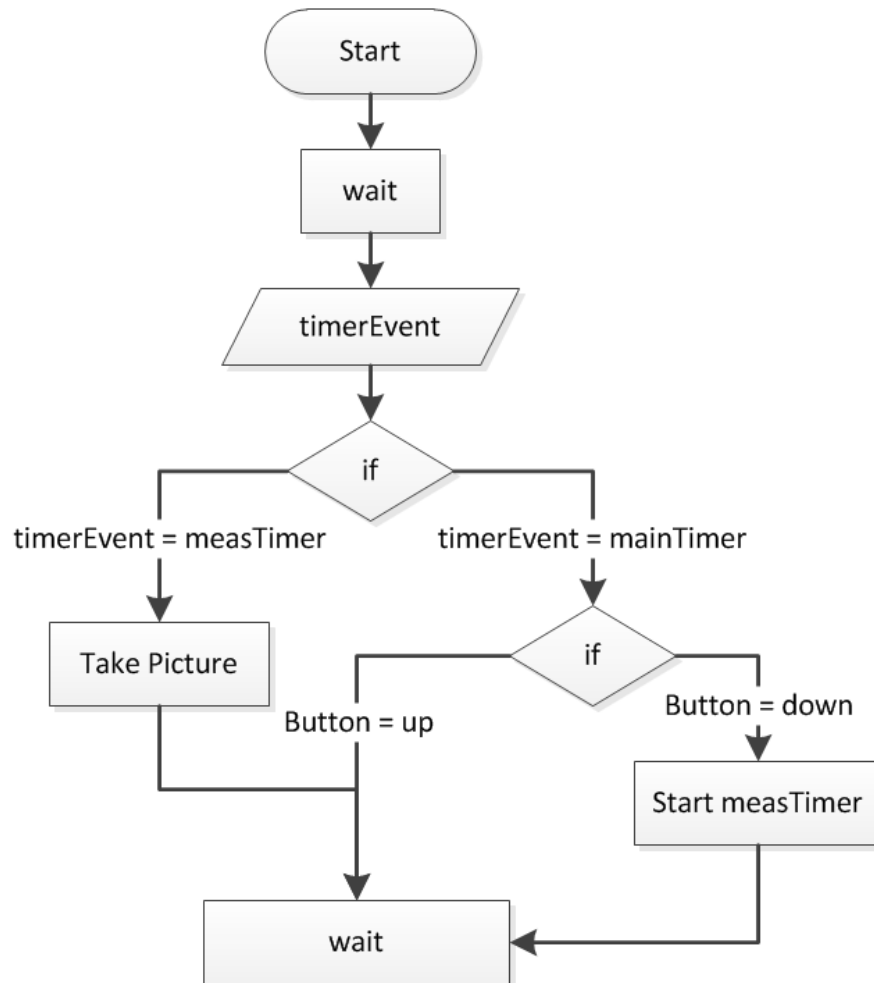


The `IdsSimpleLive` class is not mentioned at all. Since most of the methods need direct access to the GUI, the entire implementation took place in `IdsSimpleLiveDlg`.



## 10.2.2 MainTimer

The main timer plays an essential role in this software. He will be started right at the initialisation of the main window (at the end of the InitDisplayMode method). The timer then will create an event every 10ms. The OnTimer method (*appendix\_10.3.1.2*) will handle that event, and check if the button is triggered. If so, the measTimer will be startet, which will create an event every 200ms. Also this event will be handled from the OnTimer method, which can distinguish the events, and handle it properly. The process is shown in the flowchart below.



The timer also needed to be debounced. Beside the usual debouncing rules for a button, a new trigger is disabled until the measurement is finished. So it is impossible to accidentally start another measurement, while the the current one is still in process.

## 10.3 Software - code

### 10.3.1 IdsSimpleLiveDlg (main window)

Please note that IdsSimpleLive was a given sample project from the uEye company. It contained a mainwindow with frame for the videostream. This example was taken because of the already existing connection between camera and Windows software, to save time. Following methods were already given, and were just slightly changed:

- OnPaint() // draws window
- startLive() // starts live display
- stopLive() // stops live display
- loadParameter() // loads parameter file
- OnButtonExit() // exit the program
- OpenCamera() // start connection to camera
- OnUEyeMessage() // handles message from camera
- ExitCamera() // closes connection to camera
- InitDisplayMode() // initializes the display mode
- InitCamera() // initializes the camera
- GetMaxImageSize() // gets the max supportet image size of camera

Of corse there are also own written methods in this class. The most important ones will be described in this chapter.

#### 10.3.1.1 Initialize dialog

This is the very first method that will be called. When the main windows is created, it will call this method to initialize all his variables. The exposure times now are loaded with the method loadSettings(). Yet the static values are still mentioned for emergency cases.

```

BOOL CIdsSimpleLiveDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // Set the icon for this dialog. The framework does this automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE); // Set big icon
    SetIcon(m_hIcon, FALSE); // Set small icon
    // get handle to display window
    m_hWndDisplay = GetDlgItem( IDC_DISPLAY )->m_hWnd;
    m_pcImageMemory = NULL;
    m_lMemoryId = 0;
    m_hCam = 0;
    m_nRenderMode = IS_RENDER_FIT_TO_WINDOW;
    m_nPosX = 0;
    m_nPosY = 0;
    m_nFlipHor = 0;
    m_nFlipVert = 0;
    //debouncing button
    debouncing = true;
    // Calibration mode Checkbox
    calibrationMode = false;
    // Quick result mode Checkbox
    quickResult = false;
    // open a camera handle
    OpenCamera();
}
    
```

```

    /*
    // Set exposure Times, Max = 34
    exposureTime1 = 15; // exposure time of 390nm LED , 15 recommended
    exposureTime2 = 34; // exposure time of 470nm LED , 34 recommended
    exposureTime3 = 30; // exposure time of 520nm LED , 30 recommended
    exposureTime4 = 34; // exposure time of 610nm LED , 34 recommended
    exposureTime5 = 10; // exposure time of 890nm LED , 12 recommended
    */
    loadSettings();
    return true;
}

```

### 10.3.1.2 Timer

This very important method is the core of the image acquirement. It administrates the two important timers **mainTimer**, and **measTimer**. The mainTimer will check the button state every 10 milliseconds. When the button was triggered, it will start the measTimer, which will take a picture every 200ms. Both values are defined in the IDsSimpleLiveDlg.h file with:

```

#define MAINTIME 10
#define MEASTIME 200

```

```

void CIDsSimpleLiveDlg::OnTimer(UINT_PTR nIDEvent)
{
    CDialog::OnTimer(nIDEvent);
    if( nIDEvent==mainTimer )
    {
        // read the digital (trigger) input
        m_nDin = is_SetExternalTrigger( m_hCam, IS_GET_TRIGGER_STATUS );
        // start measurement if button pushed
        if ( m_nDin!= 0 && debouncing == true)
        {
            debouncing = false;
            counter = 0;
            //Take the first Picture
            takePicture();

            //Timer init & start
            measElapse = MEASTIME;
            measTimer = SetTimer(2,measElapse,NULL);
        }
    }
    if( nIDEvent==measTimer )
    {
        // take the rest of the pictures when timer ticks
        takePicture();
    }
}

```

### 10.3.1.3 Take picture

In this function the camera gets the pictures, and saves them to the desired destinations. First there are quiet a few parameters necessary to enable the image acquisition. Then the system completely relies on the counter. Each time an image was taken, the counter will increment. So each cycle the camera should save the image to a new directory. It is noticeable that there are actually 6 images instead of just 5. This behaviour solves two problems at once. First: If the picture gets taken exactly in the instance the user triggers the button, there is a chance that his movement accidently relocates the device. Therefore we wait 200ms until every motion should be decayed. Secondly the exposure time can only be changed for the after next picture. The next taken picture will still have the initial one. So with depositing the first image into a temporary folder, it helps with both problems, even if it will never be used.

```

void CIdsSimpleLiveDlg::takePicture()
{
    // prepare camera settings to take picture
    IMAGE_FILE_PARAMS ImageFileParams;
    ImageFileParams.pwchFileName = NULL;
    ImageFileParams.pnImageID = NULL;
    ImageFileParams.ppcImageMem = NULL;
    ImageFileParams.nQuality = 0;
    int nRet = NULL;
    ImageFileParams.nFileType = IS_IMG_PNG;
    ImageFileParams.nQuality = 80;

    switch(counter)
    {
    case(0):
        nRet = is_Exposure(m_hCam, IS_EXPOSURE_CMD_SET_EXPOSURE,
            (void*)&exposureTime1, sizeof(exposureTime1));
        ImageFileParams.pwchFileName = L"imgsource\\tempImg.png";
        nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
            sizeof(ImageFileParams));

        counter++;
        break;
    case(1):
        //set the exposure time
        nRet = is_Exposure(m_hCam, IS_EXPOSURE_CMD_SET_EXPOSURE,
            (void*)&exposureTime2, sizeof(exposureTime2));
        // choose fitting folder
        if(calibrationMode)
            ImageFileParams.pwchFileName =
                L"images\\Calibration\\image1_390nm.png";
        else
            ImageFileParams.pwchFileName = L"images\\image1_390nm.png";
        // save image
        nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
            sizeof(ImageFileParams));

        counter++;
        break;
    case(2):
        // similar to case(1)
        nRet = is_Exposure(m_hCam, IS_EXPOSURE_CMD_SET_EXPOSURE,
            (void*)&exposureTime3, sizeof(exposureTime3));
        if(calibrationMode)
            ImageFileParams.pwchFileName =
                L"images\\Calibration\\image2_470nm.png";
        else
            ImageFileParams.pwchFileName = L"images\\image2_470nm.png";
        nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
            sizeof(ImageFileParams));

        counter++;
        break;
    case(3):
        nRet = is_Exposure(m_hCam, IS_EXPOSURE_CMD_SET_EXPOSURE,
            (void*)&exposureTime4, sizeof(exposureTime4));
        if(calibrationMode)
            ImageFileParams.pwchFileName =
                L"images\\Calibration\\image3_520nm.png";
        else
            ImageFileParams.pwchFileName = L"images\\image3_520nm.png";
        nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
            sizeof(ImageFileParams));

        counter++;
        break;
    }
}

```

```

case(4):
    nRet = is_Exposure(m_hCam, IS_EXPOSURE_CMD_SET_EXPOSURE,
        (void*)&exposureTime5, sizeof(exposureTime5));
    if(calibrationMode)
        ImageFileParams.pwchFileName =
            L"images\\Calibration\\image4_610nm.png";
    else
        ImageFileParams.pwchFileName = L"images\\image4_610nm.png";
    nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
        sizeof(ImageFileParams));
    counter++;
    break;
case(5):
    if(calibrationMode)
        ImageFileParams.pwchFileName =
            L"images\\Calibration\\image5_880nm.png";
    else
        ImageFileParams.pwchFileName = L"images\\image5_880nm.png";
    nRet = is_ImageFile(m_hCam, IS_IMAGE_FILE_CMD_SAVE, (void*)&ImageFileParams,
        sizeof(ImageFileParams));
    // stop the measurement timer (since this was the last picture)
    KillTimer(measTimer);
    // reset debouncing
    debouncing = true;
    // reset counter
    counter = 0;
    if(calibrationMode)
    {
        // open an instance of analyze window
        OnBnClickedAnalyze();
        // create the stamp
        analysisWindow->createStamp();
    }
    break;
default:
    // if something goes wrong, this default statement will reset everything.
    KillTimer(measTimer);
    debouncing = true;
    counter = 0;
    break;
}
}

```

### 10.3.1.4 Analyze button

This method will be called when the user clicks the Analyze button. It will create a new instance of the Analysis class. After an error check it can be displayed. At the end it gets the patient informations and initializes the window.

```

void CIdsSimpleLiveDlg::OnBnClickedAnalyze()
{
    // Create new instance of Analysis Window
    analysisWindow = new Analysis(this);
    if(analysisWindow != NULL)
    {
        // Create analysis Window, return error if necessary
        BOOL ret = analysisWindow->Create(IDD_ANALYSIS, this);
        if(!ret)
        {
            AfxMessageBox(_T("Error creating Dialog"));
        }
        // Display Analysis Window
    }
}

```

```

        analysisWindow->ShowWindow(SW_SHOW);
    }
    getPatientInfo();
    analysisWindow->initAnalysis(patientInfoStr, lastnameStr, prenameStr, false);
}

```

### 10.3.1.5 Get patient informations

This rather simple method will take the patient informations out of the textBoxes and combine them to one CString named 'patientInfoStr'. The string then can be accessed from the Analysis Class.

```

void CIdsSimpleLiveDlg::getPatientInfo()
{
    CString streetStr;
    CString cityStr;
    // Get prename from Editbox to String
    CEdit * prenameEdit;
    prenameEdit = reinterpret_cast<CEdit *>(GetDlgItem(IDC_PRENAME));
    prenameEdit->GetWindowTextA(prenamStr);
    // Get last name from Editbox to String
    CEdit * lastnameEdit;
    lastnameEdit = reinterpret_cast<CEdit *>(GetDlgItem(IDC_LASTNAME));
    lastnameEdit->GetWindowTextA(lastnameStr);
    // Get street from Editbox to String
    CEdit * streetEdit;
    streetEdit = reinterpret_cast<CEdit *>(GetDlgItem(IDC_STREET));
    streetEdit->GetWindowTextA(streetStr);
    // Get city from Editbox to String
    CEdit * cityEdit;
    cityEdit = reinterpret_cast<CEdit *>(GetDlgItem(IDC_CITY));
    cityEdit->GetWindowTextA(cityStr);

    // Combine all Strings to final Patient info String
    patientInfoStr = prenameStr + "\r\n" + lastnameStr + "\r\n" + streetStr + "\r\n" +
cityStr + "\r\n";
}

```

### 10.3.1.6 Settings button

Works exactly like OnBnClickedAnalyze(). Opens the Settings Window.

```

void CIdsSimpleLiveDlg::OnBnClickedSettings()
{
    // Create new instance of Settings Window
    settingsWindow = new Settings(this, this);
    if(settingsWindow != NULL)
    {
        // Create analysis Window, return error if necessary
        BOOL ret = settingsWindow->Create(IDD_SETTINGS, this);
        if(!ret)
        {
            AfxMessageBox(_T("Error creating Dialog"));
        }
        // Display Analysis Window
        settingsWindow->ShowWindow(SW_SHOW);
    }
}

```

### 10.3.1.7 Load settings

This method opens the settings textfile in the Parameters folder to get the given exposure times. It will take the first line, and then gets value after value.

**HINT!** The counter has to be absolutely specific about the amount of values to get. Otherwise it would read uninitialized values out of the textfile which would result in unreadable gibberish.

```

void CIdsSimpleLiveDlg::loadSettings()
{
    // open settings textfile
    ifstream inData;
    string fileLine;
    inData.open( "Parameters//settings.txt" );
    int counter = 0;
    // get first line of textfile
    getline( inData, fileLine );
    istringstream iss(fileLine);
    // go through the textfile
    while (iss)
    {
        // get each number...
        double n;
        iss >> n;
        if(counter < 5)
        {
            // ... and assign it to the appropriate exposure time
            switch(counter)
            {
                case(0): exposureTime1 = n;
                    break;
                case(1): exposureTime2 = n;
                    break;
                case(2): exposureTime3 = n;
                    break;
                case(3): exposureTime4 = n;
                    break;
                case(4): exposureTime5 = n;
                    break;
                default:
                    break;
            }
        }
        // the counter is needed to restrict the region. So just the needed numbers
        // are loaded.
        counter++;
    }
    // close textfile
    inData.close();
}
  
```

### 10.3.1.8 Next button

This is the shortcut button for quick measurements without opening the Analysis Window. Since the Analysis is needed to make the optimisations and measurements, this method will create an instance of Analysis, without displaying it. It will then get the patient informations, initialize the window, make a melanin and blood analysis, and finally save the calculated results. At the end it will close the instance. If the user activated the quickResult checkbox, it will display the results of the melanin and blood analysis.

```

void CIdsSimpleLiveDlg::OnBnClickedNext()
{
    // create a new Analysis instance
    analysisWindow = new Analysis(this);
    if(analysisWindow != NULL)
    {
        // Create Analysis Window, return error if necessary
        BOOL ret = analysisWindow->Create(IDD_ANALYSIS, this);
        if(!ret)
        {
            AfxMessageBox(_T("Error creating Dialog"));
        }
        // In this case: dont display Analysis Window
        //analysisWindow->ShowWindow(SW_SHOW);
    }
    getPatientInfo();
    // initialize the analysis
    analysisWindow->initAnalysis(patientInfoStr, lastnameStr, prenameStr, true);
    // make melanin and blood analysis
    analysisWindow->melaninAnalysis();
    // close the windows
    analysisWindow->bloodAnalysis();
    if(!quickResult)
    {
        analysisWindow->OnBnClickedMelaninClose();
        analysisWindow->OnBnClickedBloodClose();
    }
    analysisWindow->OnBnClickedBtnClose();
}
    
```

### 10.3.2 Analysis

The images on their different stages are the core of this Class. For the use with OpenCV they are accessible in [Mat] format. This little overview should help to not confuse them. The images are always stored in an array, where the indexation works like this:

- index[0] = 390nm image
- index[1] = 470nm image
- index[2] = 520nm image
- index[3] = 610nm image
- index[4] = 880nm image
  
- **Mat calibrationArray[5]:** The calibration images are stored in this array. They have nothing to do with the other arrays and work completely independent.
- **Mat imgArray[5]:** This array contains the raw data. The first taken pictures, without any changement.
- **Mat gainArray[5]:** After the gain adjustment, the imgArray will be stoder here.
- **Mat finalArray[5]:** This array contains the gainArray, but without the reference regions on the left and the right, since they are cut away.
- **Mat homogeneityArray[5]:** Contains the finalArray after a homogeneity adjustment.
- **Mat smallImgArray[5]:** Due to performance isusses, the melanin analysis will be made with just a 10th of the actual size. So the smallImgArray contains either the finalArray, or the homogeneityArray with a much smaller size.
- **Mat resultArray[5]:** Contains the calculated results. on index 0 and 1 the melanin results, on index 2 the blood result.



### 10.3.2.1 Init Analysis

This method will be called on the very beginning, when the instance of Analysis is created. First it will get the patient informations, and then all the images are loaded into the imgArray. Next is the gain adaption, and depending on the homogeneityMode it will make a homogeneity adjustment. With the homogeneityadjusted parameter the program knows if it should use the finalArray or the homogeneityArray for further operations.

```

void Analysis::initAnalysis(CString patientInfo, CString lastnameStr, CString prenameStr,
bool homogeneityMode)
{
    // incur patient Informations
    CEdit * display;
    display = reinterpret_cast<CEdit *>(GetDlgItem(IDC_EDIT1));
    display->SetWindowText(patientInfo);
    prename = prenameStr;
    lastname = lastnameStr;
    homogenityadjusted = false;

    // get all images in Array
    imgArray[0] = imread("images\\image1_390nm.png", CV_LOAD_IMAGE_GRAYSCALE);
    imgArray[1] = imread("images\\image2_470nm.png", CV_LOAD_IMAGE_GRAYSCALE);
    imgArray[2] = imread("images\\image3_520nm.png", CV_LOAD_IMAGE_GRAYSCALE);
    imgArray[3] = imread("images\\image4_610nm.png", CV_LOAD_IMAGE_GRAYSCALE);
    imgArray[4] = imread("images\\image5_880nm.png", CV_LOAD_IMAGE_GRAYSCALE);

    // rectangle for calibration region removal
    Rect rec = Rect(imgArray[0].cols/4, 0, (imgArray[0].cols)/2, (imgArray[0].rows));
    for(int i=0; i<5; i++)
    {
        // tweak all gains to optimal setting
        gainArray[i] = adaptGain(imgArray[i]);

        // remove the calibration region
        finalArray[i]=gainArray[i](rec);

        // adjust homogeneity
        if(homogeneityMode)
        {
            homogeneityArray[i] = adjustHomogeneity(finalArray[i], i,
            HOMOGENITYFACTOR, 0);
        }
    }
}

```

### 10.3.2.2 Set gain

The setGain method convinces with its simplicity. It takes an image, and returns the result after it calculated the factor and offset of every pixel.

```

Mat Analysis::setGain(Mat image, double alpha, int beta)
{
    // create new image
    Mat new_image = Mat::zeros(image.size(), image.type());
    // calculate new gain of every single pixel
    for( int y = 0; y < image.rows; y++ )
    {
        for( int x = 0; x < image.cols; x++ )
        {
            // set the new gain
            new_image.at<uchar>(y,x) = saturate_cast<byte>(

```

```

        alpha*( image.at<byte>(y,x) ) + beta );
    }
}
return new_image;
}

```

### 10.3.2.3 Adapt gain

In this method the needed gain will be calculated with help of the reference regions, to ultimately adapt it with the setGain method. The key of this method are the two rectangles Rect: rec1 and rec2, which will take exactly the wanted areas out of the reference regions in the image. For further informations see 5.2.1.2 *Even out the images with gain adaption.*

The limits for the warning message are defined in the Analysis.h file:

```

#define DEVIATIONMIN 0.8
#define DEVIATIONMAX 1.2

```

```

Mat Analysis::adaptGain(Mat img)
{
    double val1 = 0;
    double val2 = 0;
    double avgVal1;
    double avgVal2;
    double avgValTotal;
    double neededGain;
    Mat calibrationImg = img;
    // get roi (region of interest)
    Rect rec1 = Rect((img.cols)/8+10, (img.rows)/3, (img.cols)/10, (img.rows)/5);
    Mat roi1(img, rec1);
    for( int y = 0; y < (roi1.rows/2); y++ )
    {
        for( int x = 0; x < roi1.cols; x++ )
        {
            val1 += roi1.at<uchar>(x,y);
        }
    }
    //calculate the avarage value of the region
    avgVal1 = val1/(roi1.cols*(roi1.rows/2));
    // draw first calibration region rectangle
    rectangle(calibrationImg, rec1, Scalar( 0, 255, 0 ), 1, 8, 0 );
    //get second roi (region of interest)
    Rect rec2 = Rect((img.cols)*6/8+20, (img.rows)/3+20, (img.cols)/10, (img.rows)/5);
    Mat roi2(img, rec2);
    for( int y = 0; y < (roi2.rows/2); y++ )
    {
        for( int x = 0; x < roi2.cols; x++ )
        {
            val2 += roi2.at<uchar>(x,y);
        }
    }
    // calculate the avarage value of the second region
    avgVal2 = val2/(roi2.cols*(roi2.rows/2));
    // draw second calibration region rectangle
    rectangle(calibrationImg, rec2, Scalar( 0, 255, 0 ), 1, 8, 0 );
    // give out warning if values vary too much
    if(avgVal1/avgVal2 < DEVIATIONMIN || avgVal1/avgVal2 > DEVIATIONMAX)
    {
        AfxMessageBox(_T("Uneven lighting! \nPlease ensure the LED's work properly,
        and no external light is disturbing the measurement!", MB_ICONEXCLAMATION));
    }
    // calculate the result
}

```

```

    avgValTotal = (avgVal1 + avgVal2)/2;
    // calculate the needed gain to bring the reference region into saturation
    neededGain = 255/avgValTotal;
    //get new gain for the image
    Mat resultImage = setGain(img, neededGain, 0);

    //imshow("img", calibrationImg);
    return resultImage;
}

```

### 10.3.2.4 Create stamp

This method will create the stamps, which includes the needed correction values for a homogeneity adjustment. Each image will get its own textfile stamp, saved in the Parameters/stampX.txt. Please notice that also in this case only a rectangle Rect without the reference regions is used. For more information about the concept see *5.2.1.4 Create Stamp*.

```

void Analysis::createStamp()
{
    char tempString[40];
    string stampDirectory;
    string imgDirectory;
    // calculate the size of the image
    double size = finalArray[0].rows*finalArray[0].cols;
    // sum and mean for every single image, initialized with 0
    double sum[5] = {0,0,0,0,0};
    double mean[5] = {0,0,0,0,0};
    double tempDouble = 0;
    // define the rectangle. Only this rectangle will be used for the adjustment!
    Rect rec = Rect(imgArray[0].cols/4, 0, (imgArray[0].cols)/2, (imgArray[0].rows));
    // go through all 5 images...
    for(int i = 0; i < 5; i++)
    {
        // chose fitting image and stamp
        switch(i)
        {
            case 0 :
                stampDirectory = "Parameters\\stamp390.txt";
                imgDirectory = "images\\Calibration\\image1_390nm.png";
                break;
            case 1 :
                stampDirectory = "Parameters\\stamp470.txt";
                imgDirectory = "images\\Calibration\\image2_470nm.png";
                break;
            case 2 :
                stampDirectory = "Parameters\\stamp520.txt";
                imgDirectory = "images\\Calibration\\image3_520nm.png";
                break;
            case 3 :
                stampDirectory = "Parameters\\stamp610.txt";
                imgDirectory = "images\\Calibration\\image4_610nm.png";
                break;
            case 4 :
                stampDirectory = "Parameters\\stamp880.txt";
                imgDirectory = "images\\Calibration\\image5_880nm.png";
                break;
            default:
                break;
        }
        //load image
        calibrationArray[i] = imread(imgDirectory, CV_LOAD_IMAGE_GRAYSCALE);
        // take the rectangle out of the image
    }
}

```

```

        calibrationArray[i]=calibrationArray[i](rec);
        //Calculate the sum, preparing for average value (mean)
        for( int y = 0; y < calibrationArray[0].rows; y++ )
        {
            for( int x = 0; x < calibrationArray[0].cols; x++ )
            {
                tempDouble = calibrationArray[i].at<byte>(y,x);
                sum[i]+= tempDouble;
            }
        }
        // start the stream (open textfile)
        ofstream fout(stampDirectory);
        // calculate average value (mean)
        mean[i] = sum[i]/size;
        // write value of each pixel into textfile
        for( int y = 0; y < calibrationArray[0].rows; y++ )
        {
            for( int x = 0; x < calibrationArray[0].cols; x++ )
            {
                // correcture = value / mean
                tempDouble = (calibrationArray[i].at<byte>(y,x))/mean[i];
                sprintf(tempString, "%.2f ", tempDouble);
                fout << tempString;
            }
            fout <<endl;
        }
        fout.close();
    }
}

```

### 10.3.2.5 Adjust homogeneity

The stamp textfile will be loaded, and used for the homogeneity adjustment. It is again solved with the ifstream, that reads line by line from the textfile. while(iss) means he reads until he reaches the last line. Again the counter is really important, so only intended values will be processed. (See 5.2.1.5 *Adjust the homogeneity*)

```

Mat Analysis::adjustHomogeneity(Mat img, int i, double factor, int offset)
{
    // create a new empty image, filled with zeros
    Mat result = Mat::zeros(img.size(), img.type());
    // open textfile
    ifstream inData;
    string fileLine;
    // open the fitting stamp file
    switch(i)
    {
        case(0): inData.open( "Parameters//stamp390.txt" );
                break;
        case(1): inData.open( "Parameters//stamp470.txt" );
                break;
        case(2): inData.open( "Parameters//stamp520.txt" );
                break;
        case(3): inData.open( "Parameters//stamp610.txt" );
                break;
        case(4): inData.open( "Parameters//stamp880.txt" );
                break;
        default:
                break;
    }
    // do it for each row of the textfile...
    for(int y = 0; y < result.rows; y++)

```

```

{
    double counter = 0;
    // get a line from textfile
    getline( inData, fileLine );
    istringstream iss(fileLine);
    // go through the line, and get a value
    while (iss)
    {
        double correction;
        iss >> correction;
        // get values, until the line is finished
        if(counter < result.cols)
        {
            double tempval;
            tempval = img.at<byte>(y,counter);
            // calculate the new value, with value/correction ... there is a
            *factor correcture to avoid saturation
            // factors around 0.8 seem to work pretty well
            result.at<byte>(y,counter) = (tempval/correction*factor +
            offset);
        }
        // the counter is used to stop reading when the line is finished!
        Otherwise it would read empty bytes
        counter++;
    }
}
// close the textfile
inData.close();
// sets fag, so the program knows now there are homogeny adjusted images aviable
homogenytyadjusted = true;
// reconstruct initial image
return result;
}
    
```

### 10.3.2.6 Melanin analysis

This is the most important and sophisticated class. Please read *5.2.2.1 Melanin analysis* carefully, before you start working with this class. Since the calculated factor is a double, the factorArray is needed. For convenient calculations also the byte Array melaninArray is used. The analysis takes a 10 times narrowed image due to performance issues. Feel free to experiment with Size sizeSmall() for better resolution, but mind the time it will take. The first loop just resizes the image. The second loop creates the 5 images with the ratio between measurement and melanin permeability function. The next two loops will calculate the mean and the standard deviation Image.

```

void Analysis::melaninAnalysis()
{
    int counter = 0;
    // resize the images by factor 10
    byte melaninArray[5][1776];
    double factorArray[5][1776];
    Mat factorImage[5];
    // define the small size
    Size sizeSmall(finalArray[0].cols/10,finalArray[0].rows/10);
    for(int i=0; i<5; i++)
    {
        // init factoreImage Mat
        if(homogenytyadjusted)
        {
            // create small image array
            resize(homogenyArray[i],smallImgArray[i],sizeSmall);
        }
    }
}
    
```

```

    }
    else
    {
        // create small image array
        resize(finalArray[i],smallImgArray[i],sizeSmall);
    }
    factorImage[i] = Mat::zeros(smallImgArray[0].size(), smallImgArray[0].type());
}
int rows = smallImgArray[0].rows;
int cols = smallImgArray[0].cols;
for( int y = 0; y < rows; y++ )
{
    for( int x = 0; x < cols; x++ )
    {
        for(int i=0; i<5; i++)
        {
            // put value into melaninArray
            melaninArray[i][counter] = smallImgArray[i].at<byte>(y,x);
            //calculate factor between melanin absorption function and actual
            pictures
            // then show it in images. white = good fit, black = bad fit.
            switch(i)
            {
                case 0:
                    factorArray[i][counter] = melaninArray[i][counter] /
                        melaninFunction(390);
                    factorImage[i].at<byte>(y,x) =
                        factorArray[i][counter]*128;
                    break;
                case 1:
                    factorArray[i][counter] = melaninArray[i][counter] /
                        melaninFunction(470);
                    factorImage[i].at<byte>(y,x) =
                        factorArray[i][counter]*128;
                    break;
                case 2:
                    factorArray[i][counter] = melaninArray[i][counter] /
                        melaninFunction(520);
                    factorImage[i].at<byte>(y,x) =
                        factorArray[i][counter]*128;
                    break;
                case 3:
                    factorArray[i][counter] = melaninArray[i][counter] /
                        melaninFunction(610);
                    factorImage[i].at<byte>(y,x) =
                        factorArray[i][counter]*128;
                    break;
                case 4:
                    factorArray[i][counter] = melaninArray[i][counter] /
                        melaninFunction(880);
                    factorImage[i].at<byte>(y,x) =
                        factorArray[i][counter]*128;
                    break;
                default:
                    break;
            }
        }
        counter++;
    }
}

// calculate the mean value
Mat melaninMean = Mat::zeros(smallImgArray[0].size(), smallImgArray[0].type());

```

```

for( int y = 0; y < rows; y++ )
{
    for( int x = 0; x < cols; x++ )
    {
        double tempvalue = 0;
        for(int i=0; i<5; i++)
        {
            tempvalue+= factorImage[i].at<byte>(y,x);
        }
        melaninMean.at<byte>(y,x) = tempvalue/5;
    }
}
// resize, show and save image to results
Size size(finalArray[0].cols,finalArray[0].rows);
resize(melaninMean,melaninMean,size);
imshow("melaninMean", melaninMean);
resultArray[0] = melaninMean;

// calculate the standard deviation value
Mat melaninSD = Mat::zeros(smallImgArray[0].size(), smallImgArray[0].type());
double arraySD[47][37];
for( int y = 0; y < rows; y++ )
{
    for( int x = 0; x < cols; x++ )
    {
        double tempvalue = 0;
        for(int i=0; i<5; i++)
        {
            double sum = smallImgArray[i].at<byte>(y,x) -
                melaninMean.at<byte>(y,x);
            tempvalue+= pow(sum, 2);
        }

        double StandardDeviation = sqrt(tempvalue/5);
        //arraySD[y][x] = sqrt(tempvalue/5);
        melaninSD.at<byte>(y,x) = sqrt(tempvalue/5);
    }
}
// resize, show and save image to results
resize(melaninSD,melaninSD,size);
imshow("melaninSD", melaninSD);
resultArray[1] = melaninSD;
}

```

### 10.3.2.7 Melanin Function

This method will return the permeability value in function of the wavelength.

```

double Analysis::melaninFunction(double wavelength)
{
    double polynom1 = -    0.000000024257 * pow(wavelength, 4);
    double polynom2 = +    0.000054677      * pow(wavelength, 3);
    double polynom3 = -    0.046066         * pow(wavelength, 2);
    double polynom4 = +    17.699           * wavelength;
    double polynom5 = -    2525.4;
    double result = polynom1 + polynom2 + polynom3 + polynom4 + polynom5;
    return result;
}

```

### 10.3.2.8 Save measurements

Usually this method will be called when the user closes the Analysis window with "Save & Close". First the current time has to be defined in a string. Then after creating all the folders, the different images will be saved into their designated directory.

```

void Analysis::saveMeasurement()
{
    // get time and date
    struct tm *newtime;
    time_t long_time;
    time( &long_time ); /* Get time as long integer. */
    newtime = localtime( &long_time ); /* Convert to local time. */
    char timeStr[80];
    sprintf(timeStr, "%i-%i-%02i_%02i-%02i-%02i", (newtime->tm_year+1900), newtime->tm_mon,
    newtime->tm_mday, newtime->tm_hour, newtime->tm_min, newtime->tm_sec);
    CString theTime = timeStr;
    // Create define directories
    CString directory = "Measurements\\";
    CString patientFolder = directory + prename + (" ") + lastname;
    CString measFolder = patientFolder + ("\\") + theTime;
    CString rawDataFolder = measFolder + ("\\raw_data");
    CString optimizedFolder = measFolder + ("\\optimized");

    // Create the folders
    CreateDirectory(patientFolder,NULL);
    CreateDirectory(measFolder,NULL);
    CreateDirectory(rawDataFolder,NULL);
    CreateDirectory(optimizedFolder,NULL);

    // Save raw data
    string rawimg1 = rawDataFolder + ("\\image1_390nm.png");
    string rawimg2 = rawDataFolder + ("\\image2_470nm.png");
    string rawimg3 = rawDataFolder + ("\\image3_520nm.png");
    string rawimg4 = rawDataFolder + ("\\image4_610nm.png");
    string rawimg5 = rawDataFolder + ("\\image5_880nm.png");
    imwrite( rawimg1 , imgArray[0]);
    imwrite( rawimg2 , imgArray[1]);
    imwrite( rawimg3 , imgArray[2]);
    imwrite( rawimg4 , imgArray[3]);
    imwrite( rawimg5 , imgArray[4]);

    // Save the optimized images
    string optimg1 = optimizedFolder + ("\\390nm.png");
    string optimg2 = optimizedFolder + ("\\470nm.png");
    string optimg3 = optimizedFolder + ("\\520nm.png");
    string optimg4 = optimizedFolder + ("\\610nm.png");
    string optimg5 = optimizedFolder + ("\\880nm.png");
    imwrite( optimg1 , finalArray[0]);
    imwrite( optimg2 , finalArray[1]);
    imwrite( optimg3 , finalArray[2]);
    imwrite( optimg4 , finalArray[3]);
    imwrite( optimg5 , finalArray[4]);

    // save the result images
    string resultimg1 = measFolder + ("\\melanin_mean.png");
    string resultimg2 = measFolder + ("\\melanin_SD.png");
    string resultimg3 = measFolder + ("\\blood.png");
    imwrite( resultimg1 , resultArray[0]);
    imwrite( resultimg2 , resultArray[1]);
    imwrite( resultimg3 , resultArray[2]);
}
  
```

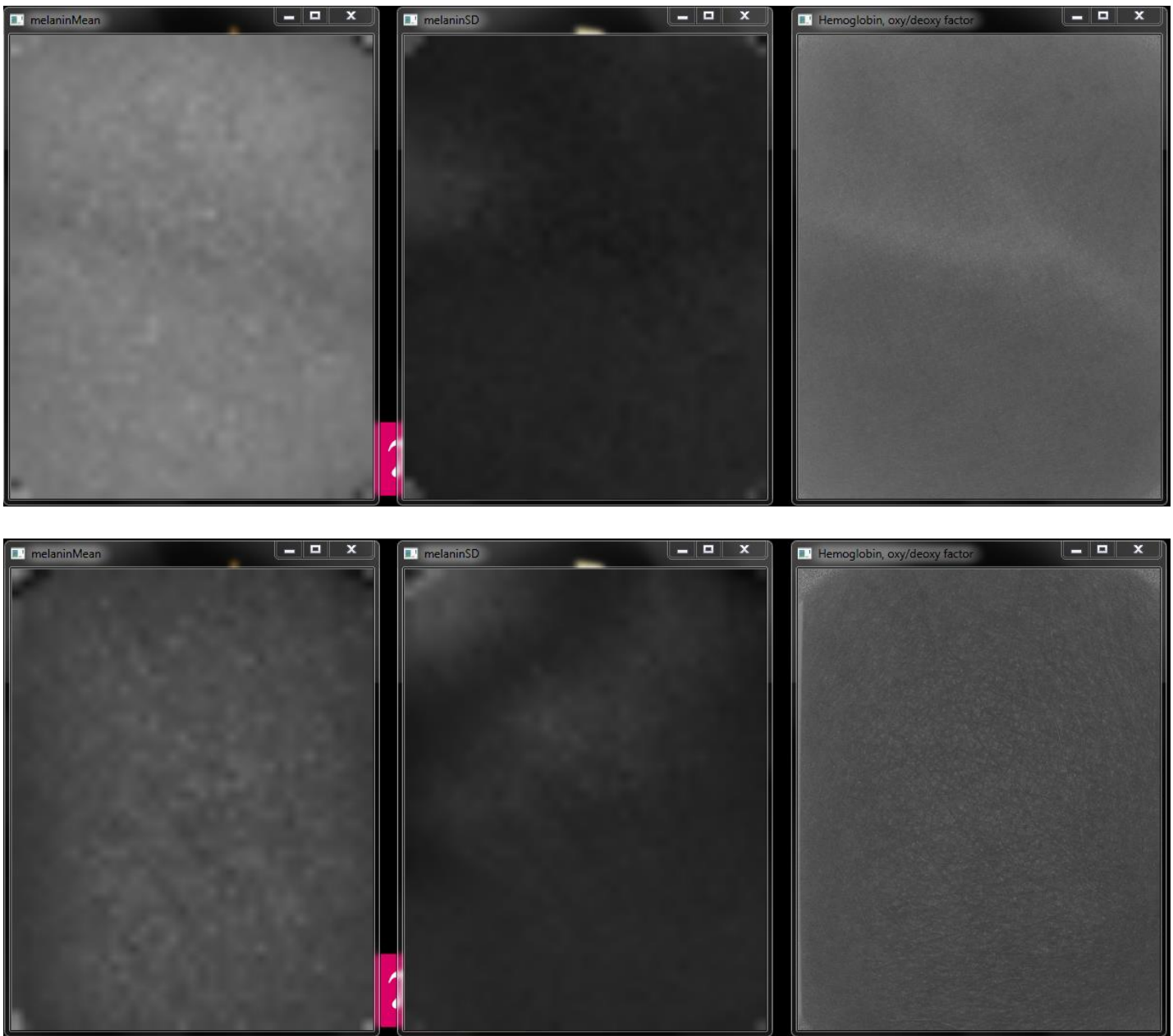


## 10.4 Tests

### 10.4.1 Melanin and hemoglobin tests

Following pictures will compare measurements from a rather brown, and a rather white skin. The melaninMean clearly shows a noticeable difference. The white skin has a much brighter result. Also an interesting observation of the first example is the vein, that was already detected from the hemoglobin test. Even without a specific calibration.

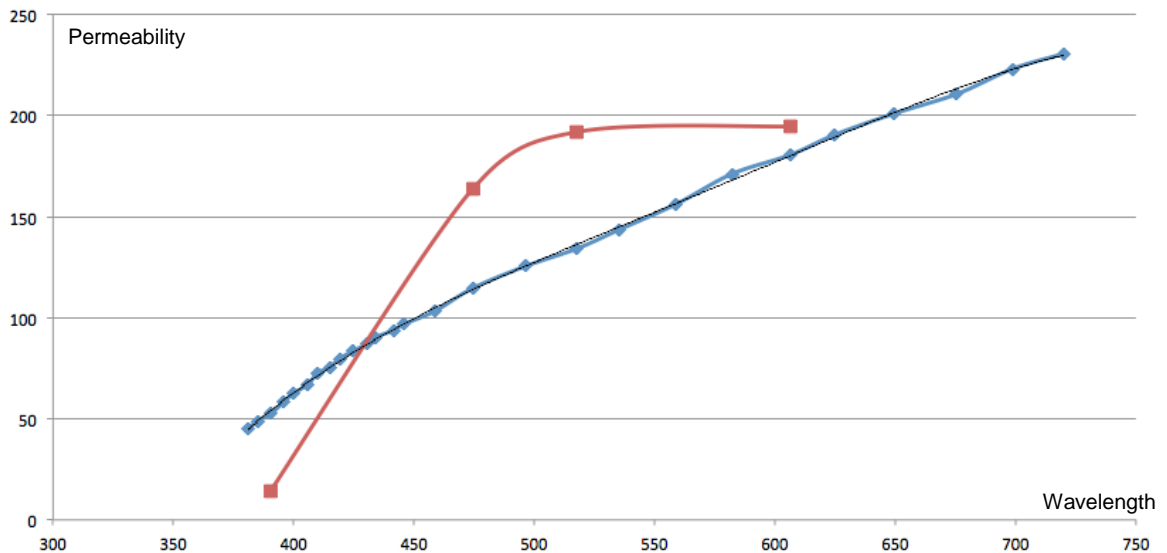
**HINT!** Feel free to explore the Measurements folder of the software, it already contains quiet a few example measurements from different volunteers.



## 10.4.2 Prove of melanin detection

### 10.4.2.1 Test 1: "skin colored" paper

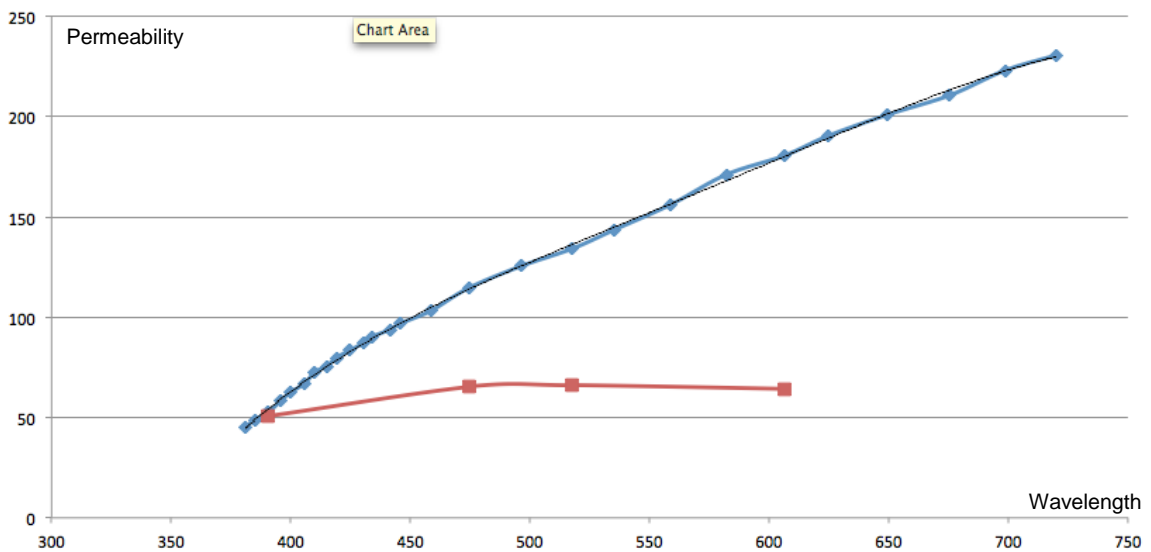
Does the Device really detect melanin? Or can it be tricked with a skin-similar surface? Following graph shows the melanin permeability curve, and the measurement of the paper.



The graph shows that it is obviously a very bad match. Even a calibration factor would change nothing. We just proven: There is probably no melanin in paper.

### 10.4.2.2 Test 2: Black plastic

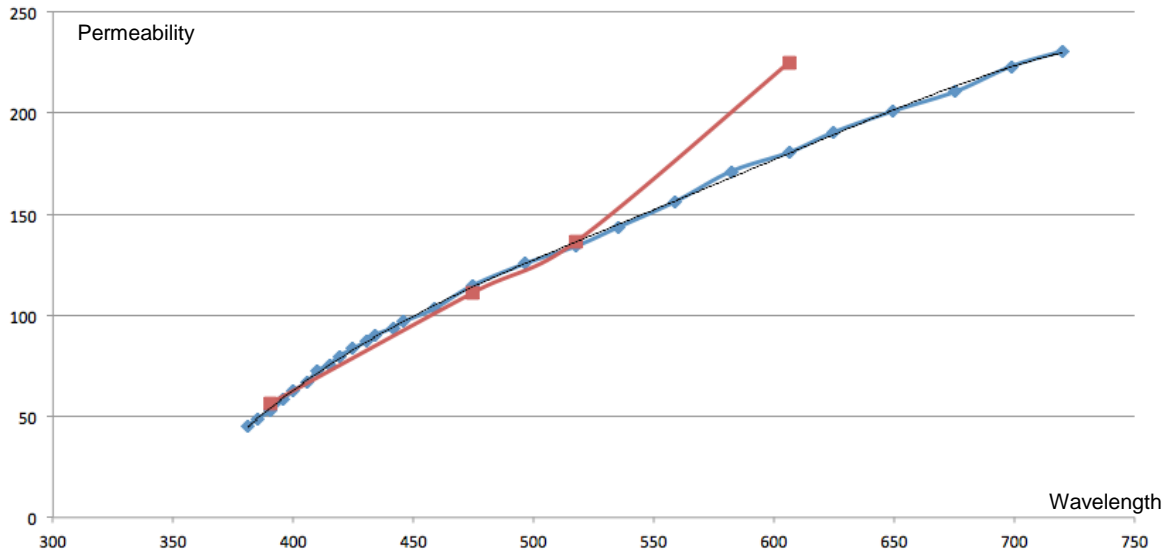
One thing is for sure: The more melanin, the darker the skin. Is it possible to simulate a malignant melanoma with black plastic?



Even after a tremendous boost of 300% the measurement hardly comes close to the desired result. The black plastic just absorbs all the light.

### 10.4.2.3 Test 3: actual skin

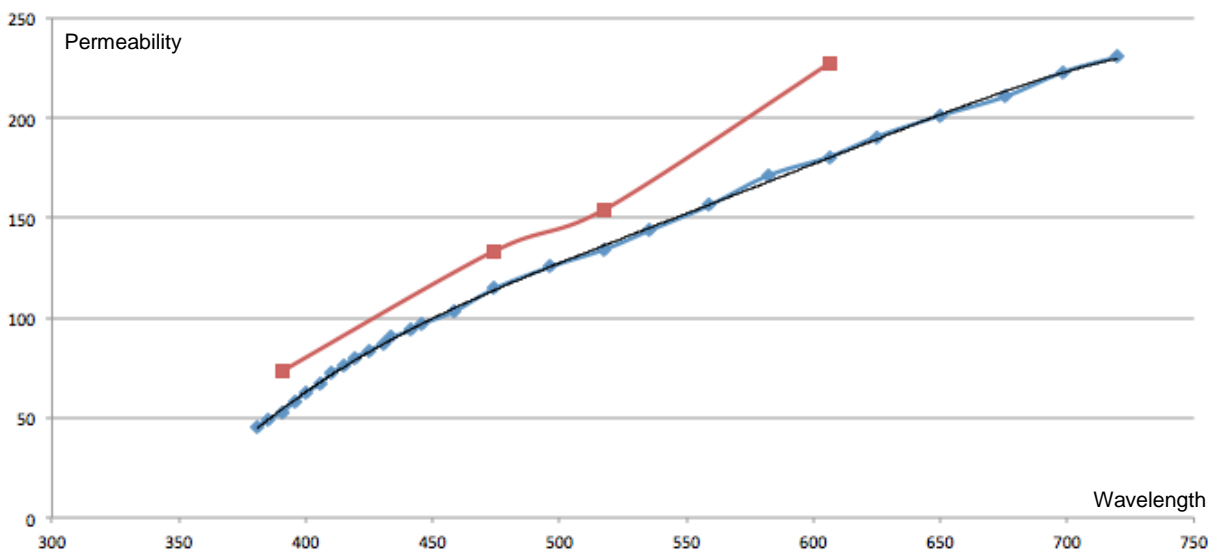
Since i captured some sunbeams lately, there should be a little melanin available. So i considered myself legitimate as a test person. As is generally known, a picture says more than a thousand words.



Please consider that the 610nm value initially wasn't even planned to participate in the melanine measurement, since it has such a small impact. A gain of 2.0 was used.

### 10.4.2.4 Test 4: second skin test

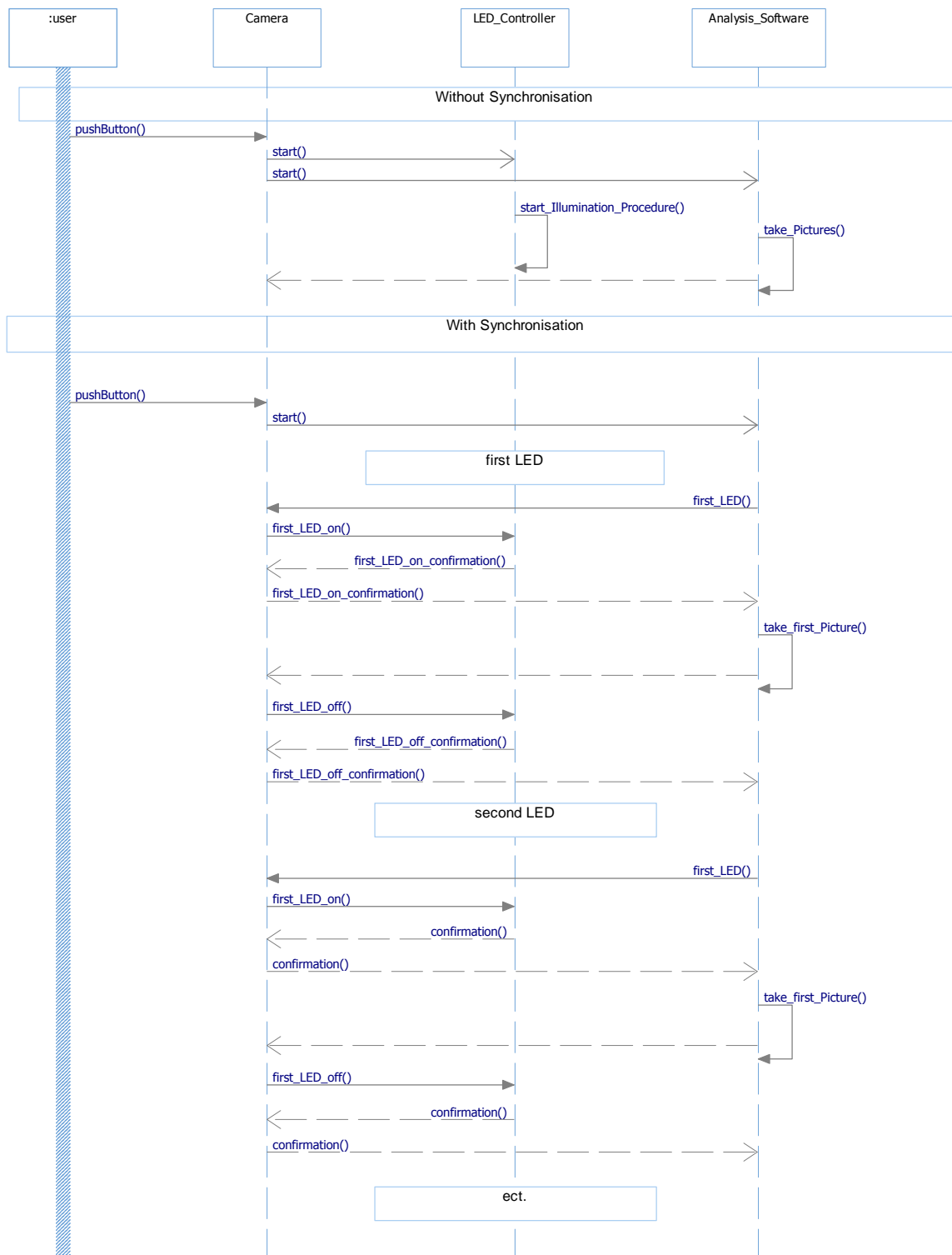
Of course the first skintest could be just a lucky hit. So a second measurement, this time a brighter sample, with probably less melanin took place. To show the melanin difference of different tanning, the same gain of the first test was used.



## 10.5 Future extensions

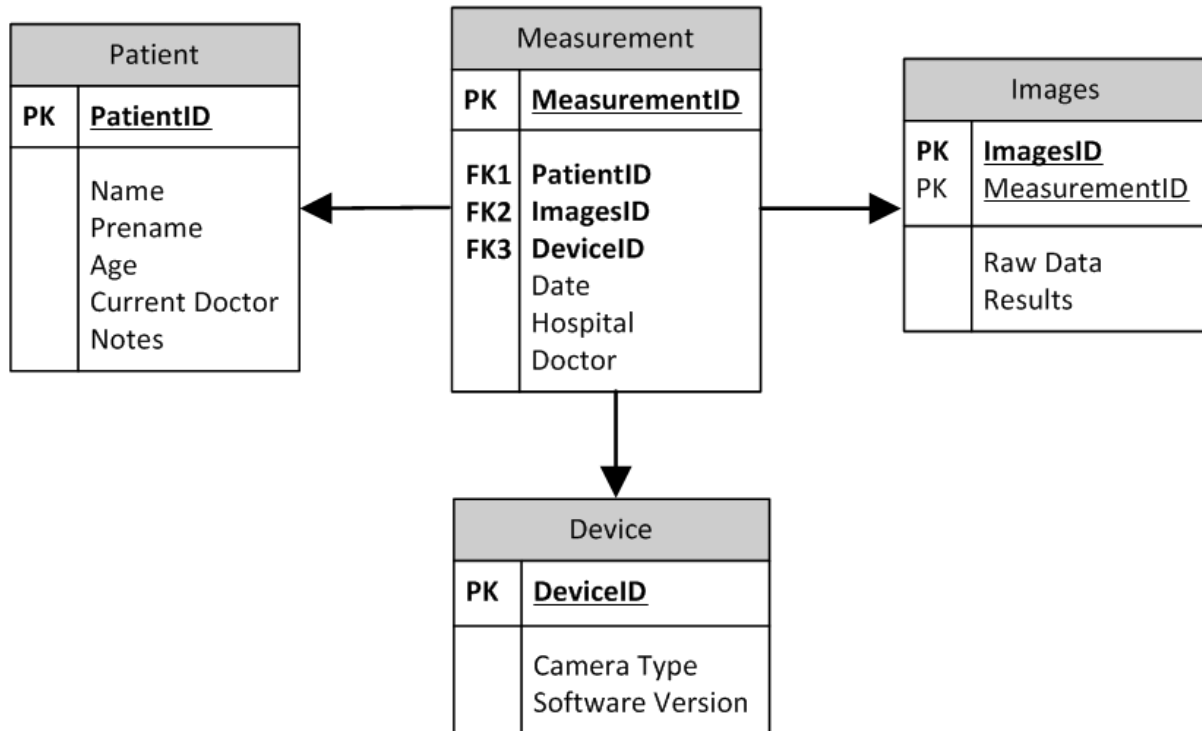
### 10.5.1 Image acquisition

The current image acquisition is held very simple. A signal from the button will trigger both, the Pic and the processor from the camera. They will then process their procedures without knowing about each other. A communication between the two devices would allow the best possible synchronisation. A sequence chart for a possible solution is shown in the following picture. Also it would be possible to take all pictures at once in a very short time window, save them into the camera buffer, and finally transmit them altogether. However, this complex extension only should be implemented if a faster image acquisition is needed.



## 10.5.2 Data storage

Following relation model shows a possible solution for the Database. It pretty much contains the important informations, so no further explanation is needed. The Current Doctor from the patient table is his current family doctor, while the doctor in the Measurement table is the one who took the measurement.



## 10.5.3 Analysis

### 10.5.3.1 blood analysis

The blood analysis is the first prototype for a oxy-deoxy hemoglobin detection. It divides the 610nm image by the 520nm image and saves it into the resultArray.

```

void Analysis::bloodAnalysis()
{
    Mat imgBlood = Mat::zeros(finalArray[0].size(), finalArray[0].type());
    double bloodFactor;
    for( int y = 0; y < finalArray[0].rows; y++ )
    {
        for( int x = 0; x < finalArray[0].cols; x++ )
        {
            double tempvalue = 0;
            for(int i=0; i<5; i++)
            {
                //finalArray[3] = 610nm
                //finalArray[2] = 520nm
                // go through the 520nm image, and divide it by the 610nm image
                double val520 = finalArray[2].at<byte>(y,x);
                double val610 = finalArray[3].at<byte>(y,x);
                bloodFactor = val520/val610;
                // since the factor value will be between 0 and 1, it is
                // multiplied by 255 for display
                imgBlood.at<byte>(y,x) = bloodFactor*128;
            }
        }
    }
}
  
```

```

    }
}
// display image, and save it to results
imshow("Hemoglobin, oxy/deoxy factor", imgBlood);
resultArray[2] = imgBlood;
}

```

### 10.5.3.2 Outline analysis

This method is also just a prototype for a contour detection. It will need a few more filters, but can be used as a basic for outline analysis. It is based on the "Sobel Derivatives" example on the OpenCV page. I recommend a combination with "Thresholding Operations" to make the first step to a proper contour detection.

Link : <http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html#basic-threshold>

```

void Analysis::outlineAnalysis()
{
    Mat grad;
    int scale = 1;
    int delta = 0;
    int ddepth = CV_16S;

    // get 390nm image
    Mat imgOutlineUV;
    imgOutlineUV = imread("images\\image1_390nm.png", CV_LOAD_IMAGE_COLOR);
    if(!imgOutlineUV.data)
    {
        AfxMessageBox(_T("No image found!", MB_ICONEXCLAMATION));
    }
    //reduce noise
    GaussianBlur(imgOutlineUV, imgOutlineUV, Size(3,3), 0, 0, BORDER_DEFAULT);

    //Calculate the derivatives
    Mat grad_x, grad_y;
    Mat abs_grad_x, abs_grad_y;
    // Gradient X
    Sobel( imgOutlineUV, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT);
    // Gradient Y
    Sobel( imgOutlineUV, grad_y, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT);
    //convert partial results back to CV_8U
    convertScaleAbs( grad_x, abs_grad_x);
    convertScaleAbs( grad_y, abs_grad_y);
    // approximate the gradient by adding both directional gradients
    addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad);
    // show result
    namedWindow( "Outline analysis UV", CV_WINDOW_AUTOSIZE);
    imshow("Outline analysis UV", grad);
}

```

#### 10.5.4 Developement tips

For researches you have to be rather specific about a MFC Application!  
Most wind32 or CLI examples probably wont work!

There are always some warnings, complaining about double to byte transformation. Yes there is an accuracy loss of 1/255 (0.0039) which should not matter too much.

Make sure to visit

<http://docs.opencv.org/doc/tutorials/tutorials.html>

for some good and fast informations about opencv.

Especially

[http://docs.opencv.org/doc/tutorials/core/mat\\_the\\_basic\\_image\\_container/mat\\_the\\_basic\\_image\\_c  
ontainer.html#matthebasicimagecontainer](http://docs.opencv.org/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html#matthebasicimagecontainer)

is important since we work with [Mat] images a lot.

#### 10.5.5 Contact informations

This project included a lot of professionals, helping to realise every specific part. This list contains contact information for open questions about specific domains.

##### **Person responsible:**

Martial Geiser - Hes-so intern

[martial.geiser@hevs.ch](mailto:martial.geiser@hevs.ch)

##### **Optics:**

Frederic Truffer, Helene Strese - Hes-so intern

[frederic.truffer@hevs.ch](mailto:frederic.truffer@hevs.ch)

[helene.strese@hevs.ch](mailto:helene.strese@hevs.ch)

##### **Electronics:**

Olivier Walpen - Hes-so intern

[olivier.walpen@hevs.ch](mailto:olivier.walpen@hevs.ch)

##### **Mechanics/Casing:**

Serge Amoos - Hes-so intern

[serge.amoos@hevs.ch](mailto:serge.amoos@hevs.ch)

##### **Database:**

Manfredo Atzori - Hes-so intern

[manfredo.atzori@hevs.ch](mailto:manfredo.atzori@hevs.ch)

##### **Expert, Dermatology**

Gianadda Elisabeth - Dr méd. dermatologue FMH

[giabe@vtx](mailto:giabe@vtx)

rue du Scex 4

1950 Sion

Tel. 027 322 11 44

Fax 027 322 11 45

##### **Expert, FotoFinder**

Christian Giradet - Director laser beate med

[direction@laserbeautemed.ch](mailto:direction@laserbeautemed.ch)

HES-SO Valais / Michael Schmid

09.07.2014