

Degree Programme
Systems Engineering

Major Infotonics

Diploma 2015

Nicolas Marty

Colours and anomalies of skin

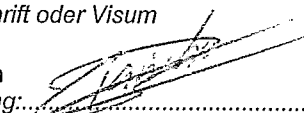
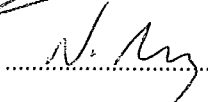
- Professor
Martial Geiser
- Expert
First name Name
- Submission date of the report
10.07.2015

This document is the original report written by the student.
It wasn't corrected and may contain inaccuracies and errors.

SI	TV
X	X

<input checked="" type="checkbox"/> FSI <input type="checkbox"/> FTV	Année académique / Studienjahr 2014/15	No TD / Nr. DA it/2015/21
Mandant / Auftraggeber <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire <i>Partnerinstitution</i>	Etudiant / Student Nicolas Marty 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 ¹ <input checked="" type="checkbox"/> non / nein	Expert / Experte (données complètes)	

Titre / Titel Farben und Anomalien der Haut
Description / Beschreibung Hyperspektrale Bildgebung ist ein vielversprechendes Werkzeug für die medizinische Diagnose der Haut. Diese Technik ermöglicht eine Abschätzung der Konzentration verschiedener Komponenten der Haut, wie zB. Blut, Melanin und Wasser. Allerdings macht die Struktur der Haut die richtige Einschätzung dieser Parameter schwierig. Die Ingenieurschule Wallis hat in den letzten Jahren drei Generationen von Instrumenten entwickelt um die Struktur der Haut und ihre Anomalien (Krebs) mit verschiedenen LEDs zu messen. Unsere vorläufigen Messungen zeigen, dass diese Instrumente ein grosses Potenzial haben und nützliche Informationen von der Haut, wie Anomalien (Krebs, Medizin) oder Creme basierende Behandlungseffekte (Alterung, Schönheitspflege) liefern. Die Ziele einer Gruppe bestehend aus drei Studenten (Pflege, Wirtschaftsinformatik und Ingenieur) sind folgende: (a) eine Bewertung der praktischen Anwendung eines solchen Instruments in der Beurteilung von Hautveränderungen (Krebs), (b) Veränderung durch die Verwendung von Cremes, (c) Entwicklung eines neuen Instruments ohne Kabel mit einfacher Bedienung und Kommunikation zu einem Server und (d) Einrichtung einer Datenbank für eine "Standard-Haut" (Student HEG). Objectifs / Ziele — Basierend auf einem Raspberry Pi2 sollen 2 kompakte stand-alone Systeme für die Bildaufnahme gebaut werden, welche 8 verschiedenfarbigen LEDs, einem Interface für die Eingabe von Patientendaten und eine Kommunikation zum Server beinhalten. — Die Mitarbeiter im Labor helfen bei der Entwicklung der Mechanik und Optik.

Signature ou visa / Unterschrift oder Visum Responsable de l'orientation Leiter der Vertiefungsrichtung:  ¹ Etudiant / Student : 	Délais / Termine Attribution du thème / Ausgabe des Auftrags: 11.05.2015 Remise du rapport / Abgabe des Schlussberichts: 10.07.2015, 12:00 Expositions / Ausstellungen der Diplomarbeiten: 26 – 28.08.2015 Défense orale / Mündliche Verfechtung: Semaine Woche 36
---	--

¹ 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.



Farben und Anomalien der Haut

Diplomand Nicolas Marty

Ziel des Projekts

Das Ziel ist die Entwicklung einer portablen Kamera mit multispektraler Beleuchtung zur Aufnahme und Analyse von Anomalien der Haut. Anwendungsbereiche sind: a) Analyse und Verfolgung von Muttermalen und Wunden, b) Untersuchung von Einfluss von Cremes

Methoden | Experimente | Resultate

Ein Raspberry Pi dient als Plattform um die Kamera sowie die Beleuchtung, bestehend aus 8 Wellenlängen, zu steuern und die multispektralen Bilder auszuwerten. Die Bilder werden in Kontext mit Patientendaten mittels WLAN auf einen Server hochgeladen. Das kompakte Gerät wird über einen Touchscreen bedient und der Akku kann mithilfe einer Dockingstation wieder aufgeladen werden.

Ein mögliches Szenario wurde mit einem Arzt besprochen und anschliessend in der Software implementiert. In Zusammenarbeit mit einem Studenten der Fachrichtung Wirtschaftsinformatik entstand ein Interface zum Hochladen der Daten zu einer Webapplikation.

Der Raspberry Pi führt selbständig eine Analyse der 8 Rohbilder durch. Die Analyse umfasst die Messung des Melaningehalts und die Rekonstruktion eines RGB-Bildes.

Das System wurde an 5 Probanden getestet. Die Rohbilder und Analysen konnten erfolgreich auf den Server übertragen werden.

Diplomarbeit
 | 2015 |

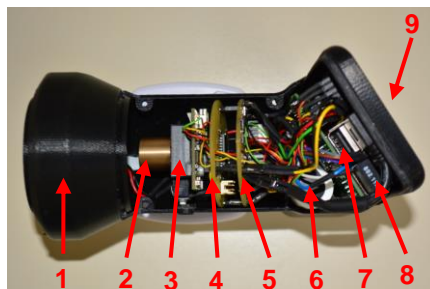
Studiengang
 Systemtechnik

Anwendungsbereich
 Infotronics

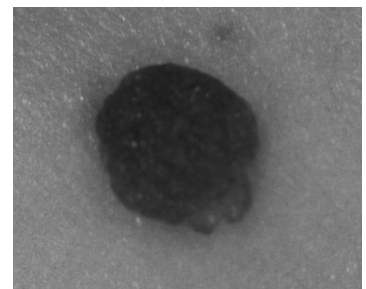
Verantwortliche/r Dozent/in
 Dr. Martial Geiser
 martial.geiser@hevs.ch

Partner
 Dr Elisabeth Giannada
 Dermatologue, Sion

OmniMedica SA
 Schlieren



1. LEDs / 2. Optik / 3. Kamera / 4. Beleuchtungssteuerung / 5. Steuerung der Stromversorgung / 6. Akku / 7. Raspberry Pi / 8. WLAN Dongle / 9. Touchscreen



Beispiel einer Aufnahme eines Muttermals. Diese hilft dem Arzt die Anomalie zu beurteilen.



Colours and anomalies of skin

Graduate Nicolas Marty

Objective

We developed a handheld camera with multispectral illumination to take pictures and to analyse anomalies of skin. The fields of use are: a) analysing and tracking skin lesions such as naevi or wounds, b) studying influence of creams.

Methods | Experiments | Results

A Raspberry Pi controls the whole device including camera, illumination and analyse and send data. The illumination consists of 8 different wavelengths. The images are uploaded to a server using WLAN and are always in context with the data of a patient. The handheld device is controlled via a touchscreen and the battery can be recharged using a dedicated docking station.

A possible scenario was discussed with a medical doctor and was subsequently implemented into the software. In cooperation with a student from the institute of information systems, an interface to upload the data to a web application was developed.

The Raspberry Pi performs the analysis of the 8 raw images on its own. The analysis includes the measurement of the melanin concentration and the reconstruction of a RGB image.

The system was tested on 5 subjects. The raw images and the analysis could be transmitted successfully to the server.

Bachelor's Thesis | 2015 |

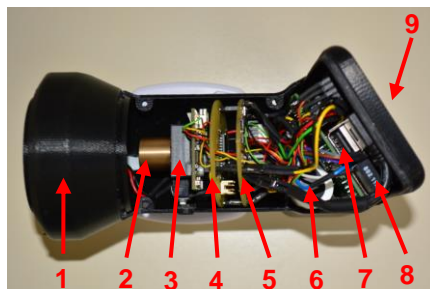
Degree programme
Systems engineering

Field of application
Infotronics

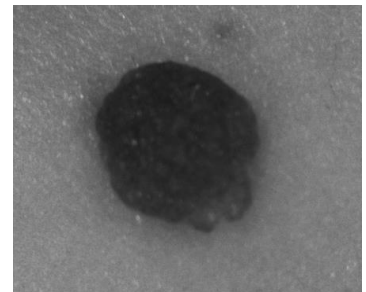
Supervising professor
Dr. Martial Geiser
martial.geiser@hevs.ch

Partner
Dr Elisabeth Giannada
dermatologist, Sion

OmniMedica SA
Schlieren



1. LEDs / 2. lens / 3. camera / 4. illumination unit / 5. power supply control / 6. battery / 7. Raspberry Pi / 8. WLAN / 9. touch screen



Example of a mole image. This image can help the dermatologist to analyse the anomaly.



Couleurs et anomalies de la peau

Diplômant

Nicolas Marty

Objectif du projet

Afin de suivre et d'analyser l'évolution, lors de traitement ou pas, de lésions de la peau (grains de beauté, plaies, tâches de sénescence, coup de soleil ...), nous avons développé une caméra portable munie d'une illumination multispectrale capable d'envoyer les images sur un serveur dédié et d'analyser ces images.

Méthodes | Expériences | Résultats

Un Raspberry Pi pilote la camera et l'illumination multi spectrale composée de 8 différentes longueurs d'onde. Toute la procédure de prise d'images et d'envoi sur un serveur dédié via le WiFi est commandée au travers d'un écran. La batterie est rechargée lorsque l'instrument est posé sur sa station d'accueil.

Un scénario possible, proposé par un dermatologue, est implémenté dans le logiciel. En coopération avec un étudiant de la Haute École d'économie a développé une interface pour télécharger et relire les données vers une application web.

À partir des 8 images brutes le Raspberry Pi effectue une analyse la concentration de mélanine dans la peau et reconstitue image RGB.

Le système a été testé sur 5 sujets. La procédure de prises de vues, d'analyse et d'envoi sur le serveur ont été effectuées avec succès.

Travail de diplôme | édition 2015 |

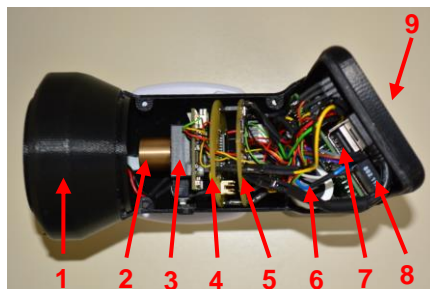
Filière
Systèmes industriels

Domaine d'application
Infotonics

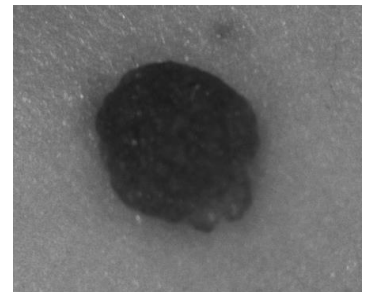
Professeur responsable
Martial Geiser
martial.geiser@hevs.ch

Partenaires
Dr Elisabeth Giannada
Dermatologue, Sion

OmniMedica SA
Schlieren



1. LEDs / 2. Optique / 3. Camera / 4. Contrôle d'éclairage / 5. Contrôle d'alimentation / 6. Batterie / 7. Raspberry Pi / 8. WLAN / 9. Écran tactile



Exemple d'une image d'une tache de vin. Cette image va aider le médecin à analyser l'anomalie.

1 Content

2	Preface.....	2
3	Introduction.....	2
4	System Specifications.....	3
4.1	Overview.....	3
4.2	Short Description.....	4
4.2.1	Controller.....	4
4.2.2	Energy Management.....	5
4.2.3	Camera & Illumination.....	5
4.2.4	Casing.....	5
4.3	Detailed Description.....	6
4.3.1	Controller.....	6
4.3.2	Energy Management.....	16
4.3.3	Camera & Illumination.....	23
4.3.4	Casing.....	25
5	Results.....	27
6	Future Improvements.....	28
7	Conclusion.....	28
8	Sources.....	29
9	Appendix.....	29
10	Signature.....	30

2 Preface

I take this opportunity to express gratitude to everyone who helped me with patience and competence during this project. A special thanks to my supervisor Martial Geiser who particularly supported me. A thanks to Frederic Truffer, he gladly answered questions and gave me convenient tips. I also want to thank Olivier Walpen and Serge Amoos who helped me with electronics and casing. Another thank you goes to Michael Clausen who gave useful hints for programming. Finally an acknowledgment to dermatologist Dr. Gianadda Elisabeth, who supported us with valuable ideas and standards.

3 Introduction

In recent years there were developed three generations of devices that take images with multispectral illumination. These images offer interesting information about anomalies of the skin and the technology shows great potential. The field of use goes from analysing and tracking moles and lesions, over to studying influence of creams until forensic.

The objective of this project is to develop a handheld device which takes advantage of the existing multispectral imaging technology. It allows the analysis of skin anomalies and transfers the data wirelessly to a server.

A Raspberry Pi serves as platform to control the camera and the illumination of 8 different wavelengths. The images are uploaded to a server using WLAN and are in context with the data of a patient. The handheld device is controlled via a touchscreen and the battery can be recharged using a dedicated docking station.

4 System Specifications

4.1 Overview

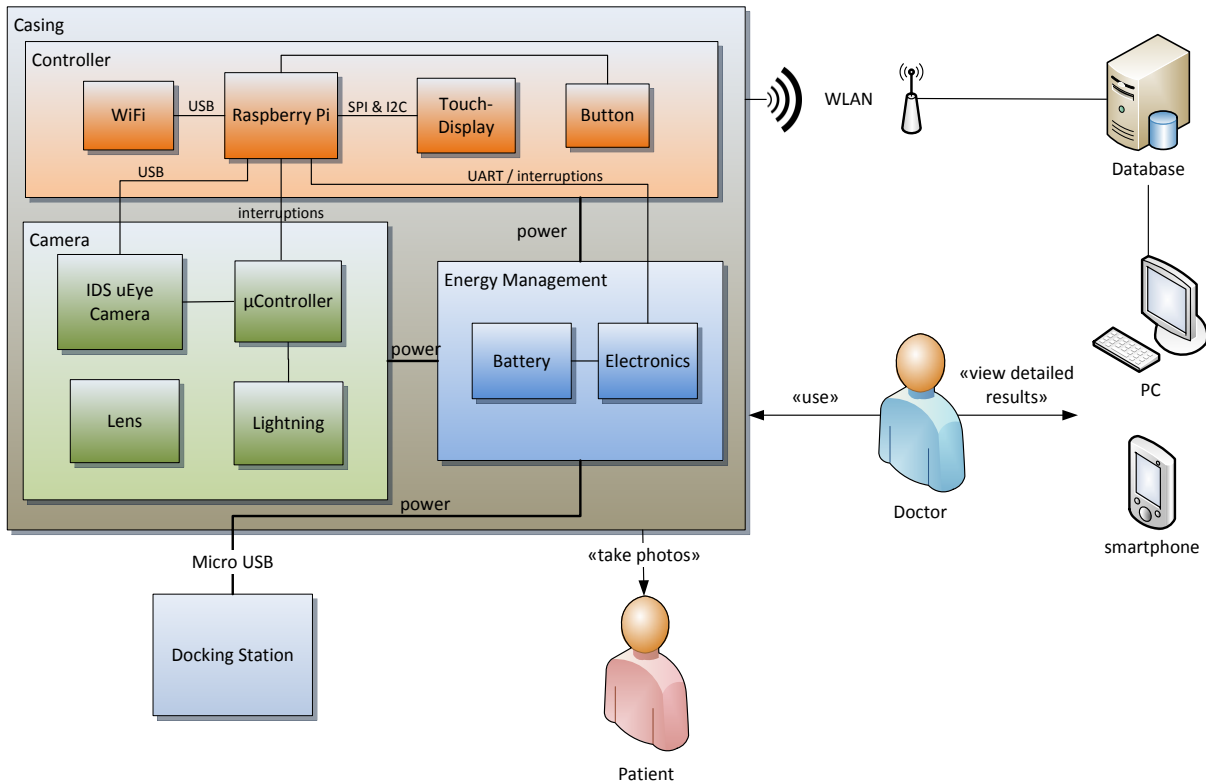


Figure 1 block diagram

The multispectral photograph system can be separated into four parts.

There is a controller that runs the main software application. It provides a graphical user interface to control the camera. Via a touchscreen the user can navigate through the application and its functions. The controller processes the taken images and uploads them to a server.

Because the system is portable, there is an energy management consisting of an energy storage, power conversion electronics and a microcontroller that interacts with the Raspberry Pi. The electronics allow recharging of the energy storage.

The third part is the camera with the optics and lightning mechanism. The optics and lightning are crucial to the quality and utility of the raw images. The lightning is made up of LEDs of eight different wavelengths. The illumination is driven by a microcontroller.

The fourth project part is a casing that contains and protects the sensitive electronics and optics. The casing allows an easy and comfortable usage of the device. A dedicated docking station can be used for recharging.

4.2 Short Description

4.2.1 Controller

Use Case

The multispectral images taken with this photograph stand always in context with a patient. The patient data and the images must be well organised and the process of taking images must fit to the workflow of a doctor's office. The complete system consists of the photograph and a web application to view the detailed patient data and images. The web application is developed by a colleague from the institute of information systems. A meeting with dermatologist Dr. Gianadda offered valuable information about the workflow in a doctor's office. In joint work with my colleague a use case and scenario has been developed and implemented in both projects.

Raspberry Pi

The semester project showed that the use of a Raspberry Pi as controller is the best solution for this project. It is a powerful tiny computer and able to drive the project-specific camera and to connect to wireless networks. A small touchscreen was installed during the semester project and the programming can be done with C++/Qt. Image processing can be done directly on the Raspberry Pi.

To hide the Linux system from the user, there were done several configuration changes to it. After start-up, it automatically launches the photograph application and on exit it automatically shuts down. A splash screen during start-up hides even more the Linux system. The resistive touchscreen used in the semester project has been replaced by its successor that has a capacitive touch panel with higher precision and behaviour like a modern smartphone, the resolution did not change though. To achieve a high frame rate on the display its communication bus was modified.

Software Application

The Software is written in C++ with the Qt Framework. This software is the project's heart. It must control the camera, the wireless data transmission, the display, the lightning system and communicate with the energy management.

Before the programming begun, software engineering was done. The components mentioned above could be realized each with its own class. A central logic class contains the system's behaviour in form of a state machine. To keep the application responsive the graphical user interface runs in a separate thread.

The state machine implements the doctor's workflow and meets the criteria that were elaborated with the information systems student, Andreas Imsand. To find a good solution of uploading the images to the server several meetings took place with him. The result of these meetings was a description of a web service that can be accessed from the Raspberry Pi via http requests.

The software offers an image calibration function to determine the ideal exposure time for each wavelength. The lightning causes slightly inhomogeneous illuminated areas. To filter these inhomogeneities a second calibration step calculates a filter for each wavelength.

The camera is coming with a driver for embedded Linux systems and an application programming interface. The well done documentation explains the vast amount of functions available for the camera. The camera class makes use of this API and implements in particular two modes: A live view

mode with reduced resolution and video capturing, and a picture mode with the maximal resolution and trigger capturing.

As soon as a series of eight raw images (one of each wavelength) is taken, the analysis is done automatically and the pictures are transferred to the server.

4.2.2 Energy Management

The Raspberry Pi consumes much energy. That's the price of having a powerful computing platform. To make it run autonomously without the need of an external power supply a solution for the energy storage was elaborated. Supercapacitors and UPS systems did not fit the project criteria. Supercapacitors are very expensive and they're used in systems where high power in short time is demanded. UPS (uninterrupted power supply) systems are used to supply the Raspberry Pi for a short time if there is a failure in the external power supply. Both ideas have the problem of being too large to install them in a small and handy casing. The choice felt on Lithium-Ion batteries. They are available at high energy capacities and the number of recharge cycles is very high. Li-Ion batteries don't need additional maintenance and can be installed fixedly in the casing. The battery can be recharged using the docking station.

A microcontroller observes the charging circuit and indicates its state with three LEDs positioned next to the display. Pushing a power button emits a signal to the microcontroller to power on the Raspberry Pi. During the period of use the Raspberry Pi's application polls the battery state via serial communication and displays a battery symbol on top of the screen indicating a battery percentage. A limiting low voltage prevents the battery from deep discharge and damage. Once the user exits the Raspberry Pi's application a signal is emitted to the microcontroller to turn off the power supply of the Raspberry Pi.

4.2.3 Camera & Illumination

The camera was used already in earlier projects. The special feature about this camera are its wide range of detectable wavelengths. The existing illumination mechanism with 8 wavelengths was re-used but some minor changes were applied. The PCB Layout has been modified to fit into the new small casing. The microcontroller software was modified to work with the Raspberry Pi. In the old version the lightning was driven directly by the camera which caused the need of a complicated synchronization mechanism. The new version is connected to the Raspberry Pi and works simpler. The Raspberry Pi activates the correct illumination and then it takes a picture.

4.2.4 Casing

The casing is made of two halves. The electronics and optics can be mounted and tested in one half and then the casing can be closed with the other half. The power PCB and the lightning PCB form a stack and are placed below the Raspberry Pi. The PCBs have been fabricated to match exactly the shape of the casing. A dedicated docking station with a Micro-USB connector allows connecting common wall adapters.

4.3 Detailed Description

4.3.1 Controller

Use Case

The system’s functionalities are inspired by Fotofinder – the device used in Dr. Gianaddas doctor’s office. It’s a specialised device for analysing moles. Before a macro image of a mole is taken with Fotofinder, an overview image of the body is taken. The assistant marks the mole to scan with a number. The following macro images will be associated with this mark. If there are numerous moles close to each other, this function helps a lot to avoid confusion. The macro images are saved with time and date. If there are already images of a mole, they can be compared. This allows keeping track of the development of the mole. The analysis returns a risk score to help the doctor doing the diagnosis.

For this project, the functionalities were simplified. The selection of the mole’s position – called region – is done without an overview image. The picture comparison will not take place on the device itself because the display is too small. The raw images and result images are uploaded to a server. The upload is done in context with patient data and region. The images can be reviewed and compared with the web application.

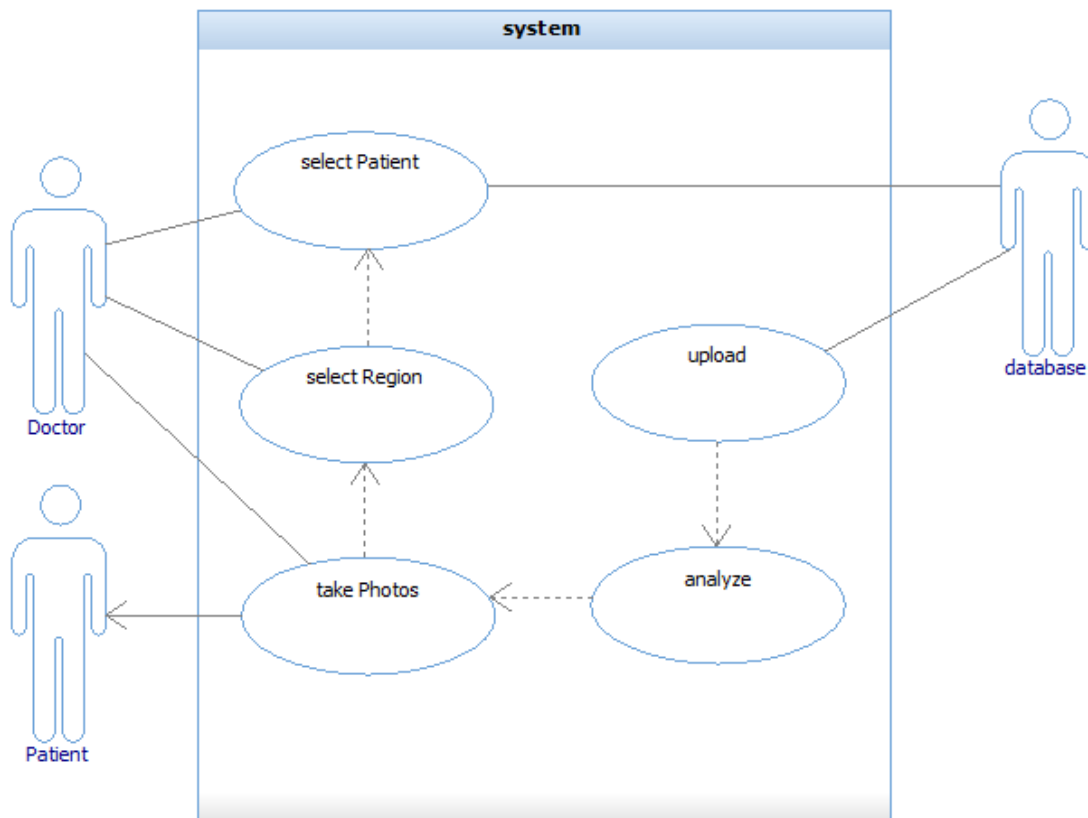


Figure 2 use case diagram

Raspberry Pi

The Linux operating system was modified to be hidden from the user because it must not see anything of the Linux system nor must he enter some commands to launch the application. The process from start up till shutdown is automat-ed.

The red path shows the lifecycle of the Raspber-ry Pi’s operating system and its application. The blue path shows the actions of the PowerMan-ager during the period of use. (The PowerMan-ager is described in the chapter of energy man-agement.) The PowerManager turns on the Raspberry Pi as soon as a push on the Power-Button is registered. The Raspberry Pi starts im-mediately with the boot sequence and execution of the camera application. As soon as the user exits the application a shutdown signal is sent to the power manager. The power manager waits a specific time to let the Raspberry Pi shutdown safely and then the power supply to the Rasp-berry Pi is turned off.

Each red box corresponds to a modification in the Linux system. The exact description of these modifications can be found in appendix A.

Another important modification is done in dis-playing the image. The Raspberry Pi’s default Video Output is shown via HDMI. The device corresponds internally to the framebuffer0. The touchscreen’s output is written to framebuffer1.

Writing to the framebuffer0 is faster than writing to framebuffer1. Further, during boot time there is no access to framebuffer1. To take advantage of the faster framebuffer0 and ability to show a splash screen during boot time, a frame buffer copy tool was installed on the Raspberry Pi. This tool raises the frame rate up to 25 fps and allows a smoothly displaying of the camera preview. The de-tailed configuration steps are described in appendix A.

The Wireless Network connection is controlled by the Operating System and the configuration (SSID, username, password, etc.) is stored statically in a configuration file. The WLAN must be reconfigured to be used in another environment as HEVS. How to configure the WLAN is also described in appen-dix A.

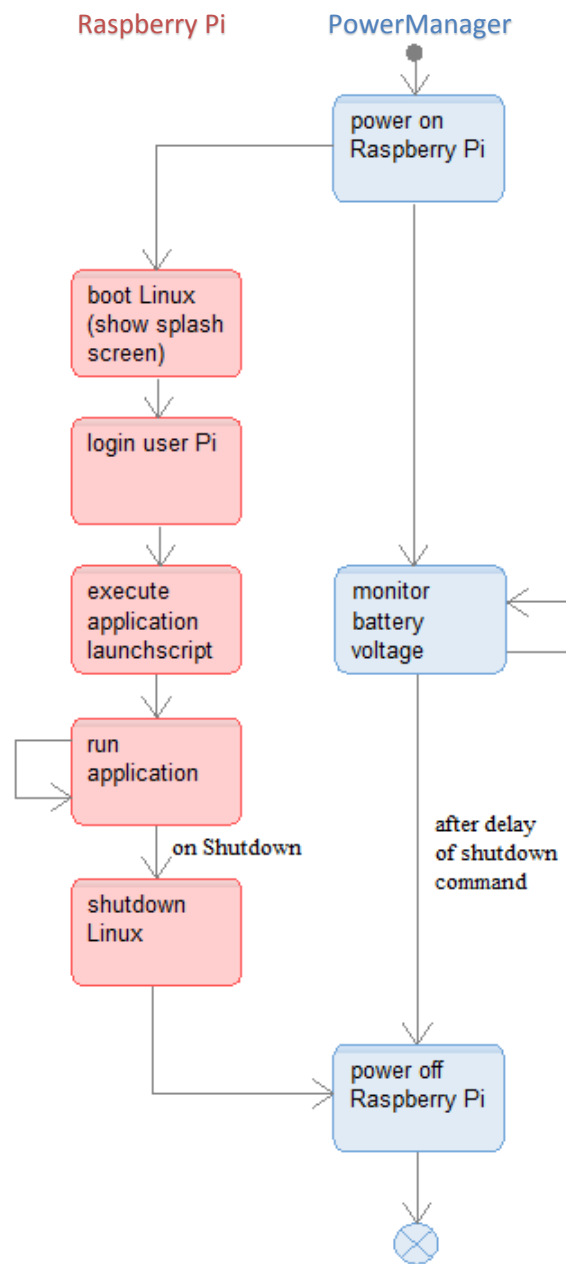


Figure 3 Raspberry Pi operation

Software Application

The programming is based on the following software engineering results. The different physical components that are used in the application could be realised each with its own class. They will be explained later. Then the Model-View-Control pattern could be applied. The result is a patient data class and a logic class. These classes decouple the data, the user interface, and the systems behaviour. The classic MVC-pattern says the controller modifies the model, the model notifies the window and the window creates events in the controller. In this application, there is no relation between the model and its view because the logic decides which data to show on the display.

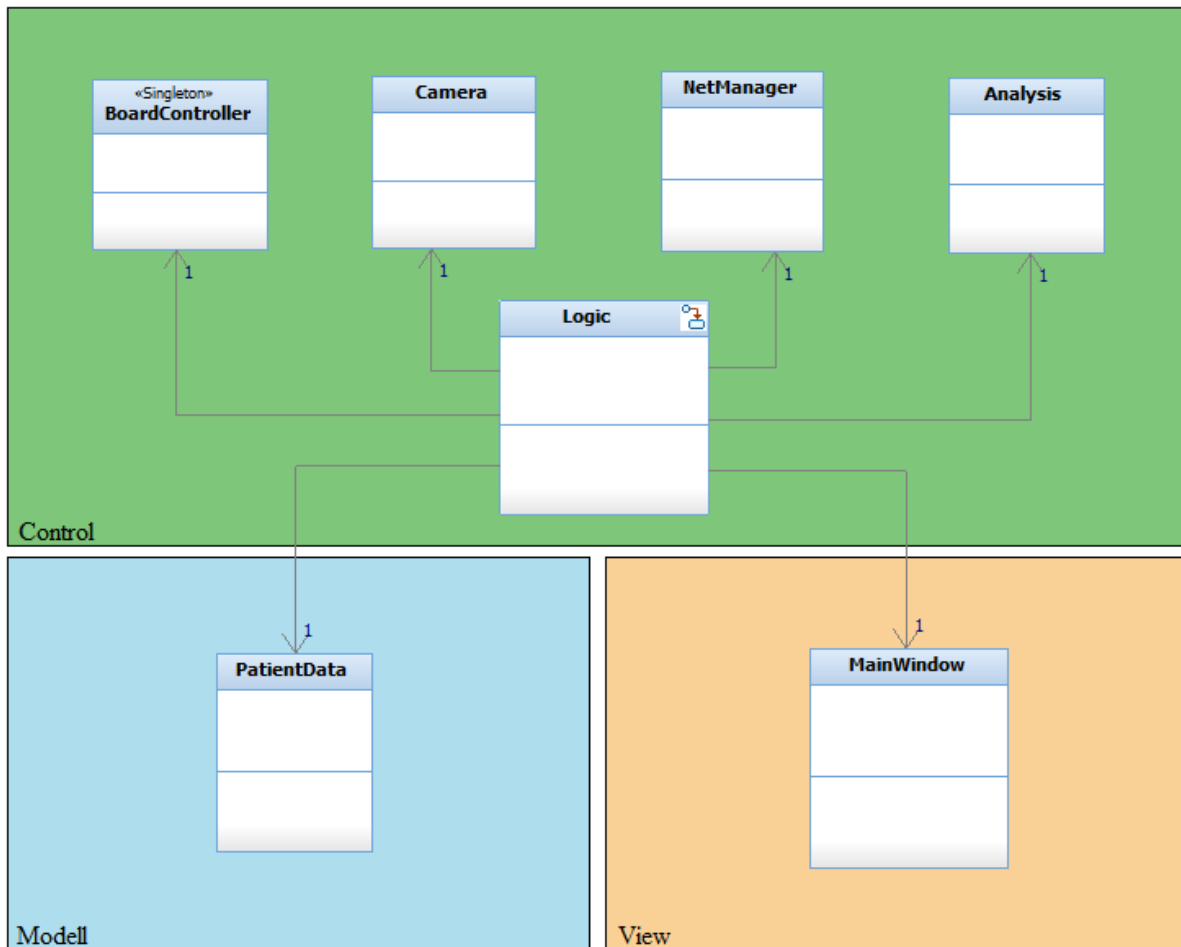


Figure 4 class diagram MVC-pattern

The reason why the logic has pointers to objects of all classes is because it acts as a factory. This means objects are initialised and connected by the logic.

To keep the application responsive and execute the logic in the background a separate thread for the user interface is created. The camera’s live view needs also an own thread. While the physical camera is running in video capturing mode, it is creating events whenever an image is ready to read. The event thread listens to the physical camera and notifies the camera object if an image is ready to read. The camera reads the image from the memory and sends it to the mainwindow. The camera reads the image from the memory and sends it to the mainwindow.

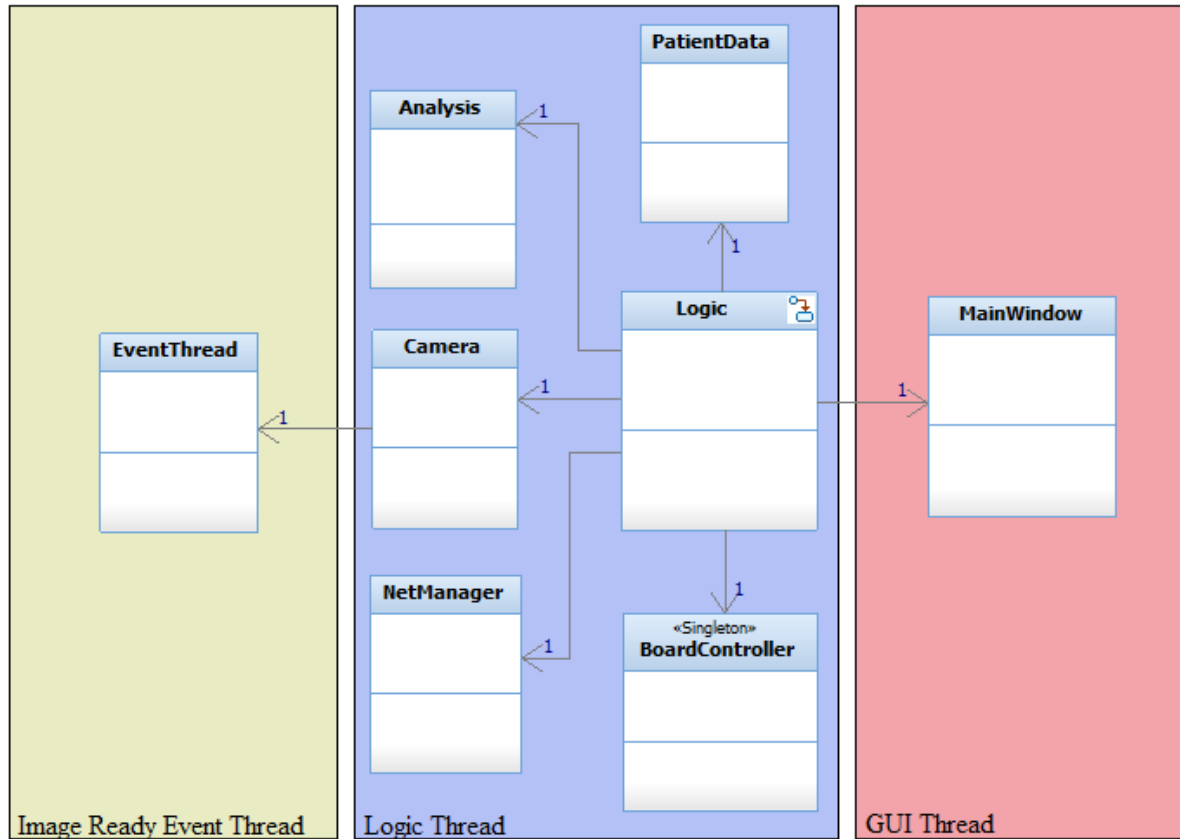


Figure 5 class diagram ordered by threads

The communication between threads is asynchronous. Qt’s Signal-Slot mechanism is made exactly for asynchronous communication. Data can be passed safely from one thread to another using this mechanism. The camera can emit the image as a signal and because this signal is bound to a slot in the mainwindow, the image will arrive to the mainwindow’s thread and there it will be displayed. The work of connecting signals and slots is done during the initialisation, right after the objects are created. Another big advantage of signal-slots is that objects do not need a pointer to the object where the data is sent to. A signal is emitted regardless of a slot is connected or not. This behaviour decouples the classes and increases the program’s modularity. Classes can be exchanged entirely without affecting the rest of the program (except the factory). To show clearly the signal flows, another diagram is added below.

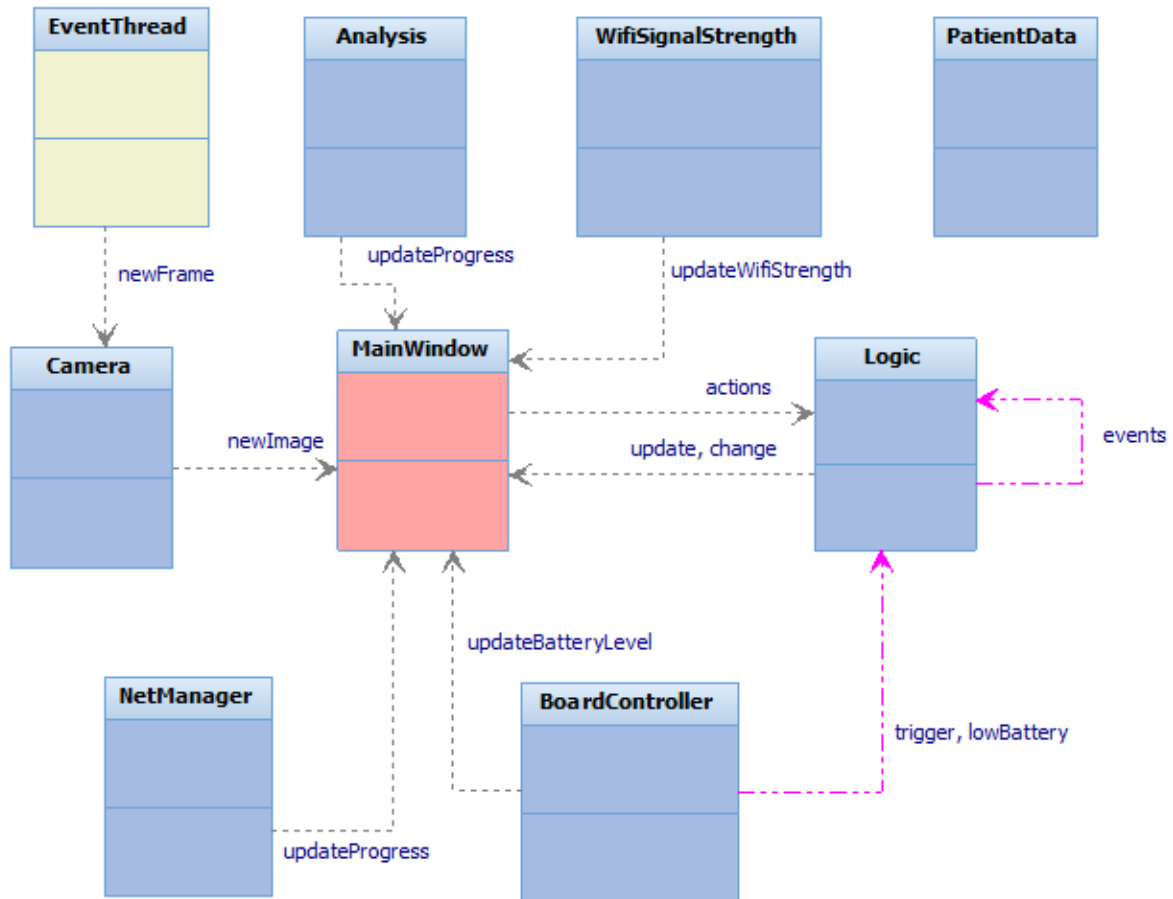


Figure 6 signal flow diagram

The colours correspond to the threads where the objects of these classes are living in. The Signal-Slot mechanism can also be used to call methods asynchronously on the same thread. The three purple signals are used this way. The trigger- and lowBattery-signal are created on a low level interruption. The event-signals are created by the state machine itself.

Logic

The most important class is the Logic. It implements a state machine that is responsible for the programs behaviour. The statechart below shows each state and the events that lead to the next state.

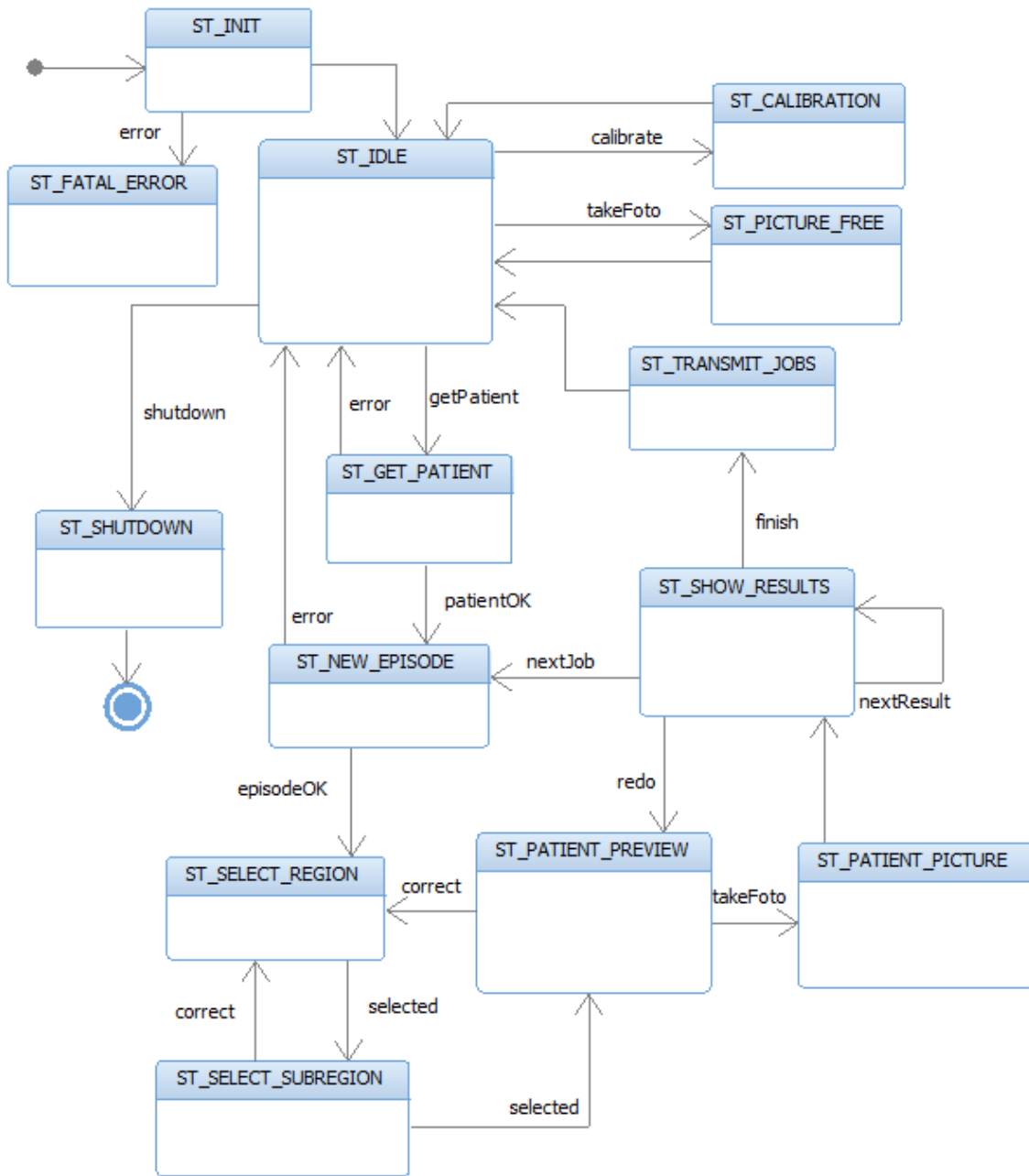


Figure 7 system state chart

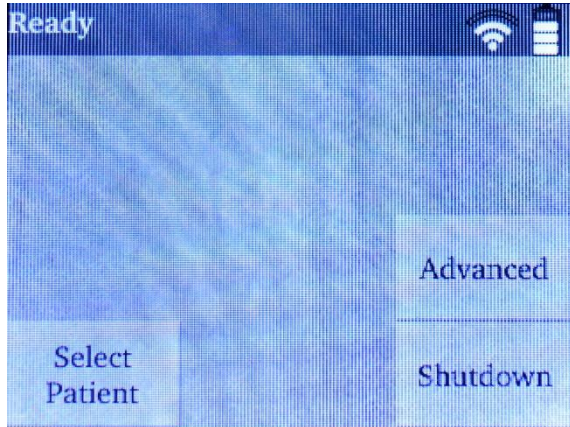


Figure 8 screenshot of idle state

The program starts with the initialisation. After that the program goes into the idle state. In this state multiple actions can be done. If the trigger button is pressed during idle state a series of pictures is taken without patient context. These pictures are uploaded to a general folder on the server. A click on the Advanced button shows additional functions. Within the advanced functions, the preview wavelength can be selected manually, the calibration can be started and the homogeneity filter can be activated or deactivated.

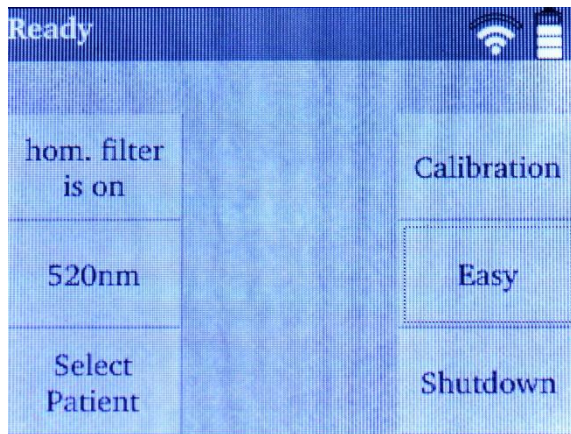


Figure 9 screenshot of idle state (advanced mode)

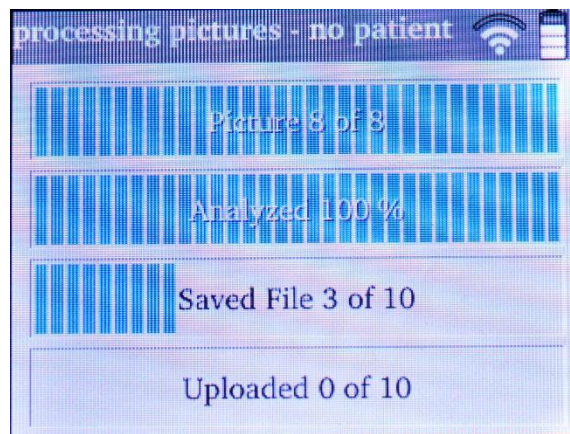


Figure 10 screenshot of free picture state

A click on select patient starts the process of taking patient pictures. To select a patient a 4 digit ID must be entered. Then the region of the mole must be chosen. As soon as the selections are done, a patient preview appears. The trigger button leads to the patient picture state where the pictures are taken, analysed and stored in the RAM.

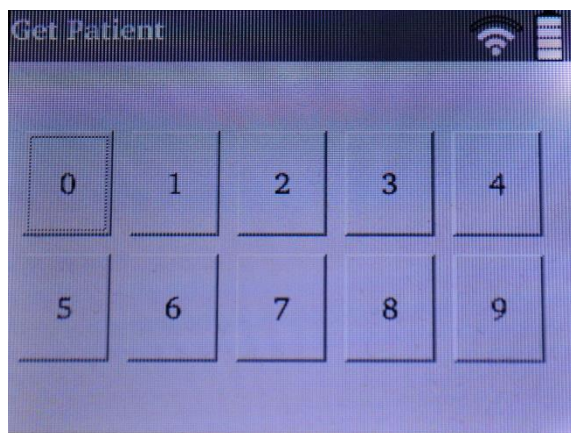


Figure 11 screenshot of select patient state

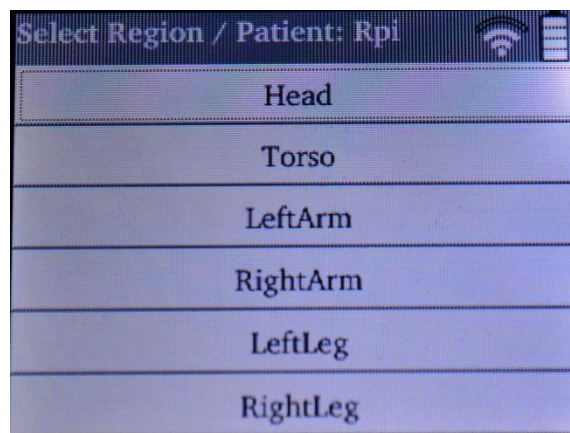


Figure 12 screenshot of select region state

The next state shows the results. The picture can be retaken if necessary. If there are more moles to scan, jobs can be added. In the end all the images in the RAM with the patient data are uploaded to the server where they can be accessed with the Web Application.

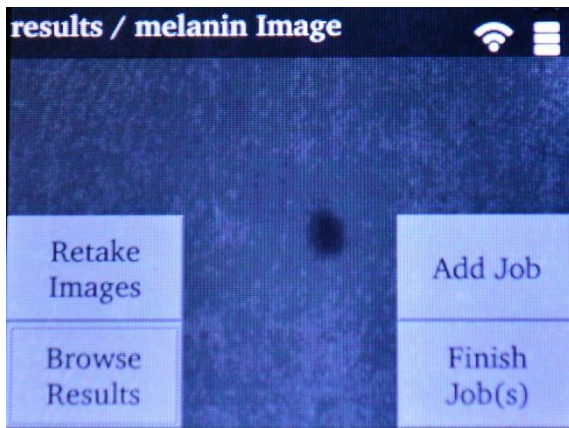


Figure 13 screenshot of show results state



Figure 14 screenshot of transmit jobs state

MainWindow

The mainwindow is the graphical class that implements all views, buttons, etc. The mainwindow itself does not implement logic. Button clicks are transmitted to the logic class where the action is processed. The mainwindow uses Qt's QStackedLayout feature to organise all the different screens. During the initialisation of the mainwindow all screens are constructed. The logic class selects the widget to display for each state.

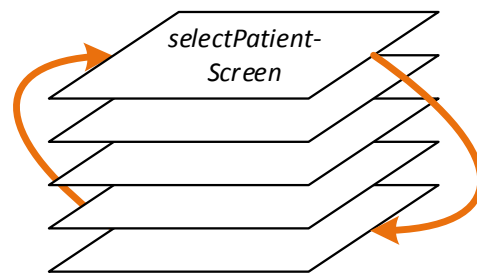


Figure 15 stack layout principle

Camera

The camera class makes use of the ueye API to access and control the camera. The camera is initialized at the program start. As soon as the camera can be accessed the picture and video memory are allocated. The camera class implements two imaging modes: A low resolution, high framerate video mode used during liveview and a high resolution photograph mode used to take raw pictures of moles. If switching from video mode to photograph mode, the EventThread is stopped, the image capturing is changed to trigger mode, the resolution is changed and the image memory is changed. The camera class implements also methods to let the logic change easily the exposure time and take a picture on demand.

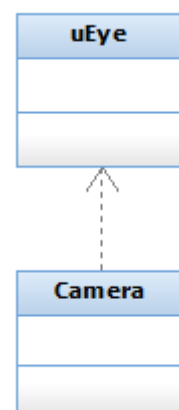


Figure 16 camera includes uEye API

Analysis

To make a meaningful analysis a calibration of the system is important. There are two calibration steps. The first one standardizes the cameras sensitivity for each wavelength. If taking pictures of a calibration surface, images of each wavelength must have the same brightness level (NORM). This is achieved by setting the correct exposure time. The algorithm to get the exposure time works approximately and is explained on the following picture.

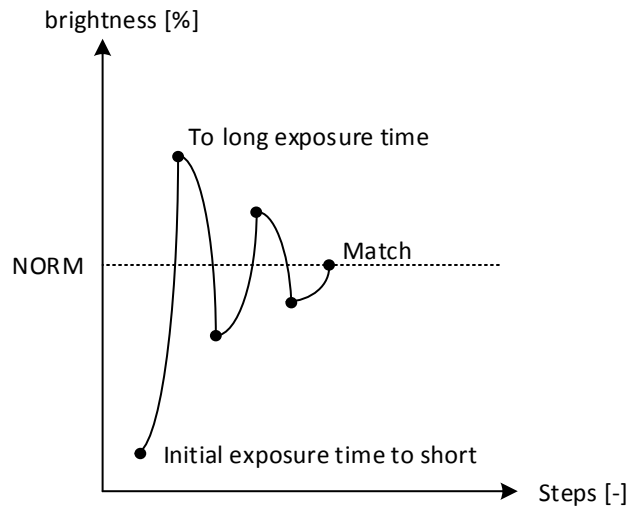


Figure 17 calibrate exposure time algorithm

The second calibration step treats the problem of inhomogeneous illuminated areas caused by the different lightning. For each set of LEDs a filter file is created. The filters are applied for each raw image taken by the camera. Michael Schmid worked already on the homogeneity filter in his bachelor thesis last year. The algorithm could be rewritten to work on the Raspberry Pi. The illumination is already very homogenous without filter.

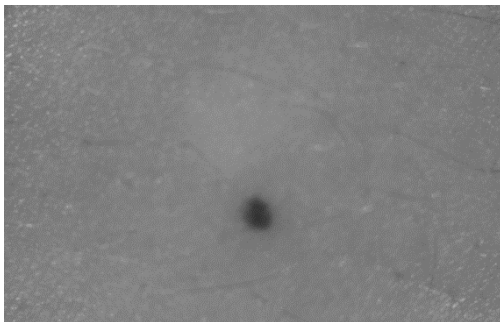


Figure 18 image without homogeneity filter

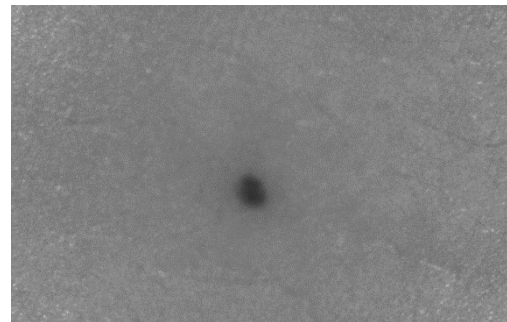


Figure 19 image with homogeneity filter

The function to detect melanin could be taken from the earlier project too and has been reimplemented on the Raspberry Pi.

Another analysis method calculates a coloured image out of the raw images with wavelengths 470nm, 520nm and 610nm. A pixel consists of 3 bytes. One for red, blue and green. The coloured image is created by setting the value of 470nm as blue part, the 520nm as green part and the 610nm as red part for each pixel. This algorithm is probably not that useful for a doctor but there is an impression of the capabilities of multispectral illumination.

The analysis class offers a method called `dynamicAnalysis()`. The logic calls this method during runtime. The function takes a series of raw images and returns a map of result images. A map consists of key and value pairs. In this case the key is from type string and contains the name of the analysis and the value is the result image. The function is named `dynamic` because the number of returned result images can vary. If analysis methods are added in the future the programmer must only work on the analysis class. Analysis methods can be added to the dynamic analysis function and the rest of the program can be left untouched.

NetManager

The network manager is responsible to upload the images to a server. It is also the interface to the web application. The following flowchart shows an excerpt of the program that works with a patient. The blue marked actions involve requests to the server.

At the beginning a patient must be selected. Because the Raspberry Pi's touchscreen is small there are only 10 number buttons to enter the patient's number. This number consists of 4 digits. It corresponds to a patient in the database. If the fourth digit is entered the `getPatientRequest()` is sent to the server. The patient information is sent back in the response. If the patient number was incorrect the response contains an error and the next steps are cancelled.

The next step creates a new episode of care. The episodes of care are used in the web application to show the history of a specific mole. This is done by sending the `createEpisodeRequest()`. The response contains the episode id. It is stored in the patient data.

As soon as the images are taken, the data can be uploaded. They must be uploaded to a specific directory. The `createDirectoryRequest()` makes the server application create the image directories out of the region and episode. The paths are sent back as response.

The `uploadResultsRequests()` takes all the information from the patient data and sends it to the server. The data (except the images) transferred to and from the server is in form of json strings.

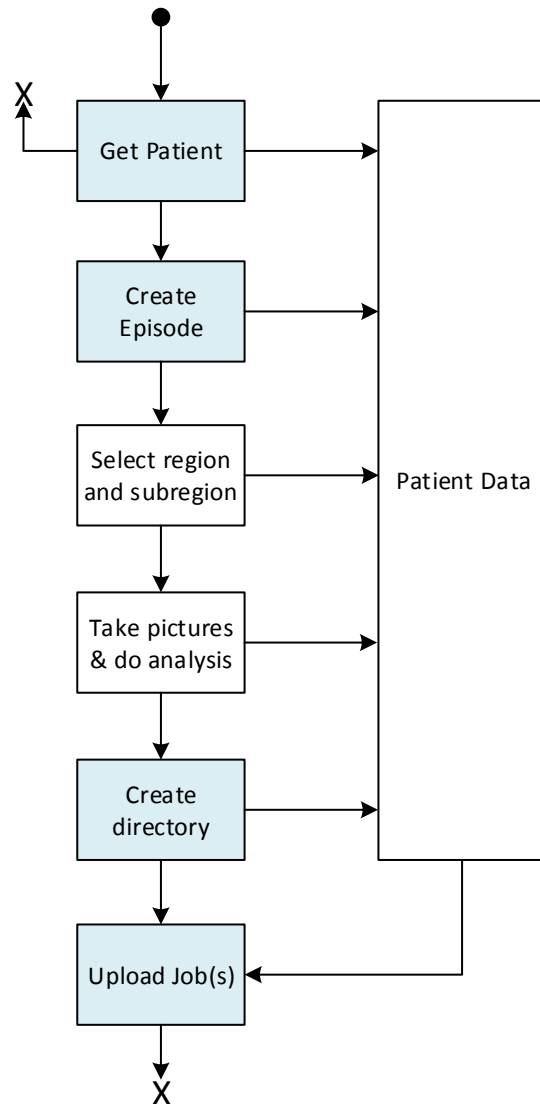


Figure 20 program excerpt

PatientData

The PatientData class stores the images in context with patient information. The data is set during a job and cleared after a successful upload.

BoardController

The BoardController uses the wiringPi library to create an abstraction layer of the signals connected to the GPIO Pins of the Raspberry Pi.

There are two interrupt routines. One creates a triggerEvent if the trigger-Button is pressed and the other one creates a lowBatteryEvent if the PowerManager raises the corresponding signal.

The BoardController implements a method to send the shutdown signal to the PowerManger and to poll the battery level. It implements also the methods to turn on/off the illumination and to change the wavelengths.

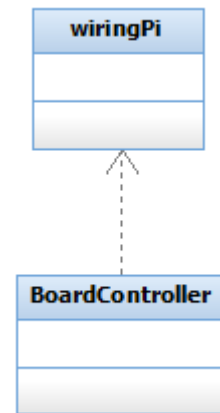


Figure 21 Board-Controller includes wiringPi library

TouchTransform

Qt has touchscreen support. If a touch device can be found at application start, touch events will be created. Unfortunately there is a bug in the touchscreens driver or in Qt and the coordinates received with a touch event are completely wrong. Luckily, a touch event has a method called rawScreenPositions(). The coordinates read from this method are correct but don't consider the screen orientation. The class TouchTransform acts as event filter that modifies all incoming touch events using the rawScreenPosition() method.

Variables

The variables file contains no class but many configuration parameters used at different places in the program. For example the server URL and the GPIO pin layout are defined in this file.

4.3.2 Energy Management

Battery choice

There are many existing battery technologies. Each technology has its own pros and cons. Lead Batteries have a huge capacity but are heavy and bulky. NiCd batteries are cheap but an old technology and known for their memory effect. The newer NiMh batteries are also cheap but they do not suffer the memory effect. Li-Ion batteries are expensive but they have a high capacity, low self-discharge and if well treated, a long lifetime. There is another Lithium based technology, called Li-Polymer. These batteries are mainly used in remote controlled planes because they are very light weight. They are very sensitive to temperature changes, impacts and need good care.

At this point Li-Ion or NiMh technology fits best for the project but there are still some more facts about these two types.

Li-Ion batteries don't need maintenance. The user of the final device does not have to cycle the batteries as he would do it with NiMh nor must he replace them. At least until a very high number of charge cycles is reached and the capacity becomes significant lower as at the beginning when the battery was new. Then the self-discharge rate of Li-Ion batteries is lower than NiMh. The only advantage left for NiMh is the price.

One Li-Ion cell has a nominal voltage of 3.6V and if fully charged 4.2V. The Raspberry Pi needs 5V, so just a one-cell Li-Ion battery with a step-up converter can be used to power the Raspberry Pi.

Li-Ion batteries have a simple charging algorithm consisting of 2 phases. The charging cycle begins with a constant current phase. As the name says, during this phase the battery is charged with a constant current. If the voltage of the cell reaches 4.2V phase 2 begins. Phase 2 is called constant voltage phase. The charge controller maintains the 4.2V and decreases the charge current slowly until it reaches the cut-off point. This point is set at 10% of the constant charging current. Setting the point lower charges the battery to a higher capacity at the cost of lifetime.

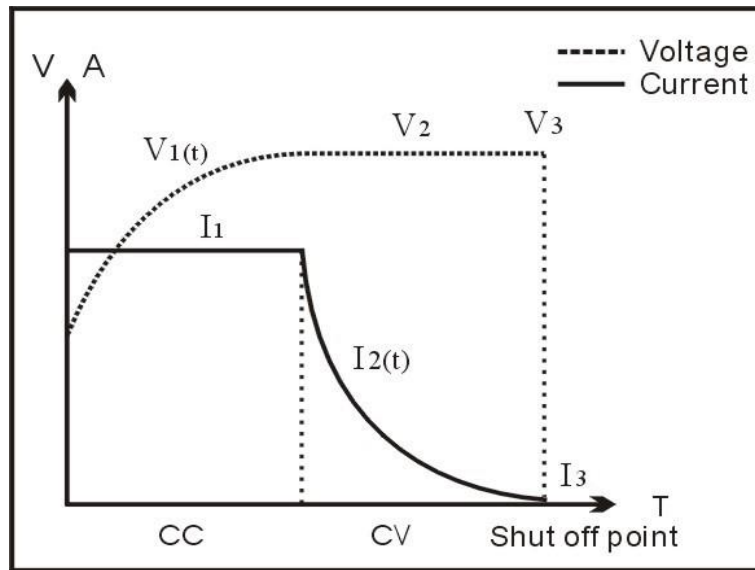


Figure 22 Li-Ion charging algorithm [2]

There are integrated circuits that implement this charging algorithm. This makes the development of an electronic circuit much easier and faster.

Li-Ion batteries offered the most advantages so this technology is used in this project. In particular a single-cell Li-Ion battery with a capacity of 2900mAh and a nominal voltage of 3.6V was chosen.



Figure 23 battery to be used in this project

Supercapacitors

Another idea was to use super capacitors as energy storage. Supercapacitors have a high capacity usually >100F. They are used in Formula 1 in the KERS Kinetic Energy Recovery System. They are capable of delivering very high power and can be charged very fast. The number of charge cycles is higher than one million. All these advantages make supercapacitors expensive. Supercapacitors have little electrical strength. The voltage at which a super cap can be charged is usually below 3V. The energy stored can be calculated as follows.

$$W = \frac{1}{2} * C_{max} * U_{max}^2$$

To run the Raspberry Pi for half an hour at full load it consumes the following amount of energy:

$$W = U * I * t = 5V * 1A * 30min = 9000Ws$$

To make an approximate calculation a current of 1A was taken.

A supercapacitor that is capable of storing this amount of energy is very expensive. To make a concrete estimation a supercapacitor from Kemet was taken. The capacitor is rated 3000F / 2.7V and costs approx. 175.- SFr. at farnell. The dimensions of this capacitor are 60.7 x 144 mm (D x L).

An estimation of run time can be calculated with a combination of the two formulas above.

$$t = \frac{C_{max} * U_{max}^2}{2 * U * I} = \frac{3000F * (2.7V)^2}{2 * 5V * 1A} \sim 37 \text{ min}$$

The biggest advantage of a super capacitor – its rapid charge and discharge times – cannot be used. An AC/DC USB adapter delivers 2A but the super cap could be charged with 145A (1s peak 2200A). The discharge rate is also far below the rated current.



Figure 24 Kemet Supercapacitor [1]

UPS

A Raspberry Pi UPS (uninterrupted power supply) is not convenient for this project as the main purpose of a UPS is to power the Raspberry Pi during a short period of time where the cable power supply is turned off. Further, these systems are mounted usually on the GPIO Header of the Raspberry Pi where the display is already placed. Any solution to use both UPS and display would lead to a too large casing.



Figure 25 Supercapacitor UPS for Raspberry Pi [1]

Electronics

General function

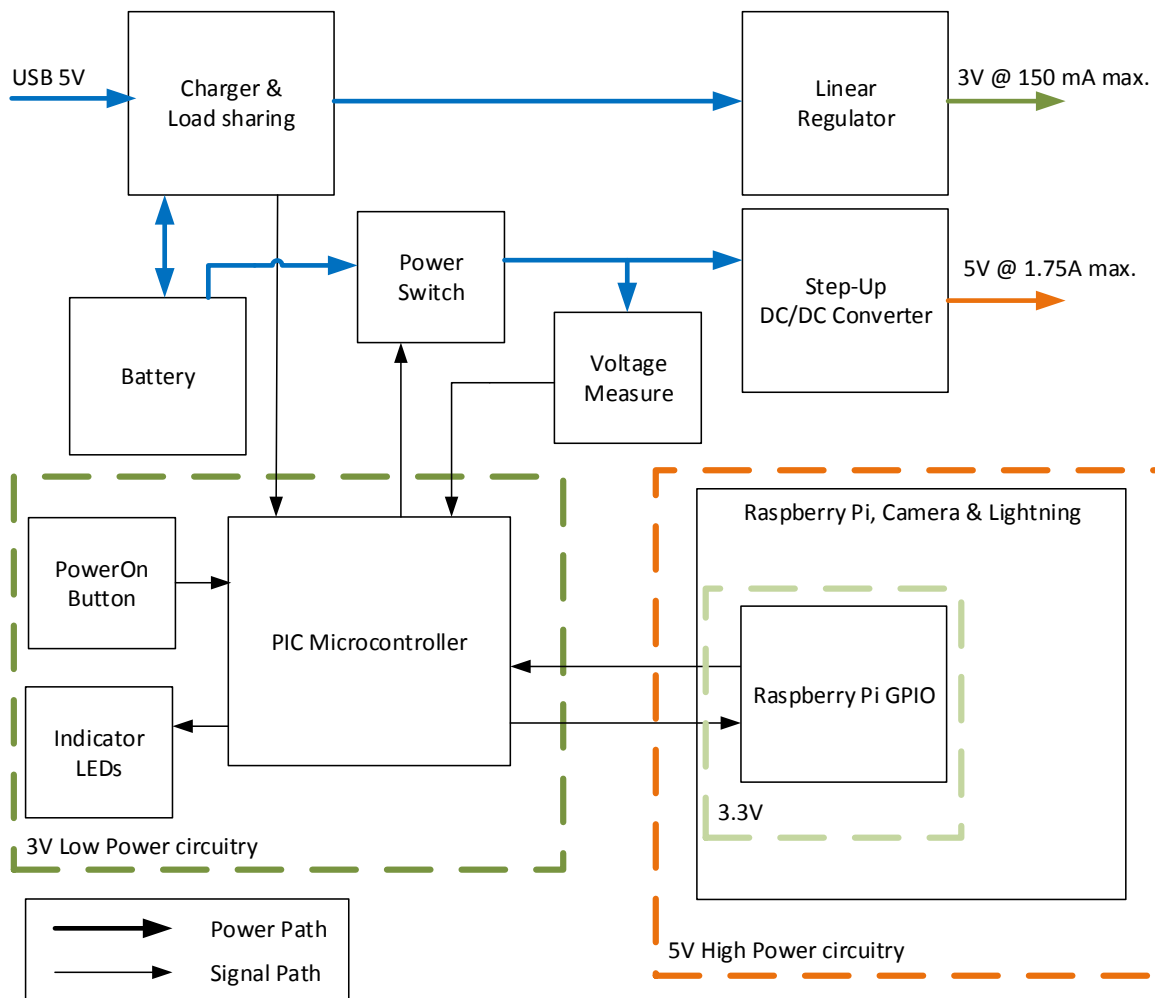


Figure 26 PowerManager electronics block diagram

Charger & Load sharing

An integrated circuit provides the functions of charging the battery and sharing the system load between the battery and the external power supply. The IC uses the CC-CV charge algorithm mentioned above. The constant charge current and the cut-off current can be determined by two resistances. The charger allows a maximal charge current of 1A. This is within the battery specifications. Three status signals are connected to the Microcontroller. They offer information about the state of the charger. These signals will be used later in the programming. The integrated circuit has also a load sharing function. With this function a device can be charged while it is in use. To use this function all the current must pass through the charger. In other words, the battery must only be connected to the charger, nowhere else. The maximal, shared current is 1.65A according to the datasheet. At high currents the charger is going to heat up until it activates the thermal protection (shutdown). The maximal current drawn from the battery is about 1.25A if the Raspberry Pi with strongest illumination activated. This current leads already to overheat. A small cooler glued on the IC with thermal compounds did not solve the problem. The battery must be connected directly to the power switch. The high currents do not pass the charger anymore and the problem is solved. The load sharing function can just be used to power the microcontroller. If the device is placed on the docking station to charge while the Raspberry Pi is running, the power switch is instantly turned off. That's because the charger assumes that all the current passes through it. The system can be damaged if this condition is not met.

DC/DC step up converter design

The step-up converter is able to convert the battery voltage going from 3V to 4.2V to the Raspberry Pi's 5V. The converter can output a maximal current of 2.1A. It is in form of an integrated circuit. It has a fixed output voltage of 5V. This is more accurate than an adjustable converter because of precise internal resistances. The worst case scenario where the battery voltage is 3V and the Raspberry Pi draws the maximal current of 768 mA (measured during semester project) leads to a battery discharge current of at least 1.28A according to the law of energy conservation.

$$I_{bat,max} = \frac{V_{rpi} * I_{rpi}}{V_{bat}} = \frac{5V * 768mA}{3V} = 1.28A$$

The maximal battery discharge current is 2.95A. A current of 1.28A is far from this limit. Nevertheless, the system should not draw more than 1.75A.

Linear Regulator

This regulator is designed to output 3V for low power devices. It has a very low quiescent current. The microcontroller will be powered from this regulator even if the Raspberry Pi in the high power branch is turned off.

Microcontroller

Because the microcontroller is powered all the time by the battery it must draw very little current if the device is not plugged in to a power supply. The choice fell on a PIC18LF25K22. It draws a minimal current of only 20nA in sleep mode. Further, there is an existing Execution Framework written in C (HEVS Picco-XF) that works well on this PIC and if implemented, simplifies many software development steps.

Voltage measure

The voltage decrease of Li-Ion batteries is almost proportional to the charge. This behaviour can be used to simply calculate an approximate charge-percentage. This percentage is used to show a battery symbol on the display. A voltage divider is powered by the battery allows measuring the voltage. It is designed to output 3V if the battery fully charged is at 4.2V. Because the microcontroller is powered by 3V, the reference voltage for analogue-digital conversion is also at 3V. A full battery will therefore result in the highest conversion value.

The voltage measure is also used for safety reasons. The Li-Ion battery must not be deeply discharged, as this will damage the battery. If the voltage drops under 3V the microcontroller will send an emergency signal to the Raspberry Pi to indicate that it must be shut down and recharged.

Power-Manager Device Operation

The result of the electronics and the microcontroller's software is a power manager device for the Raspberry Pi. Its operation is described in this chapter based on the following, common scenario:

The Raspberry Pi is turned off and the power supply is disconnected, the microcontroller waits in sleep mode until the power button is pressed. As soon it receives the power button interruption it will turn on the power switch and the DC/DC converter begins to work and power the Raspberry Pi. From now, the microcontroller takes periodically samples of the battery voltage to calculate an estimated charge percentage. The Raspberry Pi is going to read these samples from time to time via UART. The device can be forced to turn off by pressing the button for 5 seconds. This should normally not be used because the Raspberry Pi will create an interruption that indicates its shutdown process. After a delay the microcontroller will turn off the Raspberry Pi's power switch safely and go to sleep mode. Then a power supply is plugged in to the charger and it will start to recharge the battery, the microcontroller will wake up and indicate the charge state with 3 LEDs.

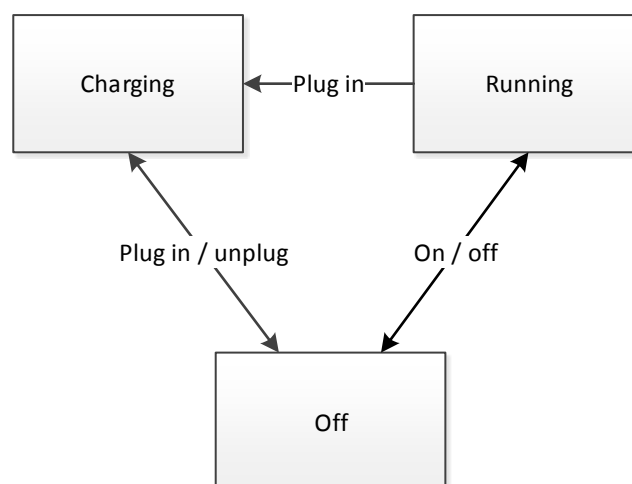


Figure 27 PowerManager state chart

Generally, the device can be in three different states. The transitions from one state to another are based on events that are created either if the Raspberry Pi turns on/off or the power supply is plugged in/out. Each state turns different peripherals of the PowerManager on or off, to consume the less energy. The Raspberry Pi can be regarded as the main peripheral. The different states contain the following actions:

- Off
 - Raspberry Pi Power-Switch turned off
 - ADC turned off
 - Charger Status-Signal pull-up-resistors not powered
 - Indication LEDs turned off
 - UART device turned off
- Running
 - Raspberry Pi Power-Switch turned on
 - ADC turned on & periodic measurement of battery voltage
 - Charger Status-Signal pull-up-resistors not powered
 - Indication LEDs turned off
 - UART device turned on & waiting on request
- Charging
 - Raspberry Pi Power-Switch turned off
 - ADC turned off
 - Charger Status-Signal pull-up-resistors powered & polling status periodically
 - Indication LEDs turned on & showing charge state
 - UART device turned off

Connection to Raspberry Pi

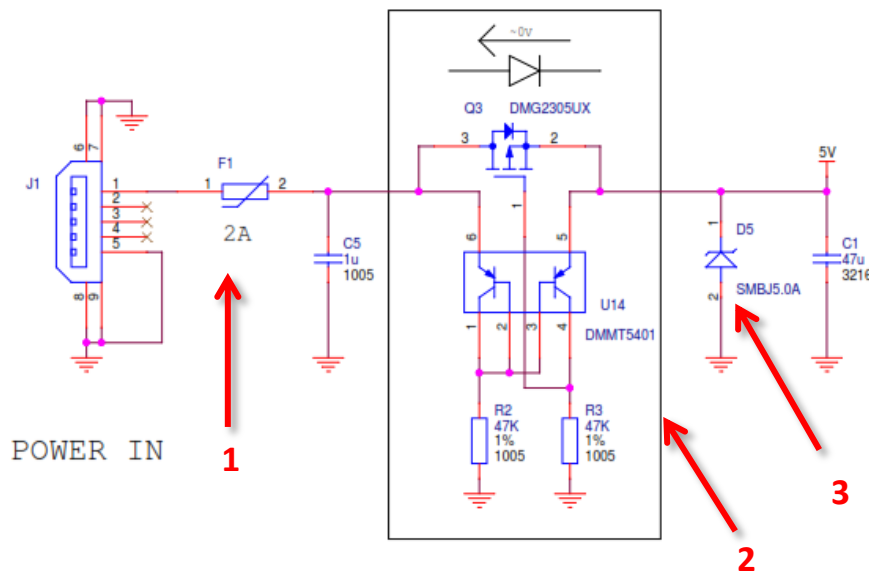


Figure 28 Raspberry Pi power supply protections [1]

1. 2A Fuse
2. Ideal Diode
3. ESD Protection

The power supply is ensured by connecting cables to the 5V pin of the Raspberry Pi's pin header. The 2A fuse and the ideal diode are bypassed. These protections are not important because the maximal current draw was measured and is below 2A. The Raspberry Pi is built into the casing and the USB connector is hidden. This another reason why it does not matter to bypass the fuse and the ideal diode.

Tests

The current draw in the “Off” state was measured to be sure that the device will not discharge significantly while it is not used. Therefore, an ampere meter was placed in the circuit as shown on the picture below.

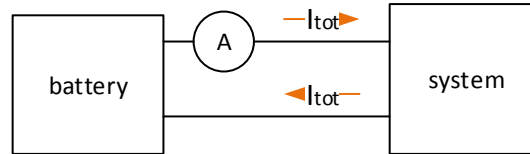


Figure 29 current measurement

The measure showed a current draw of $85\mu A$ during sleep mode. The fully charged battery has a capacity of 2900mAh. Theoretically, the discharge time would be more than 3 years.

The period of use was measured while testing the whole system in the end of the project. A period of use of more than 1h could be achieved.

The time to charge an empty battery is about 3h 30min. The time has been measured multiple times and 3h 30min turned out to be the average.

4.3.3 Camera & Illumination

The camera and the optics are taken from the earlier project. The illumination could also be reused with some modifications. The schematic below on the left describes the lightning electronics and the interface to the Raspberry Pi.

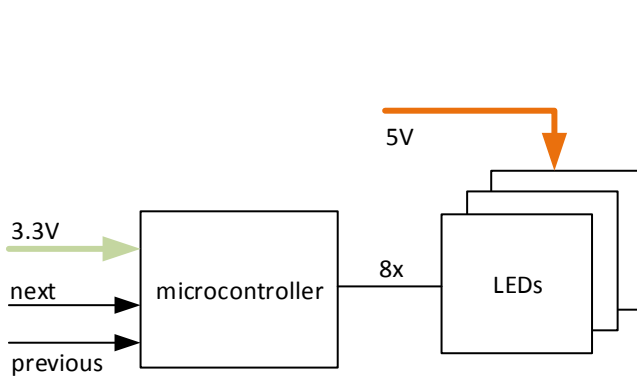


Figure 30 LightManager electronics block diagram

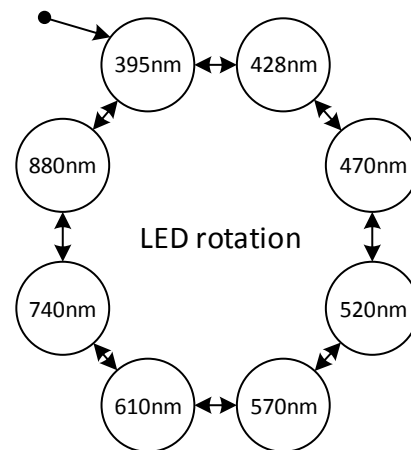


Figure 31 LED rotation

Only one wavelength can be active at the same time. With the next and previous signal, the controller rotates to the next or previous wavelength. The microcontroller draws only little current because the LEDs are powered from the 5V supply. The microcontroller supply pin is connected directly to a GPIO pin of the Raspberry Pi. To turn off the illumination the Raspberry Pi simply turns off the microcontroller. The LEDs cannot be driven directly by the Raspberry Pi because there are too less GPIO pins.

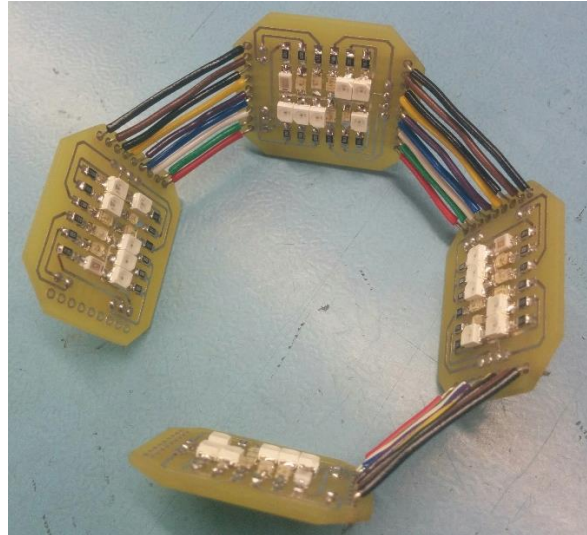


Figure 32 assembly of illumination panels

The illumination is made of 4 panels, each with LEDs of 8 wavelengths. Some wavelengths need multiple LEDs per panel to shine bright enough. The panels fit exactly into the container.



Figure 33 illumination 520nm

The LEDs do not point directly on the skin. The reflections are decreased due to this reason and the illumination is very homogenous.

4.3.4 Casing

The development of the casing was based on the experience of the earlier versions and two important ideas. The first one is that the electronic components should be stacked on top of each other to save space. The second idea was to create a casing with two separate halves. The system can be mounted and tested in one half and closed in the end. The PCB shapes of the light-manager and the power-manager have been fabricated to fit exactly into the casing.

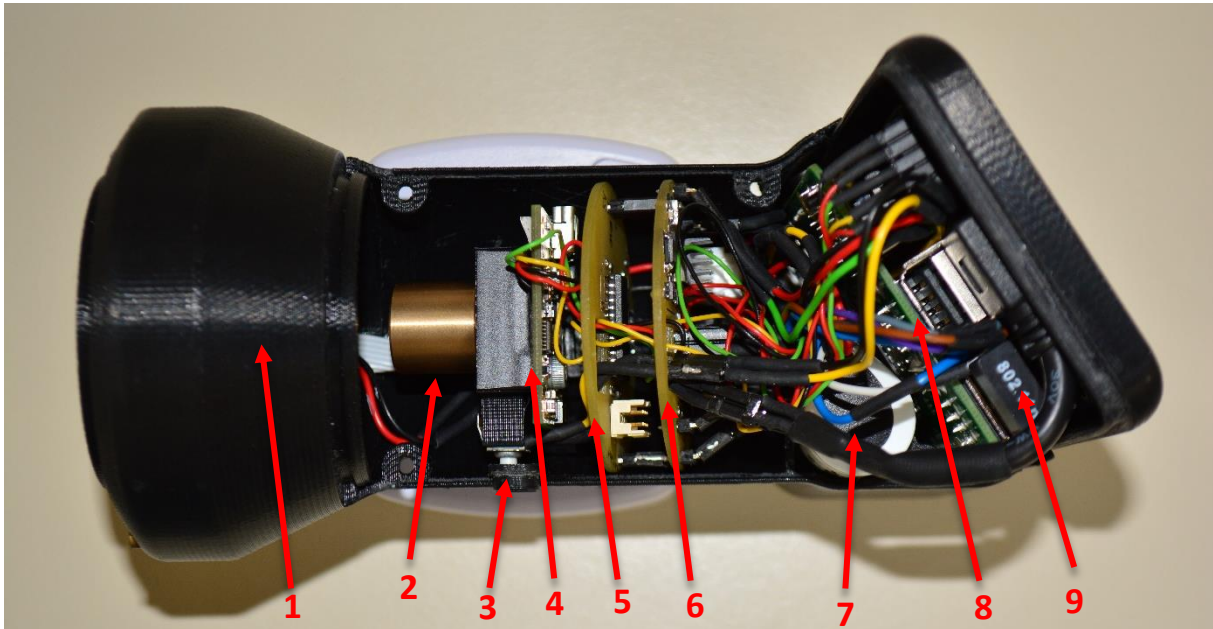


Figure 34 device interior

- | | | |
|---------------------------|--------------------|------------------|
| 1. Illumination container | 2. Adjustable lens | 3. TriggerButton |
| 4. Camera | 5. LightManager | 6. PowerManager |
| 7. Battery | 8. Raspberry Pi | 9. WLAN Dongle |

Two contacts allow putting the camera on a docking station to recharge.



Figure 35 contacts for docking station

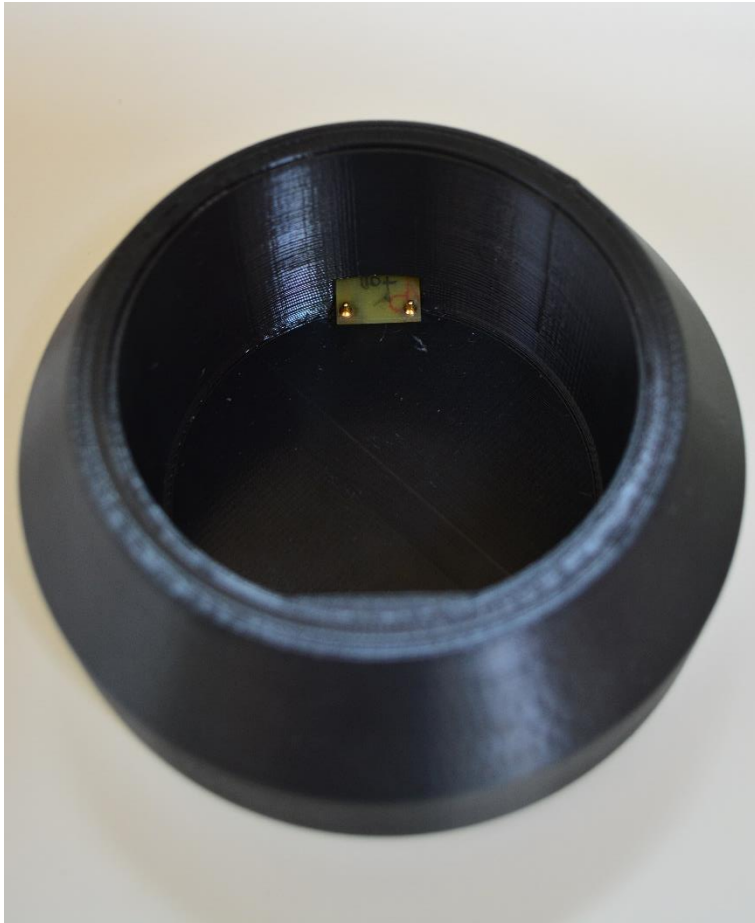


Figure 36 docking station

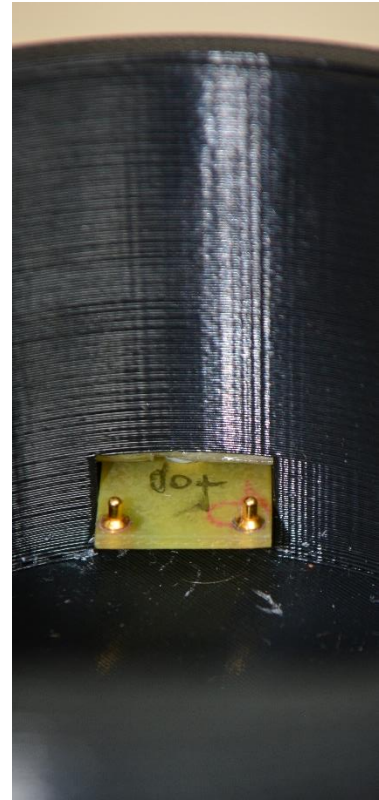


Figure 37 spring-suspended pins

The docking station contains a heavy metallic ring to avoid tipping the camera while it is not in use. The electrical contacts are ensured by two spring-suspended pins that press against the camera. Any usual 5V/1A AC/DC wall adapter with micro-USB plug can be used to power the docking station.



Figure 38 front panel

5 Results

The system was tested in the end of the project on different subjects. It can take multispectral images, analyse them and send the data to a server using WLAN. It implements the doctor’s use case where images in context with data of a patient are uploaded to a web application. The image acquisition can be calibrated and a homogeneity filter can be applied. The melanin analysis was implemented but not tested. An additional RGB image is calculated. The battery keeps the device running for over 1 hour. The recharge time is about 3:30 hours.



Figure 39 Device in use

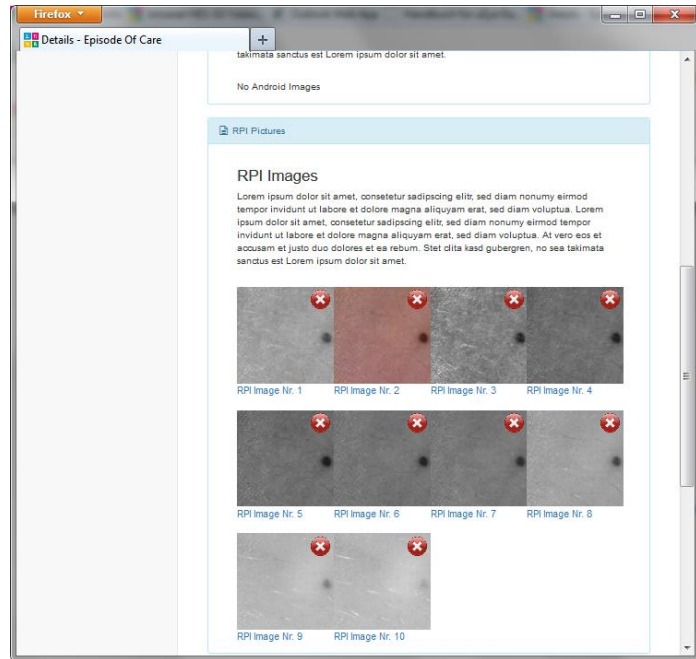


Figure 40 patient data uploaded to web application

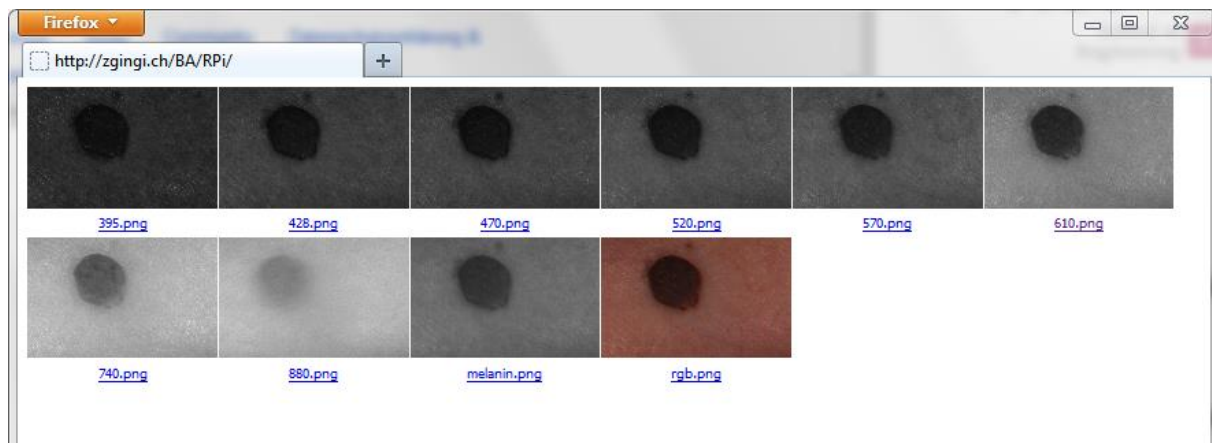


Figure 41 uploaded test images without patient data, 8 images of different wavelengths, 2 images of analysis

6 Future Improvements

- The software does not handle errors if the WLAN connection fails. This should be the first point to work out in the future.
- The method to detect the melanin must be tested.
- At the moment the wireless configuration must be done manually in the linux console. There should be implemented an easier way to connect the device to a new wireless network.
- The assembly of the system in the casing is very difficult due to little space. The fixations for the PCBs and the camera must be improved.
- It works best if the Raspberry Pi is programmed via a network cable. To be able to connect this cable to the assembled system a port in the casing should be added which can be closed with a rubber stopper.
- The assembly of the illumination panels is a nasty work. The illumination should be fabricated using flexible or semi-flexible PCBs.



Figure 42 Euro Circuits semi-flexible PCBs [1]

7 Conclusion

The development of the software took a big part of the time. To get to know the camera application interface took much time because of the vast amount of functions and settings. Luckily, the documentation is very well done and helped a lot. The camera driver runs stable on the Raspberry Pi. The Raspberry Pi itself has a great community, therefore a lot of information about configuration can be found in the internet. I became a real Raspberry Pi fan during this project not least because Qt works like a charm on it. The possibility to use Qt was great and solved a lot of problems already at the start. Only the capacitive touchscreen caused some trouble because the coordinates of touch events were not received correctly without a workaround. Perhaps a future update of Qt or the display driver solves the problem. I enjoyed the work of developing a web interface in cooperation with Andreas Imsand because he is a friend of mine in private life.

In the end there was no more time to implement an error handling for the scenario where the WLAN connection fails. The test of the melanin analysis could not be done also as a consequence of time constraints. The algorithm is the same as in an earlier project and because the illumination, camera and optics kept the same it is just a matter of time to get the melanin analysis working properly.

The selection of the proper battery type for this project required quite some time of research into battery technologies. The choice of Li-Ion batteries had the big advantage of already existing and uncomplicated integrated circuits for recharging.

The fabrication of the PCBs and the casing took place in the second half of the project. There were a lot of discussions about the shape of the PCBs and the casing, how to connect the PCBs and where to place them in the casing. Soldering the components on the different PCBs and assembling the system took quite some time and the first power-on of the system was an exciting moment. Unfortunately, the charge & load sharing IC easily overheated if the Raspberry Pi with the camera and illumination were turned on. A cooler was glued on top of this IC with thermal compounds. This did not solve the problem. The electronics had to be reworked and luckily, the power PCB could be modified without fabricating a new one.

The final tests were successful and images could be taken, analyzed and transmitted to the server.

There were ups and downs, successes and failures but in the end it was great experience and I am not likely to change the business.

8 Sources

- [1] Product image of manufacturer
- [2] Battery university: <http://batteryuniversity.com/>
- [3] Supercapacitors: <https://de.wikipedia.org/wiki/Superkondensator>
- [4] UPS: https://de.wikipedia.org/wiki/Unterbrechungsfreie_Stromversorgung
- [5] Bachelor thesis & sources, Michael Schmid, 2014, on CD
- [6] Component datasheets, on CD
- [7] Source code ueye demoprogram , on CD
- [8] Source code HEVS picco-XF, on CD
- [9] Li-Ion additional charging tips: <http://www.powerstream.com/li.htm>
- [10]Raspberry Pi: <https://www.raspberrypi.org/>
- [11]Qt: <http://www.qt.io/developers/>
- [12]uEye documentation, on CD
- [13]Semester project , Nicolas Marty, 2015
- [14]Hardware specifications, semester project [13]

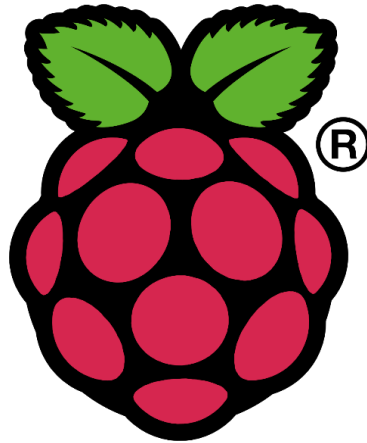
9 Appendix

- [A] Setup Raspberry Pi
- [B] Pin Header Layout
- [C] Materials List
- [D] Schematics & PCB PowerManager
- [E] Schematics & PCB LightManager
- [F] Schematics & PCB illumination panel, indication LEDs, USB adapter
- [G] Source code qt main application
- [H] Source code PowerManager
- [I] Source code LightManager

10 Signature

Nicolas Marty

Appendix A – Setup Guide Raspberry Pi and Development Environment



V2.0

Content

Introduction	3
Raspberry Pi 2 Setup	3
Raspbian OS.....	3
Adafruit 2.8" PiTFT+ Capacitive Touchscreen (Product-ID #2423).....	4
uEye Camera.....	6
WLAN.....	6
Power Saving	7
Installing the Splashscreen	7
Automatic Login & Application Start.....	9
Tweak Performance	9
Setup Development Environment	9
Ubuntu Virtual Machine.....	9
Setup Cross Compilation	10
Configure Qt Creator	11
References	12

Introduction

In order to use a Raspberry Pi for my bachelor thesis there are a couple of things to install and configure before the Qt Application can be executed. This document describes step-by-step the Raspberry Pi 2 Setup and the QtCreator configuration to cross-compile on Raspberry Pi for this specific project.

Raspberry Pi 2 Setup

Raspbian OS

Prerequisites

- You need a Windows PC, MicroSD-Card, CardReader and of course a Raspberry Pi B+.
- Download and unzip newest Raspbian Image from <http://www.raspberrypi.org/downloads/>
- Download and install newest Win32 Disk Imager from <http://sourceforge.net/projects/win32diskimager/>

Installation

Write Image to SD-Card

1. Connect MicroSD-Card to PC.
2. Start Win32 Disk Imager.
3. Select raspbian image from your download directory, e.g. ./2015-02-16-raspbian-wheezy.img
4. Select proper device (SD-Card drive letter), e.g. [J:\
5. Write image to SD-Card using the “write” button, this process takes a couple of minutes.

Booting the Raspberry Pi for the first time

1. Insert MicroSD-Card into RPi.
2. Make sure a monitor and a keyboard is connected. (SSH can be used also, raspi-config must be started manually)
3. If everything is ok, system should open the raspi-config.
4. Select the following options:
 - a. Expand Filesystem > ok
 - b. Internationalisation Options > Change Keyboard Layout > Generic 105-key (Intl) PC > *desired language* > ...keep default options... > ok
 - c. Advanced Options > SPI > yes > ok > yes > ok (enable SPI on startup, used by PiTFT)
 - d. Advanced Options > Hostname > Enter a unique hostname
 - e. Finish, reboot.
5. After reboot, login with user pi and update the device with following command: `sudo apt-get update` then `sudo apt-get upgrade` and last but not least `sudo rpi-update`
6. Set the password for the root user. Root is used later to execute wiringPi. Enter following commands: `sudo -i` and then `passwd`, set password to 123456
7. Now the initial config is done, time to backup. Shutdown system with following command: `sudo shutdown -h now`

Create Backup Image

1. Unplug SD-Card from RPi and connect it to PC.
2. Start Win32 DiskManager.
3. Select a destination and filename for the backup, e.g. ./bkp_raspian_raspi-configured.img
4. Select proper device (SD-Card drive letter), e.g. [J:\
5. Backup SD-Card Image using “read” button, this process takes a couple of minutes.

Adafruit 2.8" PiTFT+ Capacitive Touchscreen (Product-ID #2423)

Prerequisites

- In addition to *Raspbian OS* prerequisites you need a network cable with connection to the internet.
- Mount PiTFT on RPi 2.
- Optional: Download PuTTY (used for SSH remote connection)
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
- The original adafruit driver is currently not working, installing it results in RPi not booting anymore.

Manual Installation – OLD (see chapter Easy Install)

Kernel Update

1. Execute following command:
`sudo rpi-update`
 This will download and install the newest kernel + drivers. This update contains already all drivers who make the PiTFT working.
2. After the installation reboot the RPi2 with
`sudo reboot`

Setting up the display

1. Open `/boot/config.txt` with following command :
`sudo nano /boot/config.txt`
2. Add these lines at the end of the file:
`dtparam=spi=on`
`dtoverlay=pitft28-resistive`
 These lines give parameters to the already installed display driver
3. To show the graphical XWindows interface on the PiTFT, the correct framebuffer must be selected. In order to do this, open the `99-fbturbo.conf` file with the command:
`sudo nano /usr/share/X11/xorg.conf.d/99-fbturbo.conf`
4. Change `/dev/fb0` to `/dev/fb1`
 PiTFT is now selected.

Setting up the touchscreen

5. Open `/etc/modules` with :
`sudo nano /etc/modules`
6. Add this line at the end of the file:
`stmpe-ts`
 This will load the kernel module (driver) for the touchscreen
7. Create a devicemanager rule file for the touchscreen.
`sudo nano /etc/udev/rules.d/95-stmpe.rules`
8. Because the touchscreen event nr can change if keyboard and mouse are attached, it must be mapped to a constant event name, called touchscreen. Add the following line to the file and `save&exit`.
`SUBSYSTEM=="input", ATTRS{name}=="stmpe-ts", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"`
9. Remove and reinstall touchscreen module
`sudo rmmod stmpe_ts; sudo modprobe stmpe_ts`
10. Test if the mapping worked with: (should look like `/dev/input/touchscreen -> eventX`)
`ls -l /dev/input/touchscreen`

11. Download calibration tools:

```
sudo apt-get install evtest tslib libts-bin
```

12. To view touchscreen events in realtime execute:

```
sudo evtest /dev/input/touchscreen
```

13. Run the calibration with:

```
sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen  
ts_calibrate
```

Activate console on PiTFT

14. Execute command:

```
sudo nano /boot/cmdline.txt
```

and add `fbcon=map:10` at the end of the line

Rotate display

1. To rotate the display modify the boot config file

```
sudo nano /boot/config.txt
```

2. Modify the line `dtoverlay`, the parameter `rotate` changes the orientation

```
dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20
```

Possible values are:

- 0, portrait, with the bottom near the USB jacks
- 90, landscape, with the bottom near the headphone jack
- 180 portrait, with the top near the USB jacks
- 270 landscape, with the top near the headphone jack

Easy install – USE THIS

1. Add adafruit repository to the system

```
curl -SLs https://apt.adafruit.com/add | sudo bash
```

```
sudo apt-get install raspberrypi-bootloader
```

```
sudo apt-get install -y adafruit-pitft-helper
```

2. Start installation script

```
sudo adafruit-pitft-helper -t 28c
```

3. Would you like the console to appear on the PiTFT display? **n**

4. Would you like GPIO #23 to act as on/off button? **n**

5. Activate display to show graphical interface:

```
export FRAMEBUFFER=/dev/fb1
```

6. If the graphical interface is to be started via ssh, the following environment variable must be set:

```
export DISPLAY=:1
```

7. Reboot, login and execute `startx` to start the graphical interface.

8. Display and touch screen should work.

9. (Optional) calibrate display: see section above

Turn display on/off

15. Create device link with command:

```
sudo sh -c "echo 508 > /sys/class/gpio/export"
```

16. Set GPIO#508 as output with:

```
sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"
```

17. Turn on/off with commands:

```
sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
```

```
sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"
```

XWindows error message fix

When starting XWindows, there will probably appear an ugly error message. This is because of a bug in the actual Raspbian Image for RPi2. To get rid of this message open a terminal (in XWindows) and execute:

```
lxsession-edit
```

Uncheck LXPoKit in the dialog box and reboot RPi.

uEye Camera

Prerequisites

- Copy uEyeSDK-[Versionsnummer]-ARM_LINUX_IDS_[Setuptyp].tar on a usb stick and mount it on RPi 2.

Installation

1. Mount usb drive with following commands:

```
sudo mkdir /media/usbdrive  
sudo mount /dev/sda1 /media/usbdrive
```
2. Copy files to your home directory:

```
cp uEyeSDK-[Versionsnummer]-ARM_LINUX_IDS_[Setuptyp].tar ~
```
3. Unpack archive on target system using:

```
sudo tar xvf uEyeSDK-[Versionsnummer]-ARM_LINUX_IDS_[Setuptyp].tar -C /
```
4. Execute Setup-Script:

```
sudo /usr/local/share/ueye/bin/ueyesdk-setup.sh
```
5. unmount usb drive with these commands:

```
sudo umount /media/usbdrive  
sudo rmdir /media/usbdrive
```
6. To make the demoprogramm work finally, the libqt3support library must be installed with following command (do not install until you really want to test the demoprogram):

```
sudo apt-get install libqt4-qt3support
```
7. Start demoprogram with the command:

```
sudo ueyecameramanager
```

the demoprogram must be started while running XWindows (`startx`)

WLAN

Prerequisites

Plugin WLAN Dongle

Configuration

1. Get a list of the available wireless networks using the following command:

```
sudo iwlist wlan0 scan
```
2. Open the wpa_supplicant.conf file with the following command, this file contains the connection information to your desired WLAN:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```
3. Add the following at the end of the file:
If you are using WPA-PSK (e.g. home network) add these lines:

```
network={  
    ssid="YourSSID"  
    psk="password"  
    key_mgmt=WPA-PSK  
    id_str="home"  
}
```

If you are using WPA-EAP (e.g. HEVS enterprise network) add these lines:

```
network={
    ssid="secure-hevs"
    proto=RSN
    key_mgmt=WPA-EAP
    pairwise=CCMP
    auth_alg=OPEN
    eap=PEAP
    identity="username"
    password="password"
    id_str="enterprise"
}
```

4. If you are suffering problems enter the following command to get a debug message:

```
sudo wpa_supplicant -i wlan0 -D wext -c
/etc/wpa_supplicant/wpa_supplicant.conf -d
```

i

Make sure the RPi is connected and received an IP-Address. Use this command to show the network interface information and look at interface wlan0:

```
ifconfig
```

Power Saving

The Raspberry Pi does not have a sleep mode. There is just the possibility to suspend the monitor. The suspend time can be set with the following command: (it will be reset after a reboot, this function has not been used and serves only as information)

```
setterm -blank [minutes]
```

Installing the Splashscreen

To hide all the "scary" linux startup outputs from the end user a nice splashscreen must be displayed.

1. First of all, install fbi: `sudo apt-get install fbi`
2. Copy your custom splash image to /etc/ and name it "splash.png".
Example command: `rsync -avz ~/splash.png root@rpi2:/etc/splash.png`
3. Create a script called asplashscreen in /etc/init.d/. Enter the following code:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          asplashscreen
# Required-Start:
# Required-Stop:
# Should-Start:
# Default-Start:    S
# Default-Stop:
# Short-Description: Show custom splashscreen
# Description:       Show custom splashscreen
### END INIT INFO

do_start () {

    /usr/bin/fbi -T 1 -noverbose -a /etc/splash.png
    exit 0
}

case "$1" in
```

```

start|")
do_start
;;
restart|reload|force-reload)
echo "Error: argument '$1' not supported" >&2
exit 3
;;
stop)
# No-op
;;
status)
exit 0
;;
*)
echo "Usage: asplashscreen [start|stop]" >&2
exit 3
;;
esac

```

- :
- Then make that script executable and install it for init mode rcS:


```

chmod a+x /etc/init.d/asplashscreen
sudo inserv /etc/init.d/asplashscreen

```
 - The splashscreen will not appear if the device is rebooted now. That's because the startup is displayed on the framebuffer fb0 from the HDMI output. And not the fb1 from the PiTFT. The solution for this is to install a framebuffer mirroring tool. Install fbcv as follows:


```

sudo apt-get install cmake
git clone https://github.com/tasanakorn/rpi-fbcv
cd rpi-fbcv/
mkdir build
cd build/
cmake ..
make
sudo install fbcv /usr/local/bin/fbcv

```
 - Open the asplashscreen file again and go to the line just after `do_start(){`
 - Enter the following expression: `/usr/local/bin/fbcv &`
This will start the framebuffer mirroring tool. The "&" makes it run in the background.
 - The fb0 output must be adjusted to match the settings of the fb1. Open `/boot/config.txt` and search for the commented lines:


```

#hdmi_group=1
#hdmi_mode=1

```

 Change these lines and add the following lines right below, in the end it must look like this:


```

hdmi_group=2
hdmi_mode=87
hdmi_force_hotplug=1
hdmi_cvt=320 240 60 1 0 0 0

```

Automatic Login & Application Start

9. Create a bash script that contains all commands to execute the application:

```
sudo nano /home/pi/app.sh
```

10. The script must look like:

```
#!/bin/bash
```

```
#set environment variables
```

```
export QT_QPA_EGLFS_PHYSICAL_HEIGHT=60
```

```
export QT_QPA_EGLFS_PHYSICAL_WIDTH=85
```

```
#execute with sudo (due to wiringPi)
```

```
#1 = turn off powersupply on exit, 0 = keep device running on exit
```

```
sudo ./helloworld/helloworld 1
```

```
sudo shutdown -h now
```

11. To make this script executed automatically, there must be added a line in the file /etc/profile. Open this file with `sudo nano /etc/profile` and add the following line at the end:

```
./home/pi/app.sh
```

12. Now, user pi must be logged in automatically. Open the inittab file with: `sudo nano /etc/inittab`

13. Comment out the line `1:2345:respawn:/sbin/getty 115200 tty1` by adding a `#` in front of it.

14. Enter the following expression just below the commented line.

```
1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
```

Tweak Performance

1. At first the clock frequencies will be raised to get some more calculating-power out of the Pi. Open Raspberry config with: `sudo raspi-config`

Overclock > Ok > Pi2 > Enter > Finish

This is not really an overclocking configuration as it is named Pi2. With the default setting "None" the Pi2 is running "underclocked". 700MHz is 200Mhz below the nominal clock for the Pi2.

2. Now a very important setting that will rise the frames per second of the PiTFT very much. With this setting the camera preview is displayed very smooth and without any lags.

Execute `sudo nano /boot/config.txt` and search for the line `dtoverlay`

Modify speed and fps as follows: `speed=80000000, fps=60`. As the display is positioned close to RPi, the SPI lines are short and able to work at this high speed.

Setup Development Environment

Ubuntu Virtual Machine

The use of a virtual machine gives lot of advantages. A virtual machine is portable and can be executed on every computer. Furthermore the virtual machine is dedicated to the Development Environment and is not influenced by other software. Backups are easily made with system snapshots, these help a lot during the different setup steps.

- To create a Virtual Machine, VMWare Workstation and an Ubuntu image is necessary. In this guide a trial version of VMWare Workstation and Ubuntu-14.04.2-desktop-amd64.iso image are used. The virtual machine can be used with the free VMWare Player afterwards.
- Create a Virtual disk of at least 40GB to store all necessary data later. (Resizing disk afterwards is not a good idea)
- Enter a username and password and let the installation run.
- As soon as Ubuntu is running download all available updates.
- The keyboard layout can be changed in Text Entry Settings by adding a new Input Source with the tiny little “+” button at the bottom.

Install Qt Creator

Let's start easy: Go to Ubuntu Software Center and install QtCreator, uncheck every additional add-on. Done.

Install 32bit libraries

As this is a 64 bit Ubuntu and raspberry pi is a 32 bit system the compiler needs additional libraries. In particular the ia32-libs. It is not available officially on Ubuntu 14.04. But it can still be installed using the following commands:

```
sudo -i
cd /etc/apt/sources.list.d
echo "deb http://old-releases.ubuntu.com/ubuntu/ raring main restricted
universe multiverse" >ia32-libs-raring.list
apt-get update
apt-get install ia32-libs
```

Setup Cross Compilation

Prerequisites

This setup requires quite a lot of stuff to download. To store everything create a folder named x-compilation in your home directory.

Download the GCC Linaro 4.7 Raspberry Pi Compiler. Look for a package called:

```
gcc-4.7-linaro-rpi-gnueabihf.tbz
```

Extract it into the prepared x-compilation directory. It should look like ~/x-compilation/ gcc-4.7-linaro-rpi-gnueabihf/...

Next download the cross-compile-tools set. This is a set of configuration scripts. In particular the fixQualifiedLibraryPaths script will be used to create a link to the libraries on the raspbian image.

At last download the QT Source Code. It can be easily found on the official qt download page. (Look out for QT Source).

Create and mount Raspbian Image

Take the sd card from the Raspberry Pi. Make sure every driver is installed as mentioned in the RPi setup.

To create an sd-card image double click the `backup_rpi_sdcard` script and select “Run in terminal” The process takes a couple of minutes. To write back an image to the sdcard run `write_rpi_sdcard`.

Use the predefined script to mount the image. Double click `mount_rpi_root-fs` and select “Run in terminal”. Enter your password. To unmount the filesystem execute the `unmount_rpi_root-fs` script.

Modify links

Before compilation can be done execute following commands:

```
cd ~/x-compilation/cross-compile-tools
sudo ./fixQualifiedLibraryPaths ~/x-compilation/rpi_root-fs/ ~/x-
compilation/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-gnueabihf-gcc
```

Compile QT for Raspberry Pi

First, compile qtbase. Therefore execute these commands:

```
cd ~/x-compilation/qt-everywhere-opensource-src-5.4.1/qtbase
```

then:

```
./configure -opengl es2 -device linux-rasp-pi-g++ -device-option
CROSS_COMPILE=~ /x-compilation/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-
gnueabihf- -sysroot ~/x-compilation/rpi_root-fs -opensource -confirm-
license -optimized-qmake -reduce-exports -release -make libs -prefix
/usr/local/qt5pi
```

then: (time to go and drink a coffee)

```
make -j 4
```

and the last one:

```
sudo make install
```

The library qtdeclarative is also needed. To compile this library, execute:

```
cd ~/x-compilation/qt-everywhere-opensource-src-5.4.1/qtdeclarative
/usr/local/qt5pi/bin/qmake .
```

then: (time to go and drink a coffee)

```
make -j 4
```

and the last one:

```
sudo make install
```

The compiled files are now locally stored on the mounted rpi_root-fs. Copy the files to the physical RPi with the following command:

```
rsync -avz /home/username/x-compilation/rpi_root-fs/usr/local/qt5pi/
root@rpi2-xx:/usr/local/qt5pi/
```

Configure Qt Creator

Create Raspberry Pi Kit

In Qt Creator open *Tools > Options...*

Select *Build & Run* and open tab *Compilers*

Add > GCC

Name: ARM GCC

Compiler path: ~/x-compilation/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-gnueabihf-g++

ABI: arm-linux-generic-elf-32bit

Open tab *Debuggers*

Add

Name: ARM GDB

Path: /home/nicolas/x-compilation/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-gnueabihf-gdb

Open tab *Qt Versions*

Click *Add...* and select the proper qmake (/usr/local/qt5pi/bin/qmake)

Open tab *Kits*

Add

Name: Raspberry Pi
Device type: Generic Linux Device
Sysroot: /home/nicolas/x-compilation/rpi_root-fs
Compiler: ARM GCC
Debugger ARM GDB
Qt version: Qt 5.4.1 (qt5pi)

Select *Devices*

Add... > Generic Linux Device > Start Wizard

The name...: Raspberry Pi

The devices' host name: raspberrypi

The user name: root

The authentication type: Password

The user's password: 123456

(ssh port: 22)

To avoid problems with future password changes a RSA authentication can be created:

Private key file: *Create New...* > *RSA* > *1024* > *Generate And Save Key Pair* > *Do Not Encrypt Key File*
Deploy public key... > *select file* > *ok*

Authentication type: Key

Configure .pro file

Copy the relevant lines for cross compiling from the helloworld .pro file to your project.

Build & Run Configuration

Under Project > Build & Run > Raspberry Pi Kit > run

Add Deploy Step > Run custom remote command > `rm -r -f ~/helloworld`

Position this deploy step as first step

Add execution arguments: `-platform linuxfb:fb=/dev/fb1`

Add runtime variables:

`DISPLAY=:1`

`QT_QPA_EGLFS_PHYSICALHEIGHT=35`

`QT_QPA_EGLFS_PHYSICAL_WIDTH=50`

`QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS=rotate=90`

`QT_QPA_GENERIC_PLUGINS=evdevtouch:/dev/input/touchscreen`

For further information see the configuration from helloworld project.

References

- <https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/software-installation>
- <http://www.raspberrypi.org/forums/viewtopic.php?f=66&t=103281&p=714737&hilit=stmpe#p714737>
- <http://www.circuitbasics.com/setup-lcd-touchscreen-raspberry-pi/>
- https://wiki.qt.io/Raspberrypi_beginners_guide
- <http://doc.qt.io/qt-5/embedded-linux.html>

- <http://www.opentechguides.com/how-to/article/raspberry-pi/5/raspberry-pi-auto-start.html>
- <https://learn.adafruit.com/running-opengl-based-games-and-emulators-on-adafruit-pitft-displays/pitft-setup>
- <http://www.edv-huber.com/index.php/problemloesungen/15-custom-splash-screen-for-raspberry-pi-raspbian>

Appendix B – Raspberry Pi Pin Header Layout

Color	Function	Header		Function	Color
-	3V3	1	2	5V	Red
-	SDA	3	4	5V	Red
-	SCL	5	6	GND	Black
Blue	RPI_OFF	7	8	TX	Violet
Black	GND	9	10	RX	Brown
Grey	PIC_LOW_BAT	11	12	TRIGGER	Yellow
Blue	NEXT_LIGHT	13	14	GND	-
white	PREVIOUS_LIGHT	15	16	POWER_LIGHT	Green

Appendix C - Materials List

Personne concernée: **Nicolas Marty**
 Mandat no.: **39200** Chef de projet / professeur: **Martial Geiser**
 Salle: **BR05** Délai désiré **29.05.2015**

Farnell			Monnaie	CHF
Quantity	Reference	Designation	Unit Price	
4	1642489	battery management / MCP73871-2CCI/I	1.95	7.80
2	1900168	battery / PA-LN19	53.60	107.20
10	1516290	battery connector / S3B-XH-A (LF)(SN)	0.26	2.55
5	2117334	prec. Resistance / RP73PF2A41K2BTDF	0.38	1.90
5	2117292	prec. Resistance / RP73PF2A16K5BTDF	0.37	1.86
4	2450179	boost converter / TPS61232DRCT	4.40	17.60
4	2289051	inductance / XAL4020-102MEB	3.45	13.80
4	1815777	linear regulator / TPS78330DDCT	0.65	2.58
4	1840997	low power µC / PIC18LF25K22-I/SO	2.40	9.60
4	1078228	levelshifter / ADG3304BRUZ	3.65	14.60
4	2300437	Micro-USB connector / ZX62D-B-5P8	2.85	11.40
15	1655532	LED 880 nm / OIS 330 880	1.75	26.25
15	1890328	LED 740 nm / OIS-330-740-X-T	2.40	36.00
15	8530009	LED 610 nm / KP-3216SEC	0.24	3.65
40	2322108	LED 570 nm / 150141VS73100	0.42	16.80
40	1716766	LED 520 nm / OVS-0804	0.24	9.68
20	8529965	LED 470 nm / KP-3216PBC-A	3.10	62.00
15	1890331	LED 395 nm / OCU-400UE390-X-T	10.80	162.00
3	1125349	MOLEX 500075-1517 MINI USB 2.0 BUC	5.15	15.45
10	1516277	B3B-XH-A (LF)(SN) - JST Buchse	0.22	2.17
				524.89

Distrelec			Monnaie	CHF
Quantity	Reference	Designation	Unit Price	
15	633092	LED 428 nm / 67-21UBC/TR8	0.51	7.65
4	612106	p-channel MOSFET / IRF7404TRPBF	2.05	8.20
3	116795	Stiftleiste Minitex 2x3pol 2 mm	1.50	4.50
				20.35

Pi-Shop.ch			Monnaie	CHF
Hohenklingenstrasse 8 8049 Zürich Telefon: 044 508 50 77 (Mo-Fr 8.00-12.00 / 13.30-17.00) Telefon: 044 508 50 77 (Mo-Fr 8.00-12.00 / 13.30-17.00) Web: www.pi-shop.ch E-Mail: info (at) pi-shop.ch				
Quantity	Reference	Designation	Unit Price	
2	PI-RAS-PI2B	Raspberry Pi 2 Model B	45.90	91.80
2	PI-ADA-814	Miniature WiFi	18.90	37.80
2	PI-RAS-NOOBS	8GB MicroSD Karte	14.90	29.80
2	PI-ADA-2423	PI TFT Plus 320x240 2.8" + Capacitive	54.9	109.80
				269.20

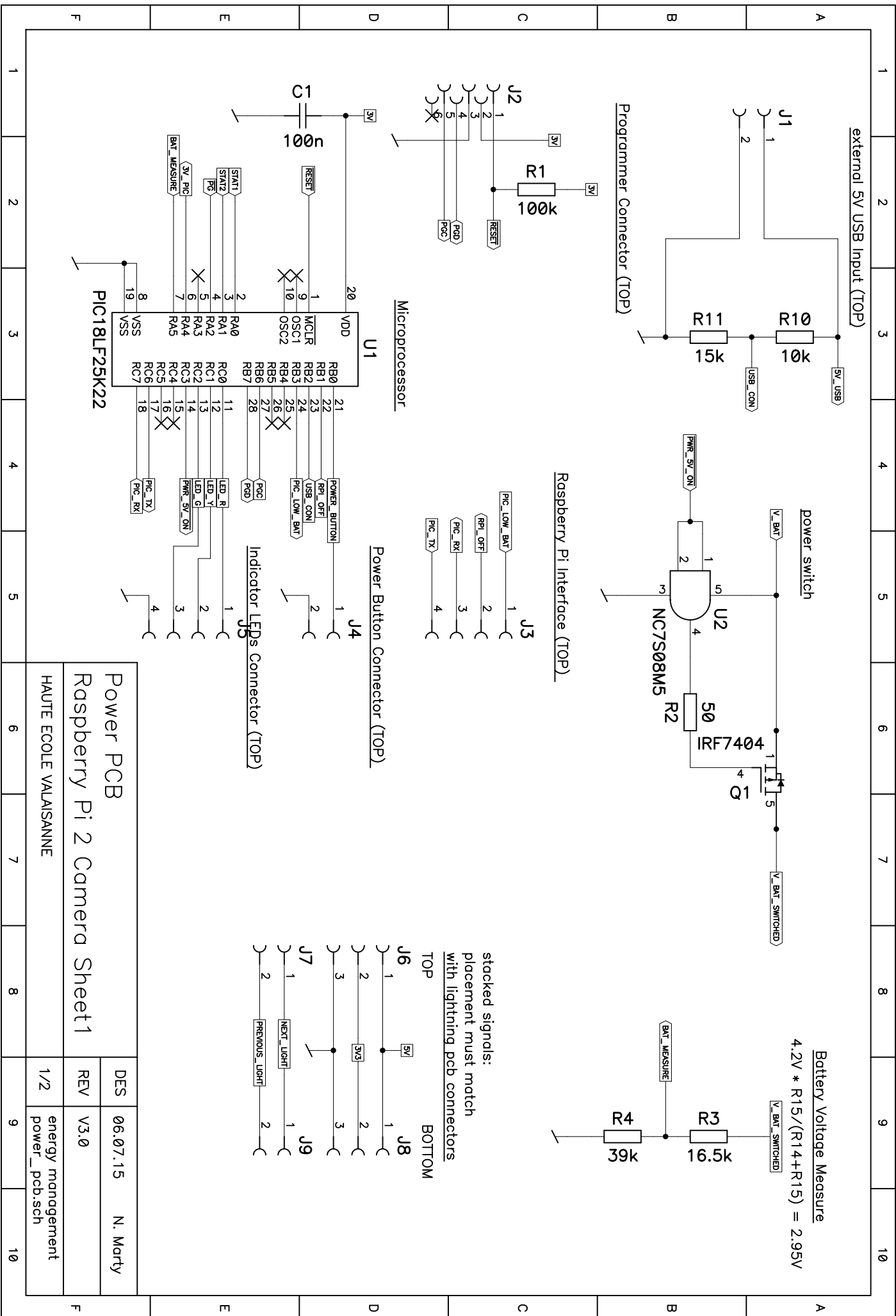
IDS Imaging Development Systems GmbH			Monnaie	CHF
---	--	--	---------	-----

Dimbacher Strasse 6-8, 74182 Obersulm, Deutschland
Tel.: +49 7134 96196 – 0
Fax: +49 7134 96196 – 99
Email: info@ids-imaging.com

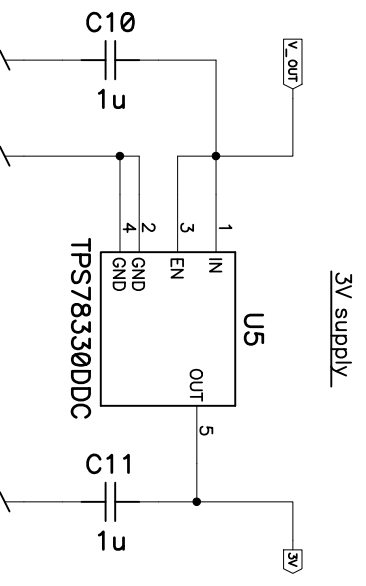
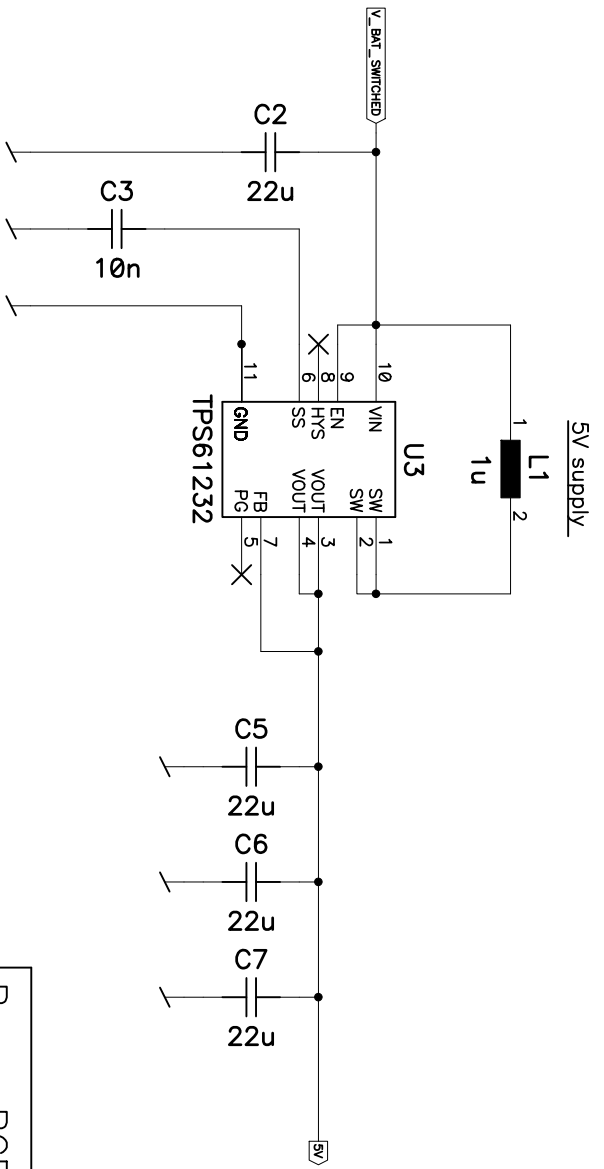
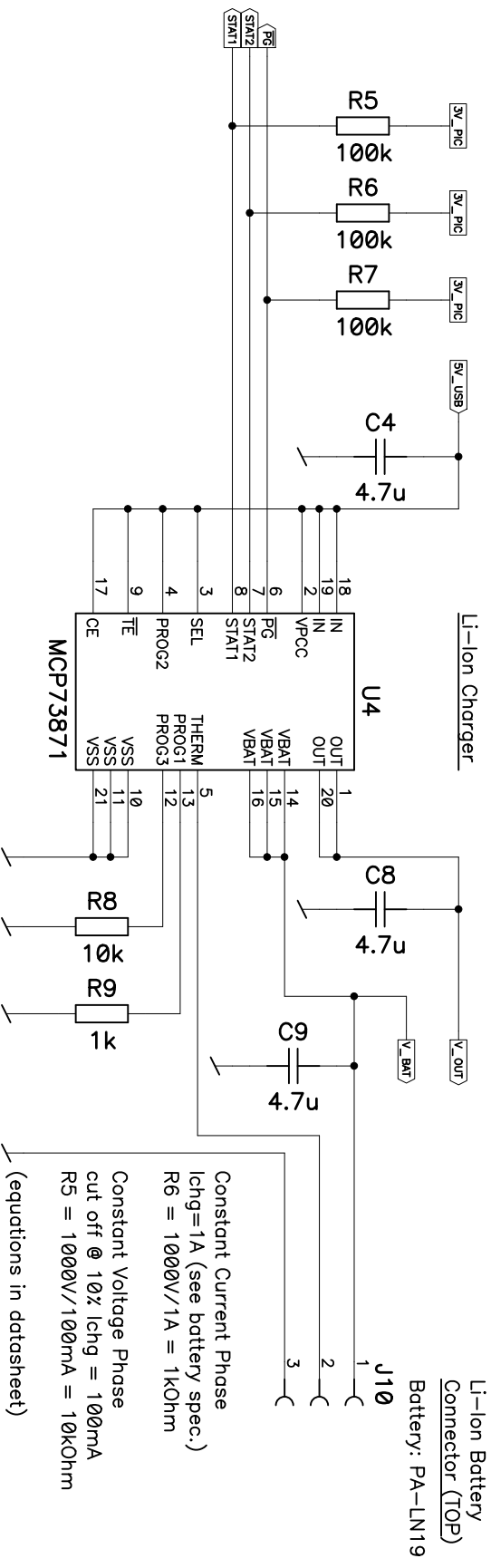
Quantity	Reference	Designation	Unit Price	
2	UI-1222LE-M	CMOS camera	280.00	560.00
				560.00

DR. ZELLMER GMBH			Monnaie	EUR
Im Erlengrund 29				
D-53757 Sankt Augustin				
Telefon: +49-(0)2241-332224				
Telefax: +49-(0)2241-345461				
eMail: info@partsdata.de				
Quantity	Reference	Designation	Unit Price	
3	CU-B06-01R	USB Kabel A-Mini B rechts gewinkelt 0.1i	9.52	28.56
				28.56

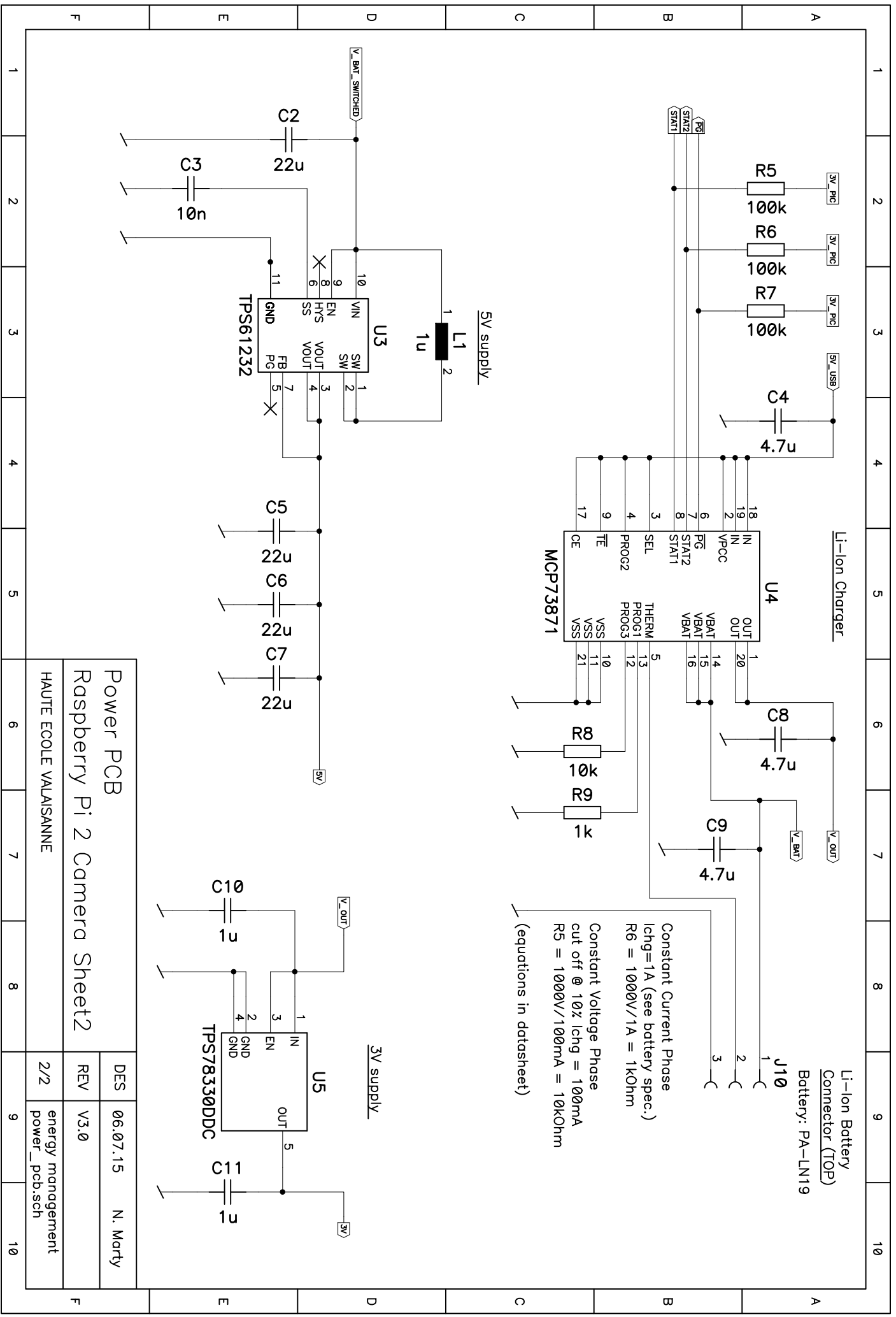
Total (€~SFr.) 1403.00

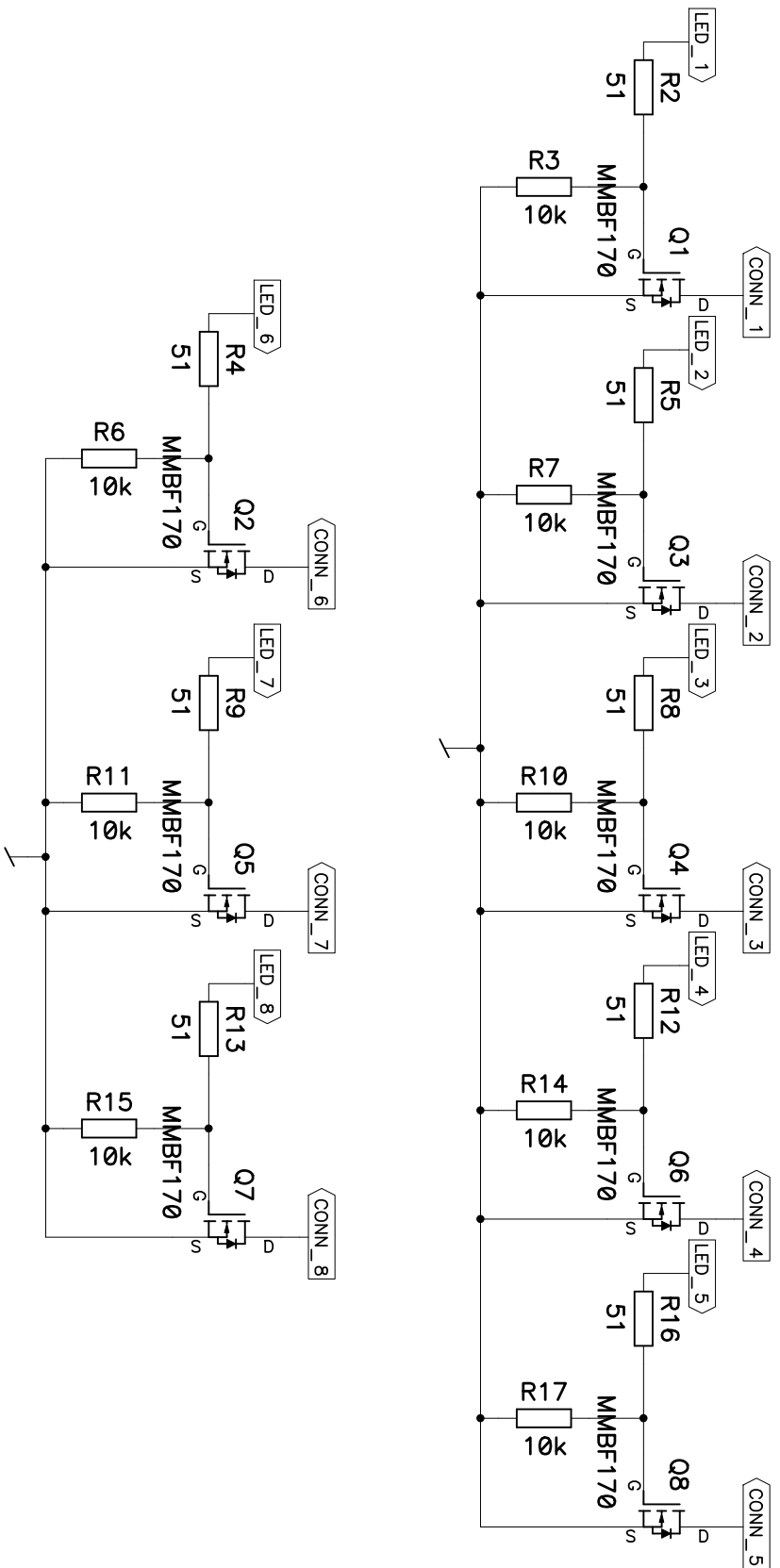


Power PCB		DES	06.07.15	N. Marty
Raspberry Pi 2 Camera Sheet1		REV	V3.0	
HAUTE ECOLE VALAISANNE		1/2	energy management	power_pcb.sch

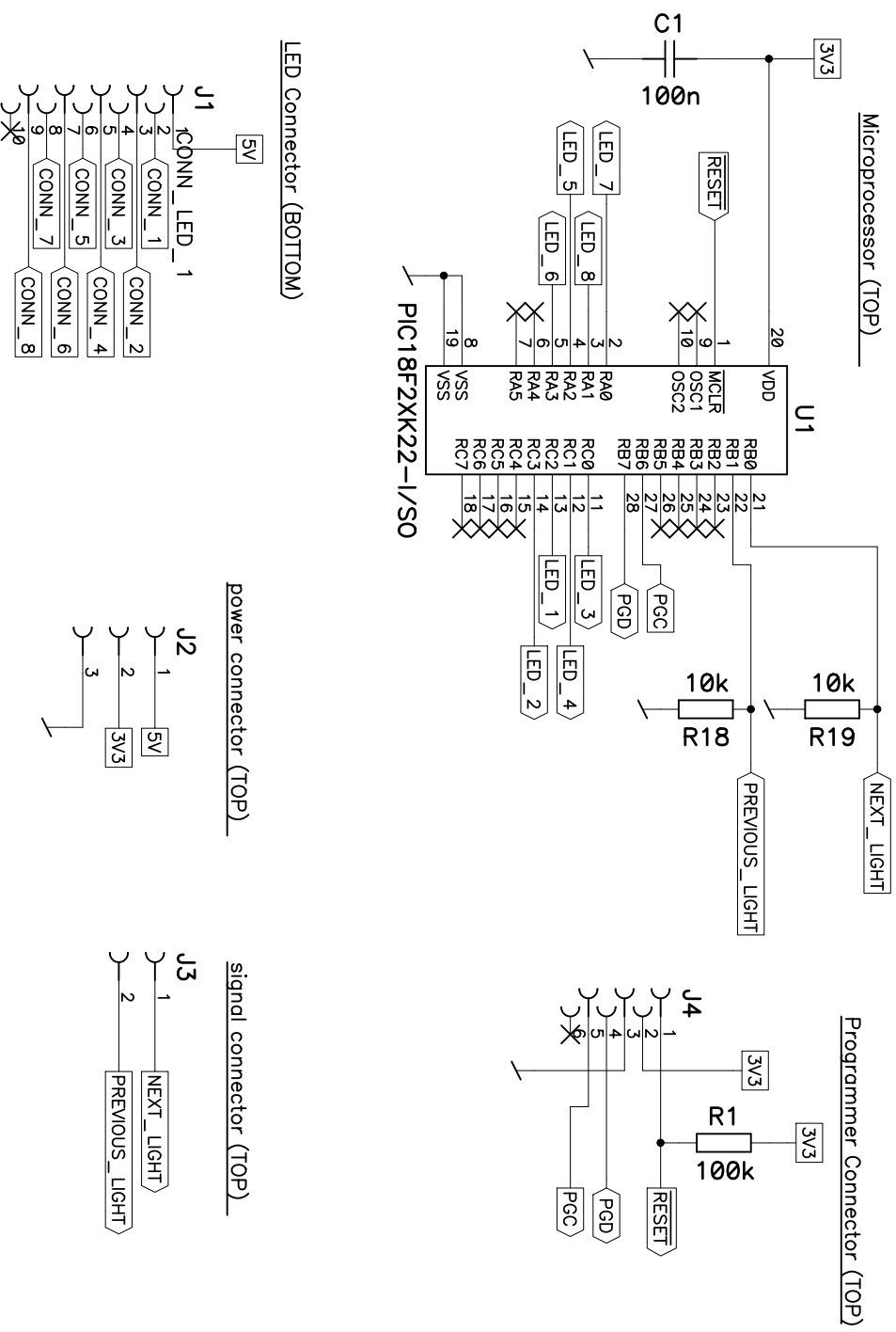


Power PCB		DES	06.07.15	N. Marty
Raspberry Pi 2 Camera Sheet2		REV	V3.0	
HAUTE ECOLE VALAISANNE		2/2	energy management	power_pcb.sch

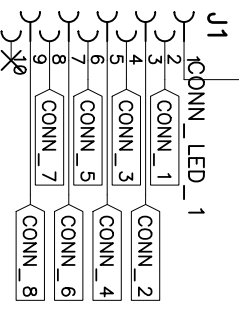




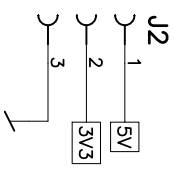
Lightning PCB		DES	05.06.2015	N.Marty
Raspberry Pi 2 Camera Sheet2		REV	V2.0	
HAUTE ECOLE VALAISANNE		2/3	light control	light_pcb.sch



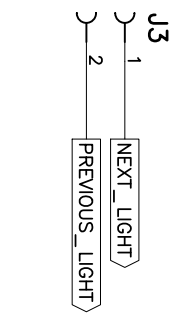
LED Connector (BOTTOM)



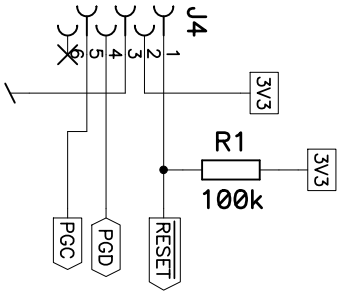
power connector (TOP)



signal connector (TOP)



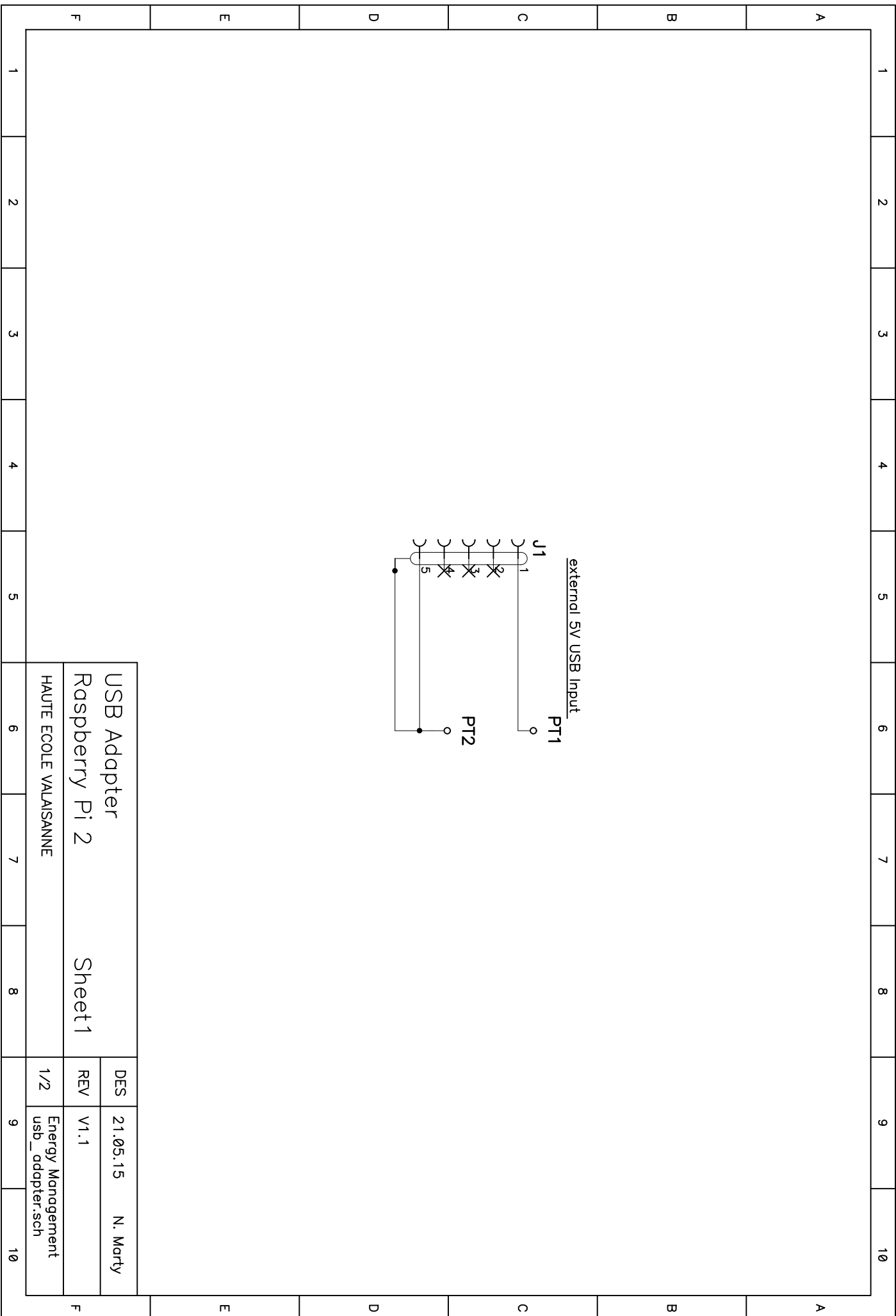
Programmer Connector (TOP)



Lightning PCB
 Raspberry Pi 2 Camera Sheet1
 HAUTE ECOLE VALAISANNE

DES	05.06.2015	N.Marty
REV	V2.0	
1/3	light control light_pcb.sch	

A	1	2	3	4	5	6	7	8	9	10
B	1	2	3	4	5	6	7	8	9	10
C	1	2	3	4	5	6	7	8	9	10
D	1	2	3	4	5	6	7	8	9	10
E	1	2	3	4	5	6	7	8	9	10
F	1	2	3	4	5	6	7	8	9	10

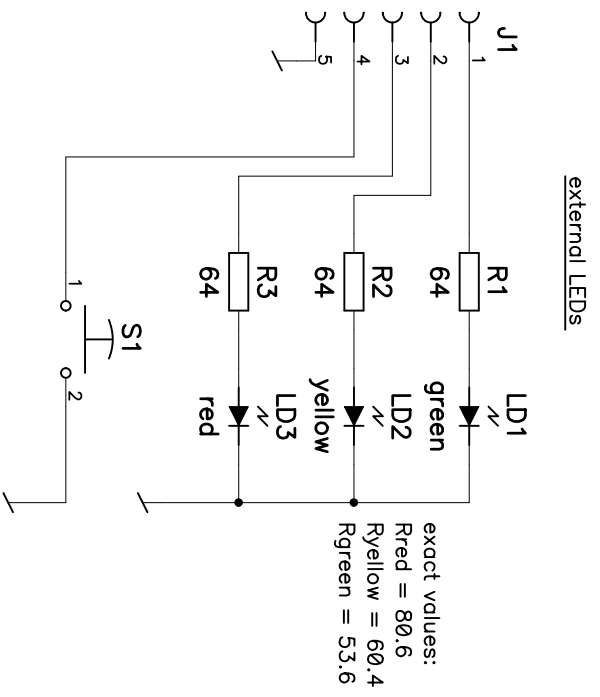


USB Adapter		DES	21.05.15	N. Marty
Raspberry Pi 2		REV	V1.1	
HAUTE ECOLE VALAISANNE		1/2	Energy Management usb_adapter.sch	

Sheet1

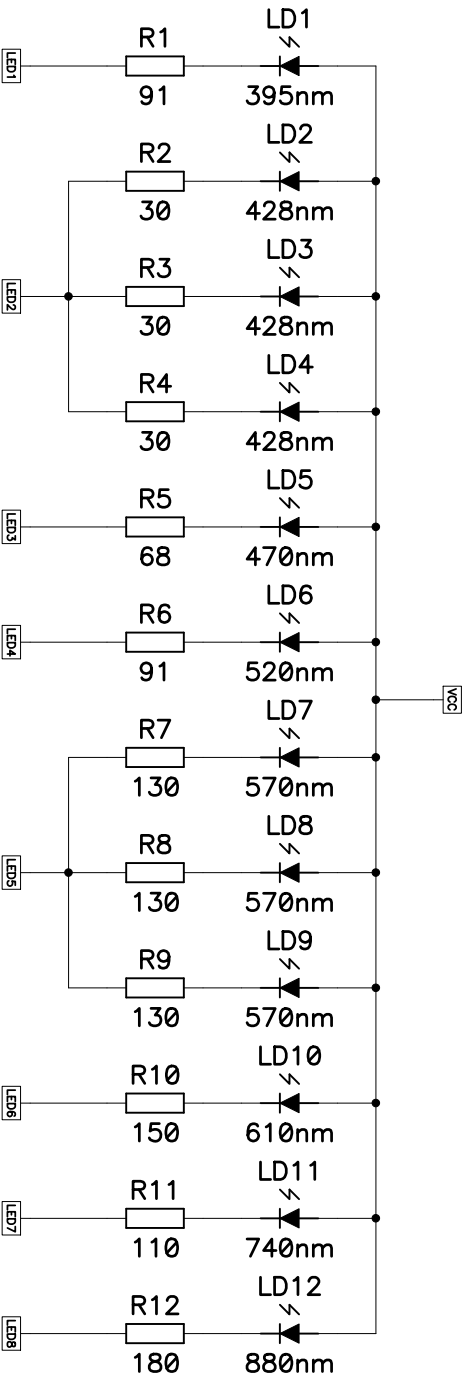
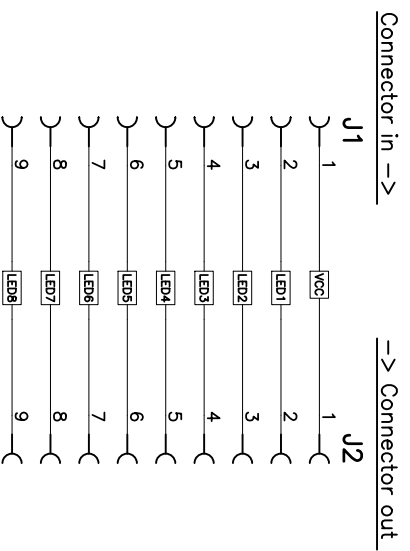
1 2 3 4 5 6 7 8 9 10

A B C D E F



external LEDs

external LEDs Raspberry Pi 2 HAUTE ECOLE VALAISANNE	Sheet1	
	DES	21.05.15 N. Marty
	REV	V1.1
1/2	Energy Management indication_leds.sch	



Illumination		DES	23.06.15	N. Marty
Raspberry Pi 2 Camera Sheet1		REV	V2.0	
HAUTE ECOLE VALAISANNE		1/2	illumination_illumination_8Wavelengths.sch	