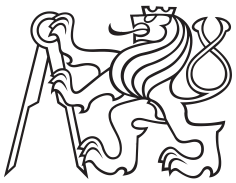
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Remote control touch panel for Thermal Smart Camera

Ján Marek

Supervisor: Ing. Jan Kovář
Supervisor–specialist: doc. Ing. Daniel Novák, Ph.D.
January 2019

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Marek Ján** Personal ID number: **406310**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Master's thesis details

Master's thesis title in English:

Remote Control Touch Panel for Thermal Smart Camera

Master's thesis title in Czech:

Dotykový ovládací panel pro termovizní smartkameru

Guidelines:

1) Get familiar with the thermal smart-camera SMARTIS, communication interface and usage of the camera in industry (working modes and operator settings).
2) Find a suitable platform for creating the touch panel and select the appropriate display screen, central and peripheral board for connection of the thermal smart-camera. The central process platform must support OS Linux and libraries for image processing and communication through an Ethernet IP.
3) According to the selected platform try and run OS Linux and development tools for the design of application software of the future control panel.
4) Propose and implement the optimal transmission and display of the thermal image depending on the options of the camera and selected platform.
5) Design and implement the application and graphical interface for remote access and control of the thermal smart-camera SMARTIS. Implement separate interface for the administrator and operator user.
6) Test the designed application and evaluate future development options.

Bibliography / sources:

[1] Vollmer, Michael; Möllmann, Klaus-Peter: Infrared Thermal Imaging: Fundamentals, Research and Applications, First Edition, Wiley-VCH, 2010.
[2] Davies, E. R.: Computer and Machine Vision: Theory, Algorithms, Practicalities, Fourth Edition, Academic Press, 2012.
[3] Yaghmour, Karim; Masters, Jon: Building Embedded Linux Systems, Second Edition, O'Reilly Media, 2008.

Name and workplace of master's thesis supervisor:

Ing. Jan Kovář, Workswell s.r.o., Prague

Name and workplace of second master's thesis supervisor or consultant:

doc. Ing. Daniel Novák, Ph.D., Analys. and Interpr.of Biomed. Data, FEE

Date of master's thesis assignment: **12.01.2018** Deadline for master's thesis submission: **08.01.2019**

Assignment valid until: **30.09.2019**

Ing. Jan Kovář Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

My first thanks is to the supervisor Ing. Jan Kovář and the company Workswell s.r.o. for allowing me to work on this project under their care.

I would like to thank my parents and my whole family for their support and infinite patience. I also thank my girlfriend that was standing next to me every day and supporting me morally.

Last but not least, I would like to thank the CVUT and my faculty for allowing me to gain strong knowledge and experience for the future life.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date ....

signature

Abstract

Main goal of the diploma work is to find suitable platform and develop a graphical application for a remote control touch panel, for thermal smart camera SMARTIS. The application contains two modes, for Operator and Administrator user.

In the thesis I describe the process of choosing the single board computer, installation of the operating system and main development of the application. Later I state the process of cross-compilation for the final platform.

The graphical application is written in C++ using the Qt libraries. The main functions of the application with examples are described in the thesis. The application was tested for long term stability and is fully functional.

In the future the application will be further improved and new functions will be added. The goal is to integrate the remote control touch panel directly to the camera and potentially introduce it to the market.

Keywords: thermal imaging, thermal camera, industrial, application development, single-board computer, Qt, touch panel

Supervisor: Ing. Jan Kovář
Workswell s.r.o.,
Libocká 653/51b,
161 00, Praha 6

Abstrakt

Hlavním cílem diplomové práce je najít vhodnou platformu a vyvinout grafickou aplikaci pro dotykový ovládací panel, pro termovizní smart kameru SMARTIS. Aplikace obsahuje dva módy, pro Operátora a Administrátora.

V této práci popisuji proces výběru jednodeskového počítače, instalaci operačního systému a hlavní vývoj aplikace. Později pak uvádím proces cross kompilace na finální platformu.

Grafická aplikace je napsána v jazyku C++ s použitím knihoven Qt. Hlavní funkce aplikace jsem popsal v diplomové práci spolu s uvedením příkladu. Aplikace byla testována na dlouhodobou stabilitu a je zcela funkční.

V budoucnosti bude aplikace dále zdokonalována a budou přidány nové funkce. Cílem je integrovat dotykový ovládací panel přímo ke kameře a potenciálně jej uvést na trh.

Klíčová slova: termální zobrazování, termovizní kamera, industriální, vývoj aplikace, jednodeskový počítač, Qt, dotykový panel

Překlad názvu: Dotykový ovládací panel pro termovizní smartkameru

Contents

Figures

Tables

Chapter 1

Introduction

The diploma thesis deals with creation of remote control panel for industrial Thermal Smart Camera SMARTIS. At the start of the project I needed to research possible options for a suitable platform based on the selected requirements of the application. I also prepared an overview of different Thermal imaging platforms, that are in use today and on which i could build on. I also created a short introduction to the problematic of thermal imaging with technologies the SMARTIS Thermal camera uses.

Next part of the work is a preparation of the selected platform, installation of Linux system and required libraries with cross-compilation of the code. The project also describes the selection of a development tools and implementation on the platform.

The biggest part of the diploma thesis is creation of the final application for the remote control touch panel, description and implementation of the individual classes and functions. Requirement of the work was also preparation of a library, that can work with the given functions and classes of the Thermal camera SMARTIS.

In the final part of the thesis I describe possible improvements and future work that could improve the final application.

Chapters in the thesis are in order as I was encountering individual problems and solved different task. I start with complete basics of the project all the way to the proposition of final product that could be potentially introduced to the market in the future.

The project is meant as a prototype of a remote control panel for the already existing Thermal camera SMARTIS, which is an intellectual property of the company Workswell. In the development of the project I received already pre-build project with functions of the Thermal smart camera, which helped me to create the communication library for my platform. During the project I encountered many difficult obstacles I had to solve to progress. Some tasks were often showing more difficult than expected and I had to do a great research to tackle these problems. In the end I managed to bring the project to the desired end and created a product, that could be potentially used in the industry.

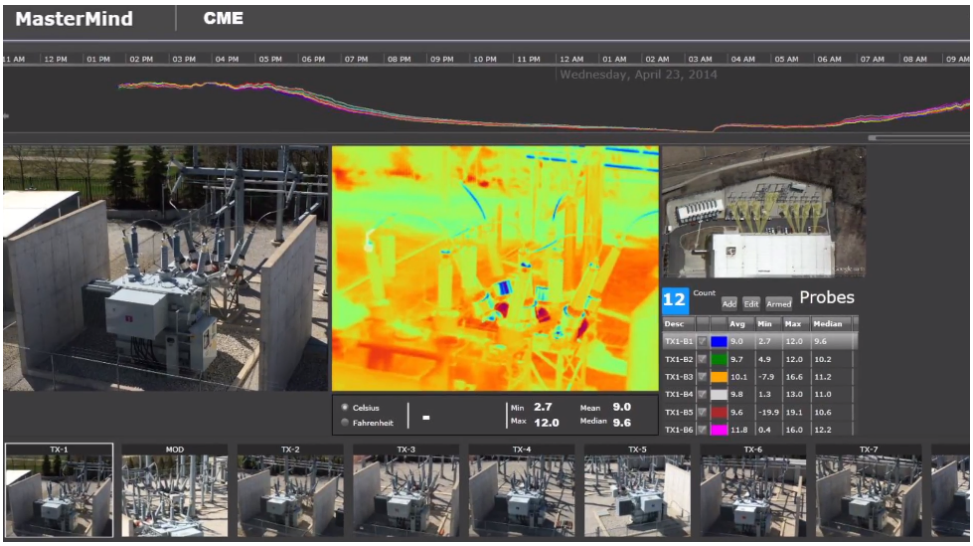


Figure 1.1: Example of professional monitoring system using FLIR thermal camera for detection of power plant failure.

1.1 Motivation

The development of thermal imaging is growing every year and thermal imaging is used in more and more areas and fields in our life. Today, thermal cameras are used in medicine, military, surveillance applications, industry and many others. Acording to the Global Market Insights research [1], the global thermal imaging market will exceed worth of 10 billion USD by the year 2024.

The final product of this thesis is aimed for professional use in the industry and serve as a remote panel for full control of stationary thermal cameras. The use of this type of camera widely spreads in the industry and the developed platform has great potential to be used in the future alongside of them.

1.1.1 General Use

The main area of use for the final product would be in a factory, where industrial thermal cameras are installed. These cameras can observe different machines or devices for control of overheating, correct temperature on individual parts of the devices or detection of foreign object on the conveyor belt of the machine.

Another use of industrial cameras is in surveillance of area, more specifically detection of fire in the room or parts of the observed area.

The remote control panel will be placed to the operator area of selected machine and serve as control unit for the camera with real time feedback while observing the measurement.

1.1.2 Personal motivation

Before getting the chance to work on this project I never truly experienced work on a product from the complete start. I was looking forward to create something new that could potentially be used in the industry in the future and have a chance to influence the development and final product.

The big part of the project was to get to know operating system Linux, which is a system I never worked with on this scale. The requirement to research hardware options for this project and combine it with created application was also big challenge I wanted to try and experience.

The company Workswell s.r.o. is smaller but growing company, that gave me the opportunity to try all of these new things while working with cutting edge technologies in thermal imaging I would have hard time to find elsewhere. The working environment was really pleasant and welcoming to my ideas.

1.2 Goals

The goal of the project is to prepare a platform for existing Thermal camera SMARTIS that will serve as a remote control panel for the camera and its functions.

The requirements could be split and explained in these following parts:

- Selection and preparation of the platform
  - Consideration of possible platforms
  - Selection of the platform
  - Installation of operating system and libraries on the platform
- Selection of development tools for the application
  - Selection of programming language
  - Preparation of the libraries for the graphics of the application
  - Testing of cross-compilation on the selected platform
- Writing of a library for the camera and its functions
  - Getting familiar with the provided classes and functions of the thermal camera
  - Creation of libraries for controlling these functions and getting response from the camera to the final application
- Main application development
  - Creation of different modes for the application
  - Writing of the individual functions and preparation of graphical interface

- Testing and proposition of future development and improvements

These individual tasks are further introduced and explained in the diploma thesis together with my proposed solution and the processes of solving the problems.

1.2.1 Limitations

There were not many limitations I had to deal with and I mostly had full control of the project itself. The main constraints were:

- Touch screen panel size
  - The size of the used touch screen needs to be at least 10" to serve as a suitable displaying platform for the application.
- Computational speed
  - The selected platform needs to be able to process the image in the fastest and most precise way possible
- Price
  - The final product is potentially targeted at the market and needs to be able to compete with other products, thus lowering the price for minimum.

1.3 Existing alternatives and solutions

In the recent years the thermal imaging technologies became popular in the industry and the demand is growing. The requirements for thermal camera in the industry applications is high precision and quick evaluation of the measured data with quick response time.

There are 2 main solutions for the industrial camera in the praxis: Thermal camera connected to a dedicated computer or thermal camera connected to control panel.

First category is usually implemented for bigger operator areas, where multiple cameras are connected to one software and the operator is overlooking different measurements from one place. The application is usually run on a computer with operating system windows and controlled using mouse and keyboard.

Thermal cameras connected to a control panel are often in close proximity of the machine or area the camera is observing. The advantage is fast access and quick response to real time need of the operator of the machine through the control panel in his reach. These control panels can be controlled with mouse and keyboard or touch panel.

(a) : FLIR ThermoVision CM **(b) :** Infratec PRESS-CHECK

Figure 1.2: Professional thermal cameras with control panel

1.3.1 FLIR ThermoVision CM

Flir ThermoVision CM [2] is an industrial application with 2 options, desktop or pc panel solution. It is a computer compatible with many FLIR cameras with a built in monitoring software.

The software allows recording of the measured data with 24/7 data streaming. This solution is used mostly in automation and safety applications.

1.3.2 Infratec PRESS-CHECK

InfraTec. PRESS-CHECK [3] is an automated test system for continuous measurement using two thermographic cameras. It is used in press hardening application, where the thermal cameras control the temperature of the material before and after pressing.

The solution is connected to a high level control system that contains the required software for the cameras. This control system is often part of the press and it allows the operator of the control software quick setting changes of the application without the need of special set-up time.

1.4 Overview of the final product

To better understand what the proposed solution is, I would like to describe it shortly in this chapter.

The core part of the product is a computer NanoPC-T3 with the ARM processor. It has 16GB eMMC on-board with the installed Linux operating system, based on ubuntu core using Qt-Embedded GUI. The eMMC also contains the running application.

SMARTIS Thermal camera is connected to the Ethernet port on the NanoPC-T3. The camera communicates with the board through TCP/IP protocol. The application sends requests and the camera responds. The request is handled in the application the results are displayed. The application sends selected settings and commands to the camera and the camera then

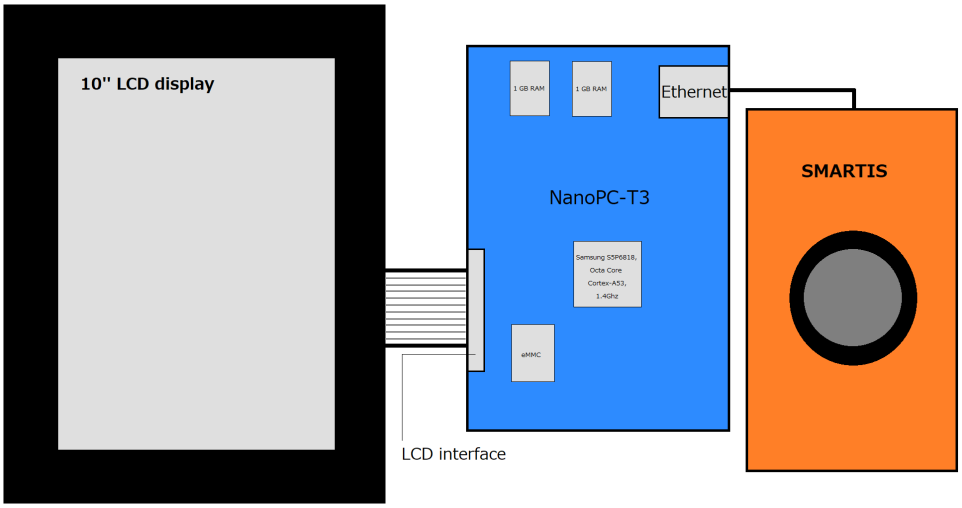


Figure 1.3: Simplified scheme of the final product. The displayed devices are not in scale

works with these settings and produces outputs during the measurement session.

The touch panel is connected through LCD connector to the board and serves as a display and input device for the camera. Through the touch panel interface the user can operate the application, observe the measurement and prepare various settings for the camera.

Chapter 2

Feasibility analysis

Before starting the work on this project, I needed to make analysis of all of the requirements and if I will be able to meet them. The first part of this process was to get familiar with the thermal camera SMARTIS.

In this chapter I will be explaining the main functions of the camera, that will need to be controlled by the remote panel.

2.1 SMARTIS

Workswell Thermal camera SMARTIS (Smart Thermal Imaging System) [4] is a thermal camera made by the company Workswell s.r.o. It is one of the intelligent thermal cameras, providing all in one solution of thermal vision, data collection and evaluation, input and output handling and more.

It is used as an industrial camera in the processes of automation and machine vision with the option of long continuous measurement in many different applications as research and surveillance or safety. The camera is equiped with a TCP/IP interface with a built-in web server for communication with a computer or any PLC unit with TCP/IP interface support.

2.1.1 User modes

The camera contains 2 different user modes: Operator and Administrator mode.

The Administrator mode allows the user to configure the SMARTIS camera and prepare the measurement with required settings for the Operator. Administrator can create new Operators or delete old ones.

Operator mode is used for observation of the measurement. The user in this mode can select from one of the previously configured settings by the Administrator and start / stop the measurement, while observing the measured values and produced outputs.

2.1.2 Products overview

The set of configurations in the SMARTIS camera is called Product and it is created and edited by the Administrator. The Administrator needs to

Figure 2.1: Workswell s.r.o. Thermal Smart Camera SMARTIS

go through all the settings to prepare the product for the future use. Each individual product is created for each user separately.

2.1.3 ROI

The main part of each product is the ROI - region of interest. These regions are drawn on the picture of the camera. Each ROI has its own set of settings selected by the Administrator. This allows the user to create different areas in the picture where the measurement evaluates different statistics and produces different outputs.

2.1.4 Inputs and outputs

SMARTIS contains digital inputs called digital triggers, 7 digital outputs and 4 analog outputs. The digital triggers are used for starting or stopping the measurement.

Digital outputs are generated based on the evaluated signal of each ROI. SMARTIS can produce digital signals with different pre-set length with rising or falling edge. Analog outputs are also generated based on the settings of each ROI and the signal evaluation. Each analog output can be set as a voltage or current. The camera can produce analog signal in range of +-12V or +-24mA.

2.1.5 Measurement modes

There are 4 different measurement modes in the SMARTIS camera.

- Frame Measurement and Evaluation
- Start-Stop Measurement and Evaluation

- Continuous Evaluation
- Non-Trigger Evaluation

The Frame Measurement mode needs a trigger to be started. After the trigger the measurement stops after predefined sequence length, which can be set by the Administrator user.

In the Start-Stop Measurement mode the user can control the measurement in 2 ways:

- One trigger - starts the measurement with Latch trigger in logical high state and stops it when the Latch trigger goes into the logical low state.
- Two triggers - the measurement is started by one Start trigger and stopped after using one Stop trigger.

Continuous Evaluation mode is very similar to the Start-Stop mode where you can trigger the measurement with one or two triggers with the option to set sample interval of the measurement. When the measurement becomes active, SMARTIS evaluates one frame repeadetly with the pre-set sampling period.

Non-Trigger Evaluation mode requires no digital triggers. The measurement is started when the Operator selects Start measurement in the Operator mode.

2.2 Ethernet communication

In the project I had an opportunity to select between working with TCP protocol or UDP protocol. After my first consideration of the requirements I opted for the TCP protocol for its high reliability and segment sequencing. I realized the data I will be sending to the camera need to come in the right order and reliably all the time.

For the data stream (picture) from the camera the UDP protocol could serve as a possible alternative, thanks to the fact, that not every frame needs to necessarily arrive and be displayed. I did not want to include another protocol and decided to use TCP for both picture data stream and other data evaluations from the camera.

Chapter 3

Platform selection and preparation

In this chapter I would like to describe my approach and thought process while selecting the platform for my project. Also I will be describing preparation of the computer as it chronologically progressed, going through the installation of the platform and also mentioning approach I abandoned for different one.

The platform I decided to choose was NanoPC-T3 with Ubuntu core Linux operating system.

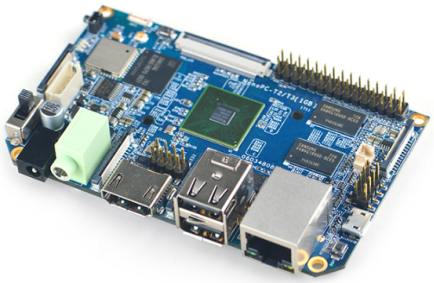
3.1 Platform requirements

The requirements for the platform are closely related to the goals and limitations of the project and can be summarized into these points:

- Powerful computational capabilities
- Ethernet port for connection of the camera
- Possibility of touch screen input
- Big enough internal storage space
- Strong background of the platform with a lot of documentation
- Storage space for future development and new functionality of saving data
- Lowest price possible

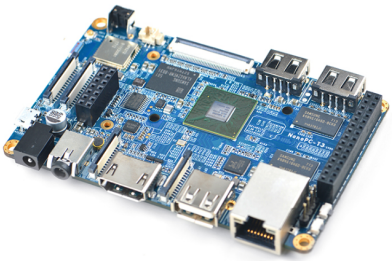
I organized the points based on the necessity and weight of each requirement, meaning that powerful processor capabilities and higher computational speed are valued higher than requirement for low price of the platform.

Thermal smart camera SMARTIS needs to be connected to the platform through Ethernet port which is the only requirement for camera connection. As a displaying device I will be using touch screen that needs to be connected to the board.



(a) : NanoPC-T2

(b) : Rico Board



(c) : NanoPC-T3

Figure 3.1: Considered single board computers for the application

3.2 Platform comparison

While selecting the most suitable platform, I focused on single board computers. These computers contain everything my project will need on single board without the need of adding additional hardware or thinking of special connections.

Today, single board computers very common and easy to find with powerful capabilities. Probably the best known single board computers are Raspberry Pi, Banana Pi or Arduino. In the past, there processors with the ARM architecture were more common, but today one can find very compact and powerful Intel or AMD processor single board computers supporting operating system Linux or Microsoft Windows.

Before selecting the platform I made research on the availability of the platform and documentation with examples for each platform, because in the past I had only little experience on using one and did not know what it might require.

In the end I was choosing between 3 platforms: NanoPC-T2, Rico Board and NanoPC-T3.

3.2.1 NanoPC-T2

NanoPC-T2 [5] is the first single board computer I tried. It is board made by the company Friendly Elec. The board has quad core processor with frequency up to 1.4GHz and it is a successor of the previous board NanoPC-T1. This board has internal storage eMMC 8GB with the option to use SD card. It contains all necessary ports I will need in my project and the documentation to this platform is great with many tutorials.

I had fairly easy access to this board and I decided to try it for my future development. I was able to install Lubuntu (lightweight Ubuntu) Desktop core on the board supporting X Windows. The reason behind this is that I wanted to develop the application directly on the board without the need of cross-compilation.

I was able to run simple code to try my graphical interface but later I opted from this platform, because I was not satisfied with the results. The environment was slow-responding and the code building was also time consuming. This platform was also already older in the time of me starting to work on my project and I moved to different options, Rico Board and NanoPC-T3

3.2.2 Rico Board

Rico Board [6] is a platform from manufacturer MYIR. It is single board computer with quad core ARM processor with frequency up to 1GHz. The Rico Board is equiped with internal storage eMMC 4GB with microSD card slot. All necessary ports are also present so the board could be potentially considered.

Big advantage of this platform is great documentation but smaller community of users to look for examples and help.

3.2.3 NanoPC-T3

NanoPC-T3 [7] is direct successor of the NanoPC-T2 platform by the company Friendly Elec. It has powerful octa-core with 2GB DDR3RAM, 16GB eMMC FLash and microSD card connector for additional storage. The platform also contains all necessary ports for my application and is suitable for future development.

Very big advantage of this platform is great documentation and big enough community that can help troubleshooting future problems that could occur during the development.

At the end I was choosing between Rico Board and NanoPC-T3 and for the better processor, bigger storage and surprisingly lower price at the time of research, I decided to select NanoPC-T3 and develop my application on this platform.

One of the factors was also that I had rather small but some knowledge with the previous platform NanoPC-T2 and wanted to use my experience further.

	NanoPC-T2	RicoBoard	NanoPC-T3
Processor	Samsung S5P4418 Quad Core Cortex-A9, 1Ghz	TI AM4379 ARM Quad Core Cortex-A9, 1Ghz	Samsung S5P6818 Octa Core Cortex-A53„ 1.4 Ghz
RAM	1GB DDR3	512MB DDR3	2GB DDR3
Storage	8GB eMMC + microSD card	4GB eMMC microSD card	16GB eMMC microSD card
Display	HDMI interface, LCD interface	HDMI interface, LCD interface	HDMI interface, LCD interface
Interface	Ethernet, Mini USB 2.0, USB 2.0	Ethernet, Mini USB 2.0, USB 2.0	Ethernet, Mini USB 2.0, USB 2.0
Price	$59	$99	$75

Table 3.1: Parameter comparison of considered single board computers

Comparison overview

For quick comparison of each platform, I would like to include Table 3.1 of the platform parameters. I got the data from the official web page of each manufacturer.

3.3 Preparation of selected platform NanoPC-T3

The manufacturer of the platform delivered the board with the operation system Android installed. First thing for me was then try to uninstall the Android and install some form of Linux distribution for my future development. The NanoPC-T3 also came with bootloader, more specifically uboot. My first course of action was to prepare linux distribution on SD card and boot the system following by installation on the eMMC flash.

3.3.1 First Linux installation

I downloaded Lubuntu Desktop image for ARM core from the manufacturers web page [9], containing installation files and utility called eFlasher [8]. eFlasher is a helping utility to install operating system image easily to the friendly ARM boards eMMC. Then I downloaded and started Win32DiskImager [10] on my Windows computer. Here I was able to create installation SD card for my NanoPC.

With the installation SD card I proceeded to install Lubuntu to my platform and installed updates for the OS.

With my new OS running on the board, I did simple tests with running video from the internet and I installed LXQt [11] as my development platform to start developing the application. After developing small testing program for my GUI I ran again into the problem of rather slow build speed and the system stopped responding. I realized that running the program and all the

development tools is too much, even for this computer and I need to find different solution, how to develop my application.

3.3.2 Preparation of Ubuntu core

On the web page of NanoPC-T3 I found link to an image file of FriendlyCore OS [12], based on Ubuntu core without X windows. I followed similar steps as during installation of the Lubuntu OS and managed to install this OS onto the board.

After running some tests I decided to work on this OS but I had to abandon my future plan of developing the application directly on the board due to the OS restrictions.

The FriendlyCore OS was able to run full HD video on my connected touch panel. The OS is still able to run single X-Windows application, which I tested by creating small GUI program and running it on the platform.

The Ubuntu Core supports Qt 5.10.0, supports Ethernet, LCD port for the touch screens with both resistive and capacitive touch, audio playing with WiFi and Bluetooth.

The Friendly Elec also provides Qt Demo application to try out all the features and capabilities of the board. After this I decided to stay on the Ubuntu Core as my main OS for the application.

3.4 Summary of preparations

The selection of the new platform was long process but inevitably so, because of my lack of previous experience. It was necessary to try out different options and consider all requirements before commiting to work on one specific board.

After trying out NanoPC-T2 and successfully installing OS on it I decided to not use this platform for the other more powerful successor NanoPC-T3.

On the NanoPC-T3 I was able to install light Linux distribution Lubuntu, which had great advantage of having all required software available and easy to use. The biggest advantage was perhaps no need of cross-compilation of the application from different system. Even thanks to powerful octa-core processor, the system was still slow and I needed to try different approach.

I installed Ubuntu Core without X-window on the board, OS, where I could not run development software but sufficient enough for everything else, especially very responsive and fast with all my tests I put it through.

NanoPC-T3 was in my opinion best suited board for the requirements of the project and I did not regret my choice during the development phase.

Chapter 4

Application development

The first thing I needed to consider while deciding how to develop the application is programming language. The SMARTIS library I received from the company Workswell s.r.o. was written in the C++ language and I decided to write my code in it as well.

Another step was to select correct set of libraries I could work with to develop complete application with solid graphical interface. From the needs of the project and my considerations came out these requirements for the library:

- Support of destined device - ARM platform
- Support of programming graphical user interface
- Support of C++ programming language
- Good documentation and strong community

The first choice of library was Qt [13], because it is a library, that is directly supported in my selected platform NanoPC-T3 and it has strong background with great documentation. Qt as a library is great for development of graphical applications, code written using Qt uses LGPL license and Qt supports many programming languages such as Java, Python or C++ which is a language i will be using.

4.1 Selection of develomplent method

Another step of my project was to consider and select correct method of developing the application. This process was very important because different options use different compilation methods and compilation speeds and with the right method I could save a lot of time while developing the final application. There are 3 main options of developing application on a platform like mine:

- Development of the application on the destined platform
- Development of the application on different host computer and compilation on the destined platform

- Development of the application on different host computer, cross-compilation it on the host computer and run the final application on the selected platform

Development of the application on the destined platform comes with the biggest advantage of no requirement of cross-compilation of the code, which can be lengthy and difficult process. The drawback is that this approach requires powerful host platform with sufficient computational speed to run operating system supporting x-windows, so the development applications can be used.

The method of developing the application on a host computer while compiling it on the destined platform is also suited for very powerful platforms. In this approach I would be writing the code on host computer and compiling it through the development environment directly using Ethernet on the selected board. First and foremost this requires that the development environment supports this option. Another complication could be debugging of the code on the selected platform and this process could prove slow and difficult, especially with the project size increasing.

Last approach - developing the application on a host computer and cross-compiling it for the final platform is the best for the selected board in terms of computational speed of the device. All the requirements of the development lie on the host computer and the final platform does not require installation of any development tools. The biggest drawback could be the necessity of cross-compilation for the final platform.

Final choice of the development method

My first selected approach was to develop and compile the application on the selected platform NanoPC-T3. With smaller applications the compilation times were good but once my program size increased the compilation times became longer and the development process started to be less effective because of this.

Considering my other options, I decided to use the third approach and develop the application on a host computer and cross-compile the final code for the final platform. This approach was in the end very comfortable and I could easily debug and tune my application with short compilation times.

In the next chapters I would like to describe my process of preparing the development environment and cross-compilation for the NanoPC-T3.

4.2 Preparation of development environment

Next step of this project was to prepare the development environment. After I decided to use Qt library and write the code in C++, the Qt Creator [14] appeared as the easiest and correct choice. It provides great tools for development graphical applications together with Qt Designer. The environment of Qt Creator is very intuitive and easy to work with.

4.2.1 Library selection

While installing the Qt Creator, I had to select the Qt version I want to work with. My choice was between Qt versions 4.X and 5.X.

Qt version 4.X is today older but still many applications use this version for the QWS system, which is Qt Windowing System that eliminates the need of X11 windowing system and the Qt applications write directly to the Linux framebuffer.

Qt version 5.X is newer version and updates are still being released to this day. The function difference between these two versions didn't appear too big and Qt 5.X promised newer support for the networking and new QtWebEngine which I considered using.

Friendly Elec and NanoPC-T3 directly support development of Qt Applications in Qt version 5.X and the company also provides tools for cross-compilation directly for their board.

For writing of the library for SMARTIS camera I chose to use Qt version 4.8.6 because it was the version which the camera supported. For the main application for the NanoPC-T3 I chose to work with Qt version 5.9.1, which was the latest version of the Qt library when I was developing the application.

4.2.2 Cross-compilation for final platform

Before beginning the work on the main application, I needed to test the cross-compilation for the NanoPC-T3. Friendly Elec provides cross compiler for Qt with version 5.9.1 directly on their page. After downloading the compiler and installing I tried to compile simple application containing one window in Qt Creator and then run the cross compiler for my platform.

To crosscompile the code I used this input:

```
mkdir build && cd build
/usr/local/Trolltech/Qt-5.9.1-nexell64-sdk/bin/qmake ../diplomka/
diplomka.pro
make
```

After that I loaded the application to the NanoPC-T3 and successfully tested it.

4.2.3 Qt Creator settings

Next step was to properly set my Qt Creator for development of the main application. The set-up of the Qt Creator can be described in three steps:

- Setting up the development kit
- Setting up the .pro file
- Setting the arguments for the application

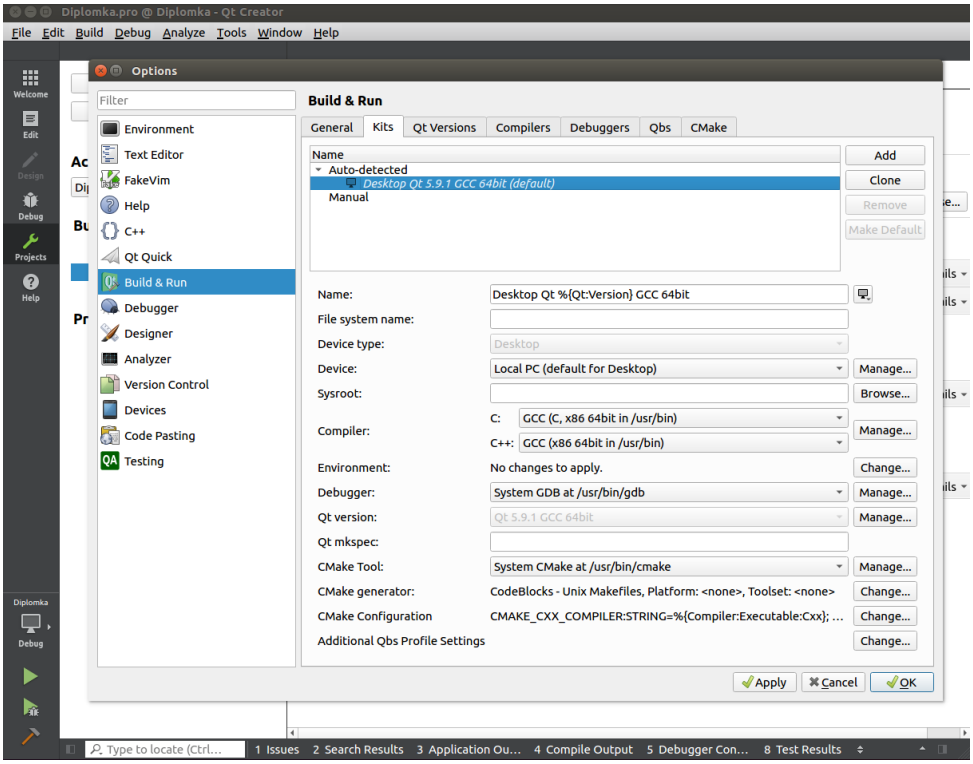


Figure 4.1: Settings of the Qt Creator for the main application development

I will be developing my application on host desktop device and then cross-compile the built application from the host computer to the NanoPC-T3. Because of that, my settings of the development kit are set for the desktop device in the Qt Creator. First I will select the Qt Version as 5.9.1, I will set the device type as desktop, select compiler for the C++ code and system debugger.

The .pro file contains all information about the project like used libraries, headers and sources together with all forms used in the application.

Lastly in the arguments I could specify that I want to build the project on more cores, I could also enter argument for running of the application, which I did not use.

4.3 Application requirements and overview

In the last part of this chapter I would like to introduce the main application, main function of the application and requirements from the project description. In the future chapters I will go into greater detail of each part of the main application and describe remaining necessary steps I had to take to finish the program.

I had to get familiar with the thermal smart camera SMARTIS and address these requirements in my application:

- Create two modes for operating the camera

- Admin mode
- User mode

- Display the stream from the thermal camera
- Create interface for setting configuration of the camera
- Allow creation of new users
- Save logs from the application

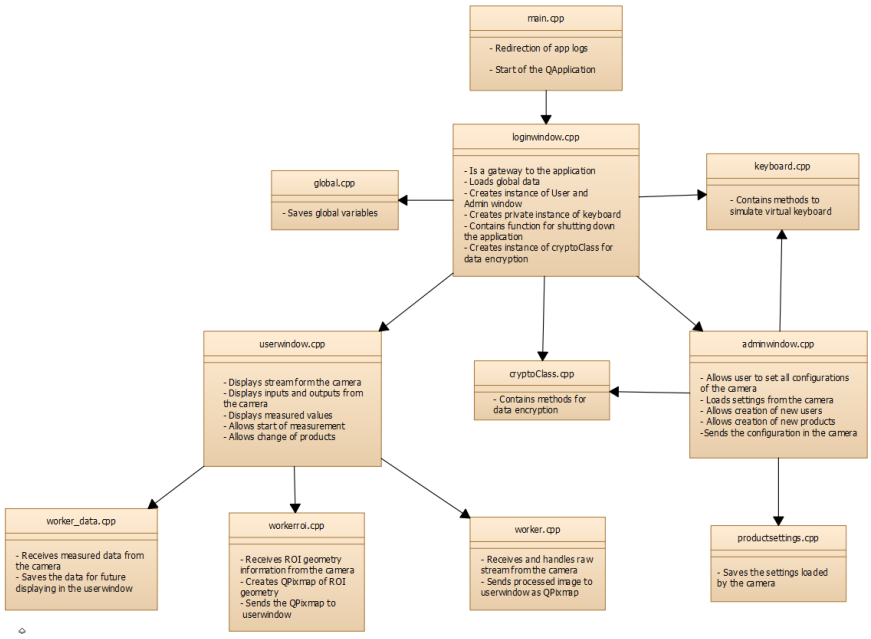


Figure 4.2: Simplified class hierarchy of the application with explanation of each class

4.3.1 Class dependency

In this part I would like to introduce the basic hierarchy of classes in my application. You can see the simplified hierarchy with description on the Figure 5.7.

The application is started from the file `main.cpp` with the function `main()`. This class contains redirection of all logs from the application to the log file called `smartis_panel_log`. In the function `main()` also the first main window is created, which is the Login window.

The file `loginwindow.cpp` contains initialisation of remaining two main windows, Admin window and Operator window. The file `cryptoClass.cpp` is called in the login window and serves as encrypting class for the user login information. The login window serves as a login gate for the user and then redirects the user to the next window of the program.

The file `userwindow.cpp` is the main window for the Operator user and it works with two important threads: `image_thread` for displaying the picture

from the camera and `data_thread` for showing all the remaining data of the measured values.

Library handling the `image_thread` is described in the file `worker.cpp` and the library for the `data_thread` is described in the file `workerdata.cpp`.

The most complex library for the Admin window is described in the file `adminwindow.cpp`. It uses libraries in `productsettings.cpp` and `keyboard.cpp` for setting all of the cameras variables and sends them to the SMARTIS camera.

4.3.2 Main windows overview

There are three main windows in my application - Login window, Operator window and Admin window. Work on the design of these windows was great part of the project. My main goal was to always put all necessary information on the screen while not overwhelming the user. The big challenge was also to make the control of the application intuitive and easy without the need of reading manuals.

Login window

Main function of this window is to display the connection status of the camera and provide a gateway to the application through login screen.

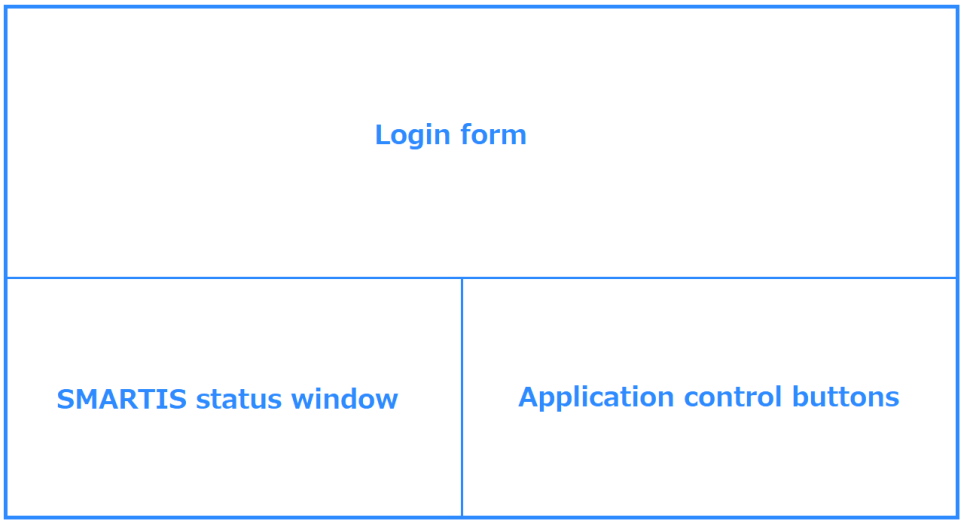


Figure 4.3: Simplified scheme of the login window GUI

Operator window

This window is aimed at the Operator of the camera, who is observing the measurement. The main goal was to display all measured values from the camera and the picture from the SMARTIS. In this window the user can:

- Increase the size of the picture

- Change the color palette of the picture
- Select product he wants to measure
- Start and stop the measurement
- Observe the camera status and measured values

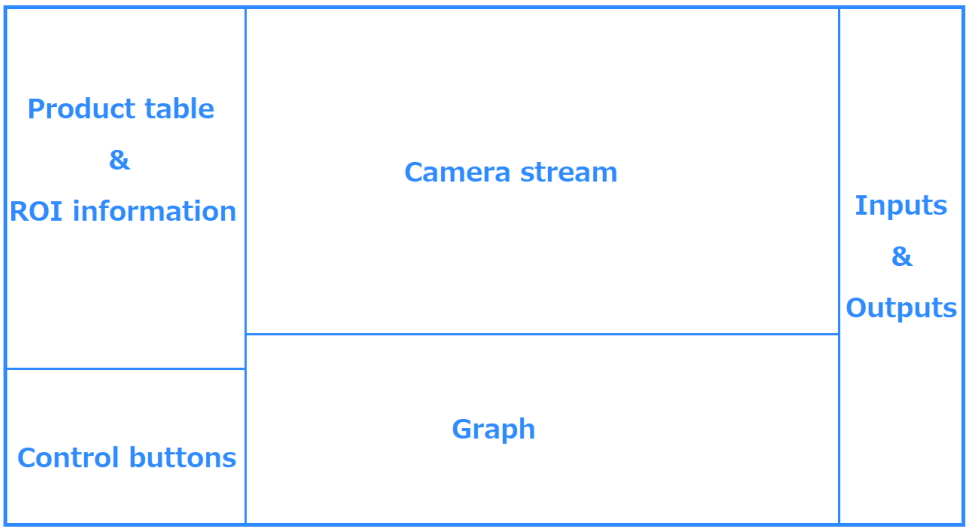
The simplified scheme of the Operator window is displayed in Figure 4.4a.

Admin window

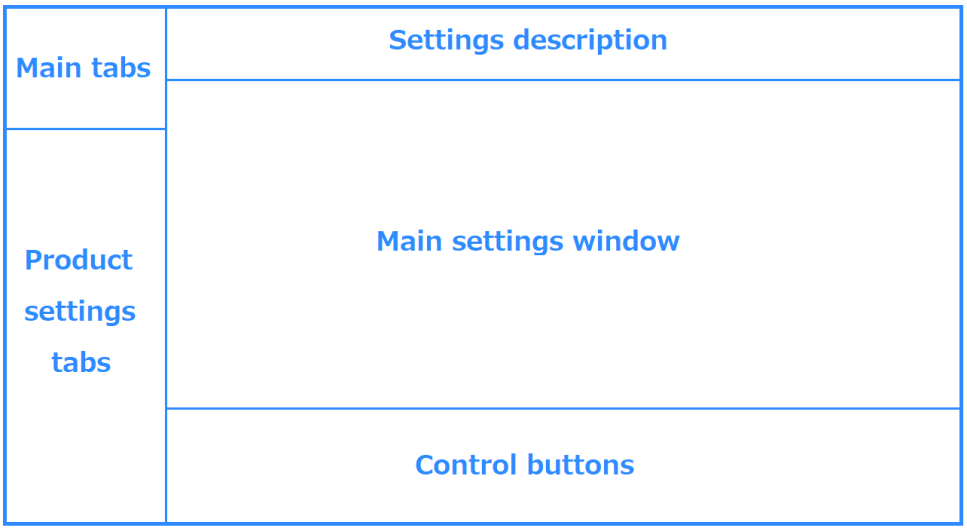
Admin window is the most complex window. It provides the user with many functions for setting the data for the product, create users and set basic camera values. The window changes as the user progresses through settings, providing him with many options while trying to stay informative and intuitive. User can in this window:

- Set basic camera settings
- Create, delete and edit users
- Create, delete and edit products for each user
- Set radiometric settings for the camera
- Set measurement mode settings for the camera
- Set ROI and measurement statistics for the camera
- Set Input and output settings of the camera
- Set Visualisation settings
- Save the settings for the Operator user

The simplified scheme of the Administrator window is displayed in Figure 4.4b.



(a) : Operator window GUI



(b) : Administrator window GUI

Figure 4.4: Simplified scheme of the Operator and Administrator window GUIs, note the space for graph in the Operator window is in the current version unused

Chapter 5

Aplication functions

The last chapter of my thesis will be about explaining the functions of my programmed application. First I will show how I prepared the library for the SMARTIS camera, then I will introduce the main application.

The main application is split in three sections, based on the three main windows user will see when operating the control panel. There is a log in window as a gateway to the application and GUI for two different modes - Operator mode and Administrator mode.

5.1 SMARTIS firmware library development

Before I was able to start working on the main application, I had to get familiar with the SMARTIS libraries and code provided by the company Workswell s.r.o. My first task was to prepare library that will get data from the camera and send it through TCP/IP to my application. This means I had to prepare server side library for the SMARTIS to listen to my requests for the data from the camera.

5.1.1 SMARTIS environment preparation

For the development environment I used Qt Creator and prepared it as stated in the subsection 4.2.3. Qt version for this part of the project was 4.8.6. which is the version the camera supports. I used C++ compiler provided by Workswell s.r.o. and I downloaded debugger for my code.

5.1.2 Library overview

The library is large but easy to understand. The main function of the library is `int runAppServer(void)`. In this method the program waits for the camera to start and then prepares the server side of the connection. After initialisation, I call infinite while loop that is accepting and handling the requests. This part of the library was taken from the tutorial for client-server communication [15] and reworked to the need of the application. The code from the Beej's Guide to Network Programming is free of any licence [16].

Another important method is `int receiveAppMessage()`. In this method the server receives the message and calls appropriate function based on the request. The requested messages must have length of three characters. These characters tell the server, what kind of data are requested, for example message `"GTE"` is a message to request emissivity settings of the TAU2 core of the camera. Some messages may start with three characters follower by space and rest of the request. This kind of message is usually for setting the data to the camera, for example message `"STE 0.90"` requests the camera to set the TAU2 core emissivity to 0.90.

List of all possible requests and messages is written at the beginning of the class in the commentary. If the camera receives wrong request, the error message will be send back. If empty message would be send as request, the communication will stop.

Last important function for the communication is `int sendAppMessage(int message)`. This method is called by the server after successfully receiving the request and serves as a handler. Depending on the result of the function `int receiveAppMessage()` the returning message will be either `"ERR"` for incorrect request, `"OK"` for the setting of variables into the camera or requested data in the form of stream for the client.

The rest of the library contains individual functions for accessing the data of the camera or setting the data in the cameras configuration.

This library was later compiled and imported in a testing unit of the Smart Camera SMARTIS and lent to me for the purpose of this project.

5.2 Login window functions

After the application is started, the Login window is the first screen the user encounters. The window contains three push buttons and two editable text inputs.

The main function that is called is `connect_smartis()`. In this function new TCP Socket is created and connects the application to the server library described in the subsection 5.1.2. The window acts as a client to the server and sends requests. After successful connection, request for the user and password is send. The application receives encrypted list of users and passwords, that needs to be decrypt for the future use. This process is explained in the subsection 5.2.2.

The user can enter the name and the password and press the push button with the text `"Login"` to enter further into the application. Depending on the user type, the window for Administrator or Operator will appear and the Login window will close. Administrator user is only one for the whole application and its user name is *Admin*. Operator is every other user with different name than *Admin*.

By clicking on the button `"Exit"`, the application will end and all connections will be disconnected.

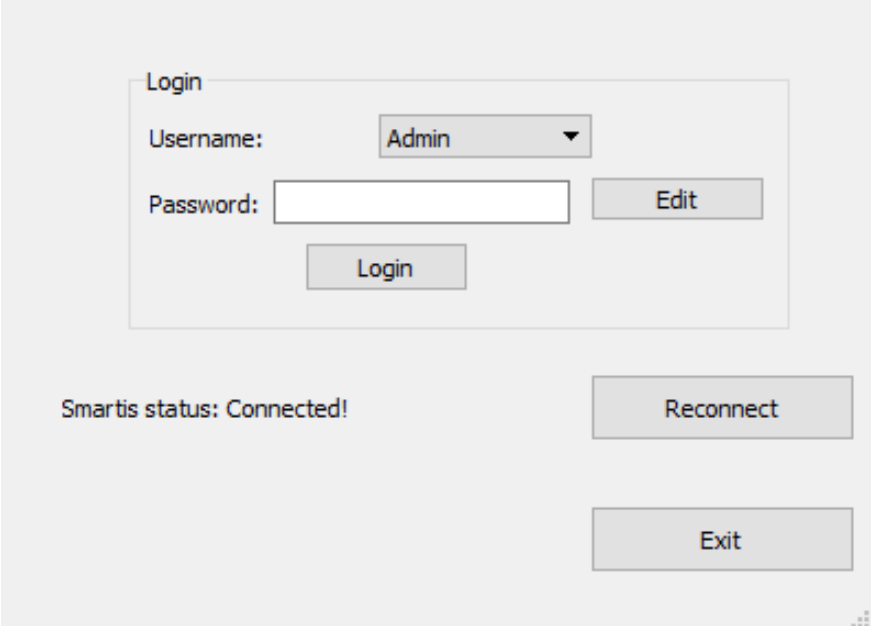


Figure 5.1: The Login window as seen in the application

5.2.1 SMARTIS connection

At the beginning of the connection process to the SMARTIS server, the application loads the IP address stored in the root of the application, in the file called `"ip_settings"`. This address is then loaded and kept as the global variable `global_ip_address` in the `global.h` class.

Creation of the socket and connection to the SMARTIS in Qt is shown on a short example:

```
socket = new QTcpSocket(this);
socket->connectToHost(ip_address, port);


if(socket->waitForConnected(3000))
{
    qDebug() << "Connected!";
}
else
{
    qDebug() << "NOT CONNECTED!";
}
```

The parameters of the function `connectToHost(ip_address, port)` are IP address taken from the global variable and port. List of ports was given to me by the company Workswell s.r.o. for use on the camera SMARTIS. The list of the available ports is:

- 2252 - used for the camera picture stream
- 2252 - used for my implemented library
- 2240 - used for starting measurement of the camera

Different threads of the program connect to different ports during the running of the application. In the future part of the thesis I will be referencing this example in describing the connections.

5.2.2 Password encrypting

The class used in the encryption of the user names and passwords is called `Cryptoclass.cpp`. The code for the algorithm in this library was provided by Workswell s.r.o. to ensure the secure connection to their product.

After creating the member of the class, public key is inserted. This key is known by the application and the camera SMARTIS. Private iv key is then generated and used only by the application. The iv key is used by the algorithm in the hash function for encryption and decryption of the data.

After sending the request for the user login information, the application decrypts the data from SMARTIS. The data are searched for any match with the login name and password in the Login window. When the match is found, the application takes values from the name and password inputs, encrypts them and sends them to the camera with a request to log in that specific user. The camera takes the request, checks the decrypted values with its internal database and logs the user in.

5.3 User window functions

After the user logs in as Operator, the Login window will close and the User window will appear. This window provides environment for the user to select pre-configured settings of a measurement and start / stop it. The window contains Camera image widget, where the camera stream appears. On the left side of the screen the user can observe measured statistics of different ROIs (Region of Interest). In the lower site of the window is information panel about the camera and on the right side are panels of input and output widgets.

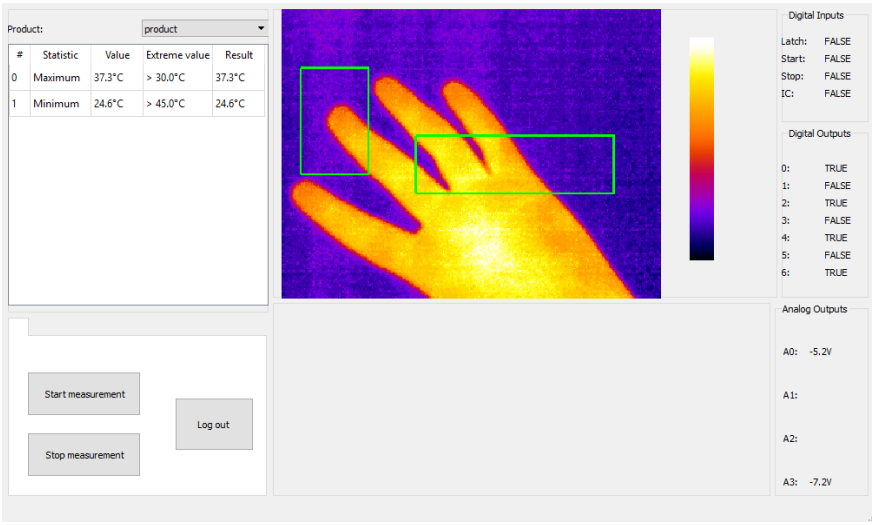


Figure 5.2: Operator window of the application

There are 3 main threads running in the application while the user is logged in the User window:

- Thread for getting the camera stream - `"image_thread"`
- Thread for getting the ROIs geometries for the image - `"roi_thread"`
- Thread for getting the measured data - `"data_thread"`

When the User window is initialised, all threads are created and work independently form each other. Each thread is more closely explained in the following sections.

The Operator is not able to enter Administrator mode without knowing the required login informations. This means Operator user can not configure the measurement and always needs prior assistance of the Administrator user for setting up the measurement.

Once the Administrator creates and saves the configuration of a measurement for the specific Operator user, his assistance is no longer required, only if the user would want to use different settings in the future. Configuration of the camera created by the Administrator is called Product.

The amount of Products for one Operator user is not limited.

5.3.1 Camera image widget

This is the central widget of the User window and it serves to display the stream from the camera. The camera stream is provided by two threads, `"image_thread"` and `"roi_thread"`. The `"image_thread"` operates in the class `"worker.cpp"`

Main function of the image thread is `process()`. At the beginning of this method, the connection to the SMARIS is created, using the port `2252`.

When the camera is connected, the application sends constantly requests to get the data stream from the SMARTIS camera.

Processing of the data

The thermal camera sends array of raw data that needs to be processed before displaying. To be able to work with the stream of data like this, I will need to know basic information about the picture:

- `height` of the picuture
- `width` of the picuture
- format of the data
- desired size of the full frame

Camera send the data in format `uint16_t` and the size of the frame can be received from the SMARTIS sending requests for height and width of the picture. The desired size of the full frame is:
`desired_packetSize = (heigh * width * 2)`

The next goal is to transform the data to usable RGB format, for future displaying. After loading the buffer with the raw data I aimed to get real temperatures from the stream. This is very difficult process with many variables. I was unable to find formula for calculation of the data, used by the company FLIR in their software and I used known transformation of raw data to temperature from the company Workswell s.r.o. This formula can be also searched on the FLIR forums [17]. The final formula I used to transform raw data to the temperature values is:

$$T = \frac{p}{25} - 273.15 \tag{5.1}$$

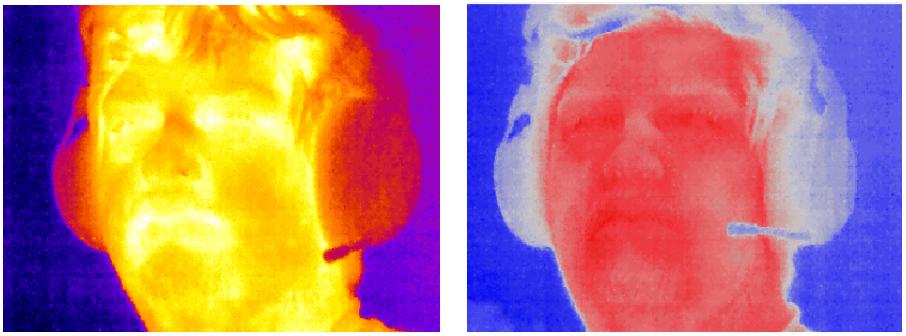
Where T is final transformed temperature and p is the raw value.

Implementation of image palettes

After calculating the temperature data from the raw, I created image palettes to represent the data with RGB values. The image pallet was made as a `.png` picture with the dimensions 255 x 10 px. I loaded the palette as a `QPixmap` and transformed it to `QImage`.

After that I took the calculated data and transformed them on a scale 0-255, where 0 represents the coldest pixel and 255 the warmest. Then, I took the exact color of the pixel from the palette, which has width 255 and used that color to represent the final stream from the camera.

```
// Scale the data as 0-255
for (int i = 0; i<heigh * width; i++)
{
```



(a) : Palette a **(b) :** Palette b

Figure 5.3: Comparison of a image with two different color palettes

```
    int x = (int)(((255-0) * (result_temp[i] - temp_min)) /
    (temp_max-temp_min) + 0);
    if (x < 0)
        x = 0;
    if (x > 255)
        x = 255;

    //take RGB values from the palette on position x,10
    px_color[i] = paleta_image.pixelColor(x,10);
}

int cnt = 0;

//Go through the whole frame and set the color of each pixel
for (int y = 0; y < heigh; y++)
{
    for (int x = 0; x < width; x++)
    {
        image.setPixel(x, y, qRgb(px_color[cnt].red(),
        px_color[cnt].green(), px_color[cnt].blue()) );
        cnt++;
    }
}
```

Roi geometry

The image is not fully complete without displaying the ROI geometries on the top of the stream. To get these geometries form SMARTIS, `roi_thread` creates an instance of the class `roiworker` and starts the method `update_user_gui()`. At the beginning of this method, similarly as with the other *worker* classes, connection with SMARTIS is established. This thread connects to the port 2254 and sends message requesting data about active ROIs in the Product.

Geometry	Number of needed points
Point	1
Line	2
Polyline	2 +
Circle	2
Rectangle	2
Polygon	3 +

Table 5.1: Table of necessary points for drawing specific geometry in Qt with QPainter

Once the full amount of ROIs is known, the method obtains the shape of each ROI. These shapes are represented in the returned message as either `"POINT"`, `"LINE"`, `"POLYLINE"`, `"RECTANGLE"`, `"CIRCLE"` or `"POLYGON"`. After the shapes of each ROI are known, the values of the data-points are needed. Each returned message, depending on the ROI shape, has different length and amount of the points necessary for drawing the specific shape on the scene.

Qt library can directly use the know shape and draw each specific geometry. In the Table 5.1 I will be showing amount of points necessary for drawing each geometry in the Qt, depending on the shape.

Once the geometry shape and the required points are available, the method creates QPixmap with forced alpha channel for to be transparent. On this transparent QPixmap the geometry is finally drawn by QPainter. Geometry is then saved and sent to the User window for future processing.

I decided to paint the ROI with green color to distinguish them on the final image. This could be further improved and the user could get the ability to select the color of each ROI.

Once the ROI QPixmap is created, the `roi_thread` is stopped and the TCP socked is released for the future connections from `data_thread`. The `roi_thread` will not be connecting to the camera again, since the ROI geometries are not changing during the measurement in the User window.

Image data displaying

Transformed and coloured data are send at the end of the image thread to the User window and represented as QPixmap together with ROI geometries QPixmap. The ROI geometries map has forced alpha channel to be transparent and is placed on the top of the coloured image QPixmap. The final QPixmap is then finally drawn by QPainter on the Image Widget. The final picture of the Image Widget is displayed on the Figure 5.5.

5.3.2 Real time data displaying

Last thread running in the User window is `data_thread`. It creates instance of `worker_data.cpp` class and starts method `get_data()`. Before this func-

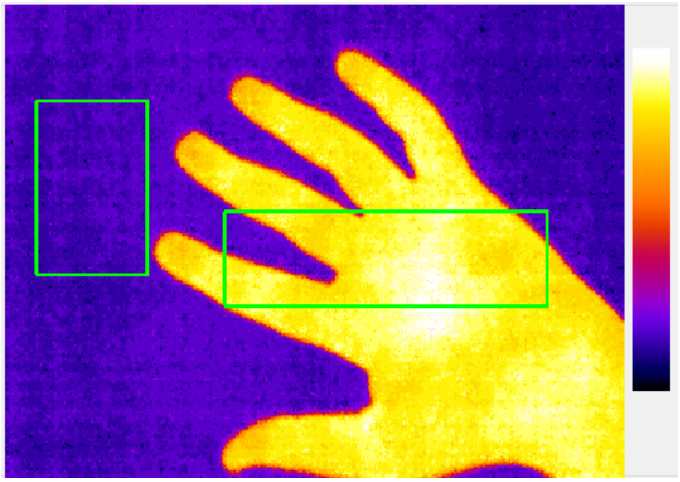


Figure 5.4: Final image after combining coloured picture with the ROI geometries

tion establishes connection with the SMARTIS, it needs to wait for release of the TCP port 2254, which will be used for the communication. Once the `roi_thread` releases the resources, the communication channel is created with the camera.

The main task of the function is to send requests for all the measured data from SMARTIS and save them in respective global variables for future use. Unlike `roi_thread`, the `data_thread` continues to request data during the run of the program and updates the received values for displaying.

Product table

Product table is the main widget for displaying the measured data for each ROI of the product. On the top of the product table is a select box with the option to switch Product for the measurement.

Each line of the table represents one ROI. Values, that are represented in the table are dynamically changing based on the measurement from the camera.

#	Statistic	Value	Extreme value	Result
0	Maximum	28.2°C	> 30.0°C	37.8°C
1	Minimum	23.2°C	> 45.0°C	22.6°C

Figure 5.5: Example of the product table during measurement

Input and Outputs widgets

Thermal camera SMARTIS has 4 analogue outputs, 4 digital inputs and 8 digital outputs. The inputs are used to start or stop the measurement, the outputs represent results of the measurement.

Each ROI has its own statistic settings and its own analogue outputs. When the conditions of one ROI are met, multiple digital outputs can trigger. The final result of the digital output is the combination of ROI settings and logical table for the combination of digital outputs. I will explain more about this settings in the Administrator window.

The input and output widgets are situated to the right side of the application. Digital inputs and outputs are represented by text value `"TRUE"` or `"FALSE"` and the analogue outputs display the value of the signal.

Similarly as in the Product table widget, Input and Output widgets are dynamically updated with every cycle of the `worker_data` main function, `get_data()`.

5.4 Administrator window functions

Once the Administrator user logs in the application with the name Admin and correct password, the Login window will close and the Administrator window will appear. The main function of this window is to provide tools for the Administrator user to prepare settings of the measurement that will be used by the Operator user. The list of settings of each single measurement is called Product.

The Administrator has an option to change basic settings of the camera for every user, create and edit users and create and edit their products.

The Administrator window contains many input widgets and select boxes to allow configuring all the functions of SMARTIS. Once the Administrator starts editing selected product, he needs to go through all the settings and finally decide to either save the changes or discard them.

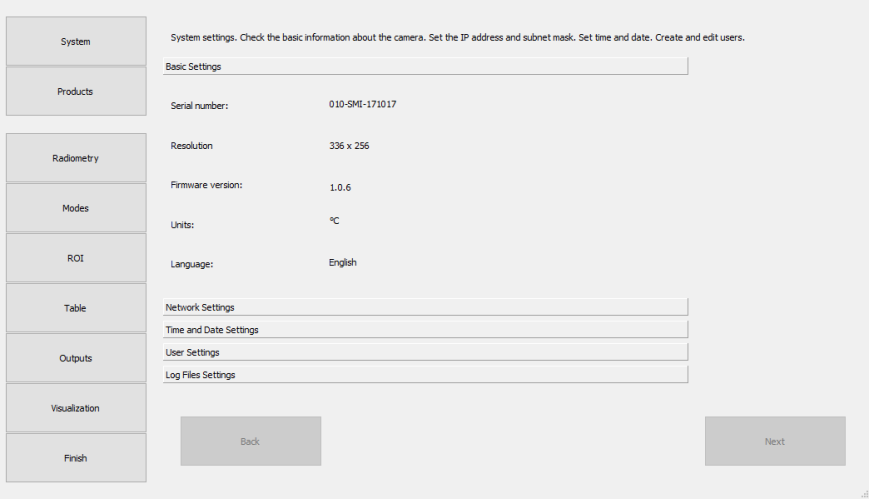


Figure 5.6: The first screen of the Admin window

5.4.1 General settings

The General settings of the Administrator window is the collection of all possible settings for the camera, that are not exclusive for each Product.

The General settings screen contains five tabs of different options that can be customized. At one time only one options tab is active and the Administrator can cycle through them freely. The tabs are:

- Basic settings
  - It is a collection of camera parameters and settings about used units and language of the application. The language can not be changed at the time but the goal will be to allow this option in the future.
- Network settings
  - Here the Administrator can change the network settings of the camera. This section tab is further explained in subsubsection 5.4.1.
- Time and Date settings
  - This option allows settings of date and time of the internal camera clock.
- User settings
  - The Administrator has the option to add, delete or edit other users. This section tab is further explained in subsubsection 5.4.1.
- Log Files settings
  - In this tab the Administrator can erase the internal logs of the camera or allow automatic log deletion, once the memory is full.

Network settings

The Administrator is able to change the IP address and the subnet mask of the camera. In the upper part of the tab current settings of the network are displayed. The window contains two inputs for each setting respectively.

Once the user clicks the push button with the option `"Set Network` the edited network settings will be saved and the camera will apply them. This means the camera will reboot and needs to be reconnected. The Administrator mode will log out and the user will be redirected to the Login window, where he needs to reconnect the camera after it reboots.

Once the camera is successfully connected, the Administrator can again log in.

User creation

The User settings tab is split into two parts. On the right side, the Administrator can create new user by entering the name of the user, password and confirmation of the password. If the password and confirmation does not match, the user will not be created.

Once the Administrator creates the user, the name and the password will be encrypted and send to the SMARTIS for the internal database of the camera.

On the left side of this tab the Administrator can edit password of existing users or delete the users. To edit the user, Administrator selects the name from a combo box and enters new password with confirmation.

If the Administrator clicks `Delete` button, the selected user form the combo box will be deleted from the database. User `"Admin"` can not be deleted.

The combo box of user names dynamically changes with creation and deletion of users.

5.4.2 Product creation

When the Administrator selects option `"Products"` he will be redirected to the Product creation tab. Here he can select an operator and create new Product as a set of new settings for measurement.

Existing products are visible in the table bellow. The Administrator can either delete or edit the products from the table. The table refreshes set of product depending on the selected user.

5.4.3 Product settings

Once new Product is created, the Administrator needs to go through all of the settings to prepare the measurement. At the beginning of the product editing, the application sends requests about every possible setting of the product and pre-loads all forms of the application with the received values.

Next sections will briefly explain each setting and implementation.

5.4.4 Radiometry settings

Radiometry tab contains options for setting the emissivity, atmospheric temperature, reflect temperature, temperature range and synchronous image correction limit of the SMARTIS camera.

Each setting is handled using editable text inputs. Camera temperature range option is selected from the combo box, since the possible temperature range is strictly given by the manufacturer and can not be freely changed.

5.4.5 Camera mode settings

The upper part of this tab contains combo box with the possible four camera modes: Frame Measurement, Start-Stop Measurement, Continuous Measurement and Non-Trigger Measurement. The modes are further explained in the subsection 2.1.5.

Once the mode is selected, further options become available for specification of each mode like type of trigger, sampling interval or start delay. These options are handled through text inputs.

On the right part of the screen can Administrator select when will the triggers activate by setting either logical high or logical low option for each trigger. The user will select the option from combo box.

5.4.6 Region of interest selection

In this tab the user can create simple ROI of rectangle shape for the product in the middle of the screen by clicking the `"Create ROI"` button.

For the purposes of the diploma thesis this simple option was sufficient and further options to the ROI creation will be available in the future version of the application.

5.4.7 Statistic settings

The statistic settings is list of configuration for each ROI. The configurations say, when the ROI produces signal and for what output will be the signal produced.

This window contains two main tables, for ROI statistics configuration and for the output selection. When the Administrator opens the Statistic settings, the application sends request to get every configuration value of each ROI from this Product and fills up these tables. Each line in the tables represents one ROI. The statistic configuration table is made from combination of combo boxes and editable text inputs. When the conditions of the ROI configuration are met during the measurement, one ROI can produce multiple digital outputs (1-7) and one analogue output.

There are only four analogue outputs of the camera so this option is represented by combo box. Digital outputs are represented by check boxes, and Administrator will select which digital output should be produced after the ROIs requirements are met based on the configuration. If the user selects same digital output for multiple ROIs, he can further specify the logical functions under the check box table. These functions say, what logic will be used during the generation of the digital outputs while evaluating each ROI.

5.4.8 Output settings

The Output settings are represented by two table, for analogue outputs and for digital outputs respectively. After the statistic settings are finished the output tables is filled based on the selected configuration.

For analogue settings, the user can select the type and range of each output represented by combo box. The minimal and maximal temperature that will be coresponding to the measured data and the range of the output is represented by text input.

In the digital output settings, Administrator can enter pulse length and type of signal edge that the output will produce.

5.4.9 Visualisation settings

The visual settings is a combination of yet not used options for future graphical visualisation in the User Window. The Administrator can select up to 4 different ROIs that will be represented. These settings will be saved in the camera but the work with the values received from SMARTIS is not yet supported in the panel application.

5.4.10 Saving of the settings

When the Administrator finishes all the configurations, he can decide from one of the following:

- Save changes and log out
- Save changes and stay logged in the Administrator mode
- Save the changes and log in the Operator mode as selected user
- Discard the changes and log out

After saving the changes the application sends requests containing settings of the configured Product and SMARTIS saves them to the internal file. The user will be either redirected to the beginning of the Administrator window or logged out.

Administrator can also save the settings and log in as the operator to start the User window.

When the settings are discarded, the Administrator is redirected to the login page.

5.5 Extra functions

In the last part of this chapter, I would like to introduce the remaining functions I created for the application.

5.5.1 Keyboard

For comfortable work on the remote control touch panel I had to use virtual keyboard for entering all the different configuration values for the camera.

I programmed the keyboard as a set of push buttons with the option to erase input, cancel the input and save it.

The keyboard is called every time, the user clicks to edit specific input field. When the keyboard is called, the value that was in the input field is stored and remembered. If the user decides to save the value from the keyboard, the input is replaced, otherwise the input stays the same as before editing.

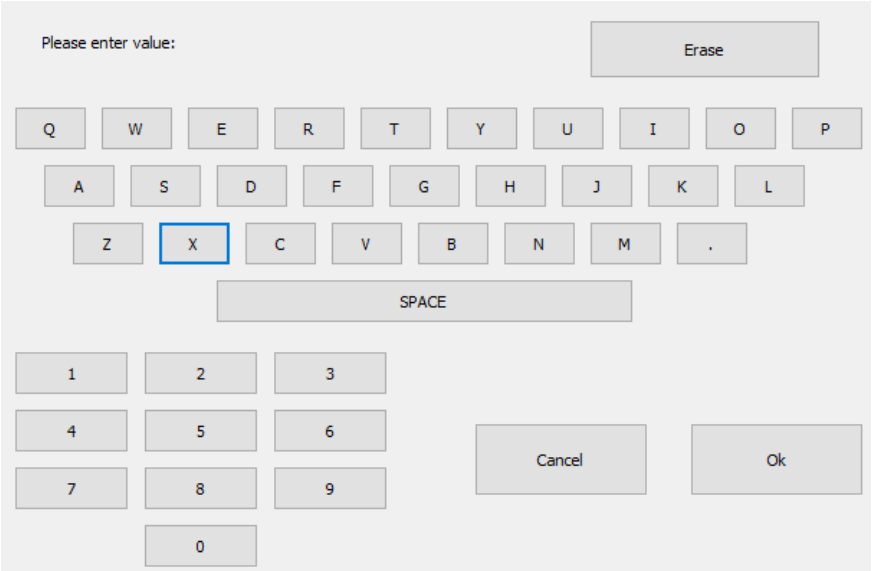


Figure 5.7: Instance of Keyboard class - virtual input for the touch screen panel

5.5.2 Application logging

The application is logging all outputs into the log file. From the log files the developer can see the activity of the user and errors that could potentially appear during the testing and using of the application.

If the log file is manually deleted, the application will create it automatically with the next start up.

The log file is not erased after new start of the application and continues to save logs for the future inspection.

5.5.3 Documentation

For the project to be successful and sustainable, good documentation is needed. During the development of the application I did my best documenting all the necessary parts of the code for the future development or possible corrections.

I automatically generated the documentation with Doxygen [18]. It is a tool for generating the documentation from the comments of the program. The documentation is together with the code on the attached CD.

Chapter 6

Conclusion

In the diploma thesis I was choosing and preparing a suitable platform for a remote panel, that will be used as a control device for the Thermal Smart Camera SMARTIS. After selecting the platform, I needed to prepare a library for communication with the SMARTIS camera and write the application for the control panel.

For my platform, I decided to use powerful single board computer NanoPC-T3 with ARM processor. While preparing the platform for development, I encountered problems with compiling the code directly on the board and I had to reinstall the operating system and select different approach of programming. The final operating system I was using on the NanoPC-T3 was FriendlyCore(64bit) supporting Qt 5.9.1. This system is based on Ubuntu core without X-Windows.

After installing and testing the operating system I connected the camera and the touch panel to the board while running simple application to ensure the communication is working.

In the next part of the project I prepared development environment for the application and the needed SMARTIS library. I described the process of configuration in the thesis for future reference or installation on another device. As a development environment I used application Qt Creator with Qt libraries. I was writing my applications in C++ on a host computer while cross-compiling the code for the final platform.

The library for the communication with the SMARTIS is a server application with many possible requests, that are handled within the camera functions and sent to the remote panel application. I tested the library and the functions and created handlers for error inputs. This prevents the application to send incorrect messages and potentially damage the settings inside of the camera.

The application of the remote platform contains login screen, where I used encryption of the login information to make the process more secure and prevent future tempering with the settings from unverified users.

I created two modes for the control panel, Administrator and Operator mode. Each mode has various separate functions and different type of user will be accessing each mode respectively.

Main function of the Operator mode is to display data from the camera

and observe the results. The thermal camera was sending raw data, which I needed to process and produce visible thermal image. After processing the image, I used color palettes and displayed the Image in the Operator window. Alongside the stream from the camera, I created a function to dynamically display the measured data.

In the Administrator mode I created an environment allowing the user full control over the configuration of the camera. This mode contains many options to change the settings and save it for the Operator. List of configuration of the camera is called Product. The mode also contains ability to securely create new Users and edit old ones.

In the remote application I also made an option to log the data from the application and save it for future inspection.

During the development of the application I was creating documentation for the code and instructions for installation of the system and all used platforms.

Alongside the work on the application I was running tests for inputs and outputs of the application. After development, I successfully cross-compiled the final application on the NanoPC-T3 and made stability test for this platform. I also made simple measurements and ensured that all setting are working.

The application is fully functional and can be used to control the SMARTIS thermal camera. In the future I would like to work on the remote control panel more and introduce new options for the configuration that the camera would support. The first thing is going to be live graph of the data for the camera, and an option to freely insert Regions of Interest (ROI) to the picture through the panel.

I would also like to test the platform more and improve the graphical user interface based on the feedback from wider range of people.

Appendix A

Final product

At the end of the thesis I would like to summarize the final product that I developed. It consists of NanoPC-T3, 10" touch screen panel LCD-HD101 [19] and Thermal smart camera SMARTIS. The final product can be seen on the Figure A.1.

- NanoPC-T3
  - Processor Samsung S5P681 Octa-Core Cortex-A53, 1.4GHz
  - RAM 2GB DDR3
  - Internal storage 8GB eMMC
  - 3x unused USB 2.0 port
  - 1x unused microSD card port
- LCD-HD101 touch panel
  - 10" screen
  - Resolution 1280x800
  - Powered by the board, connected on LCD port
- SMARTIS camera
  - Connected through Ethernet cable to the boards Ethernet port
  - Communicating through TCP/IP protocol with the board
  - Power Supply 18VDC - 32VDC
- Application running on the board.

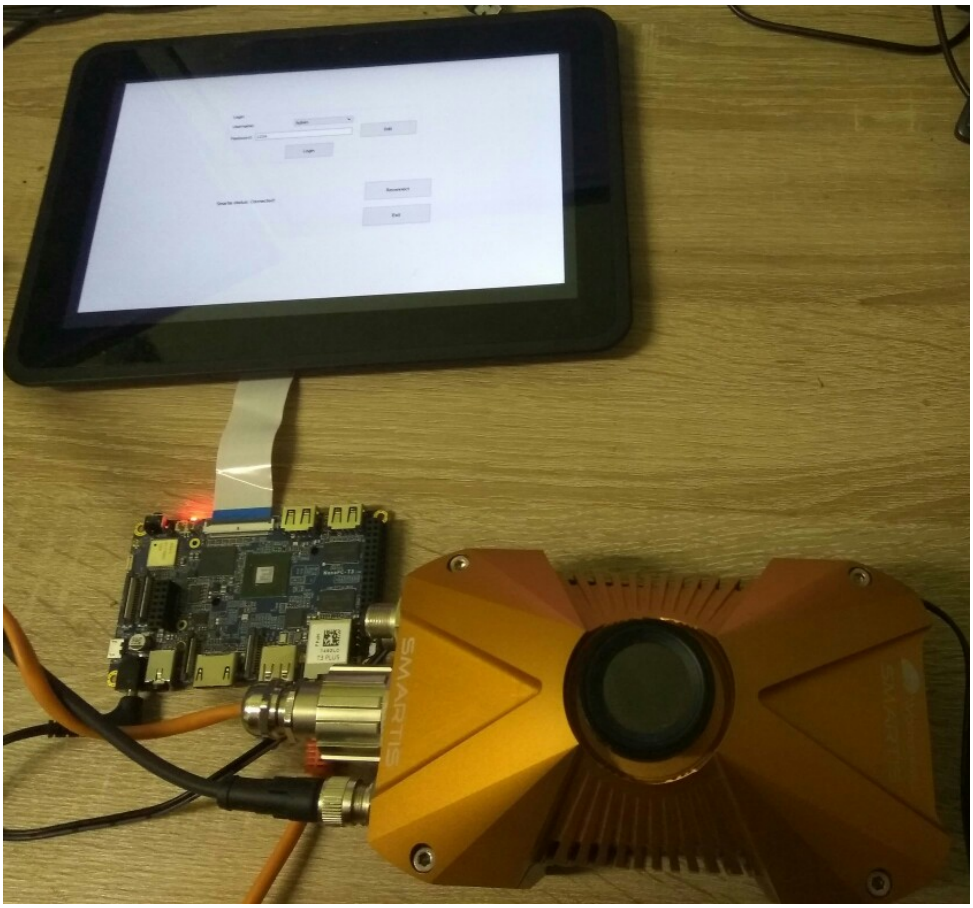
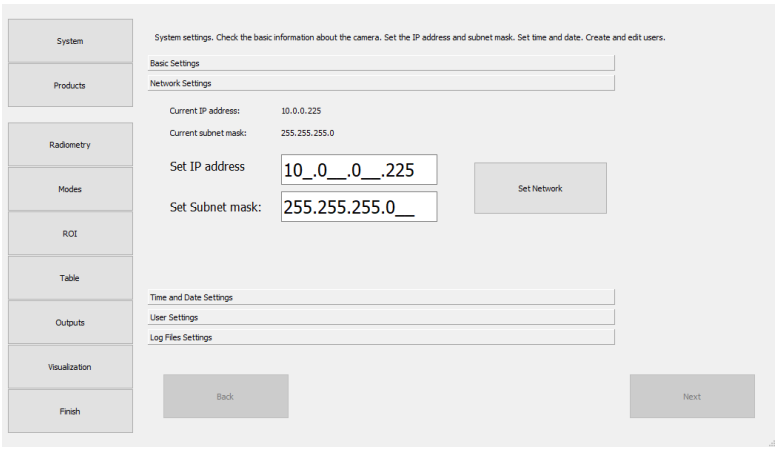


Figure A.1: Picture of the finished product, NanoPC-T3 running the application, SMARTIS and LCD touch panel connected to the board

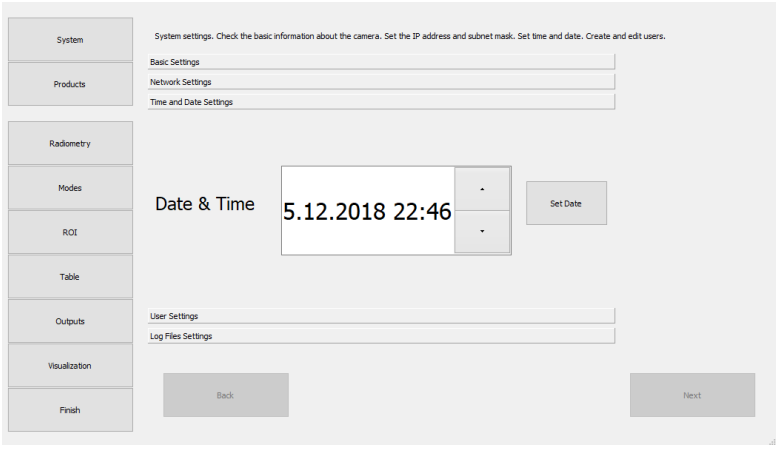
Appendix B

GUI of the application

This part contains more pictures from the application with short description.

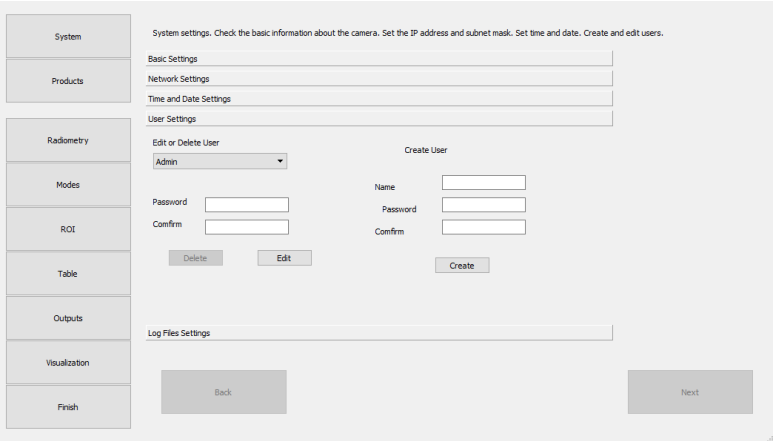


(a) : Network settings

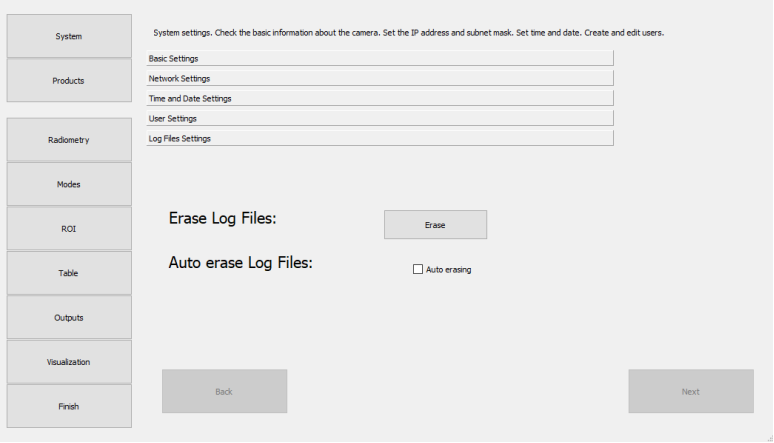


(b) : Time and date settings

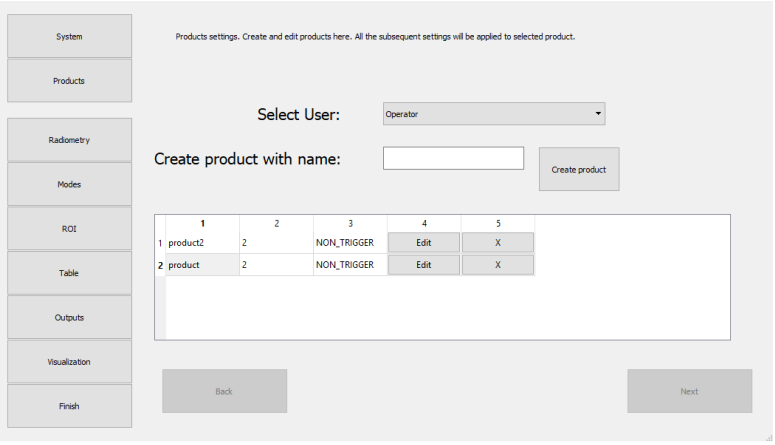
Figure B.1: Pictures form the General settings of the Administrator mode



(a) : User edit settings

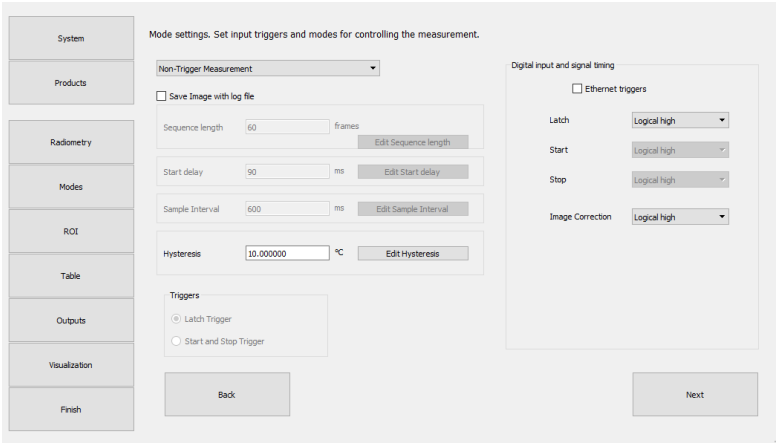


(b) : SMARITS logs settings

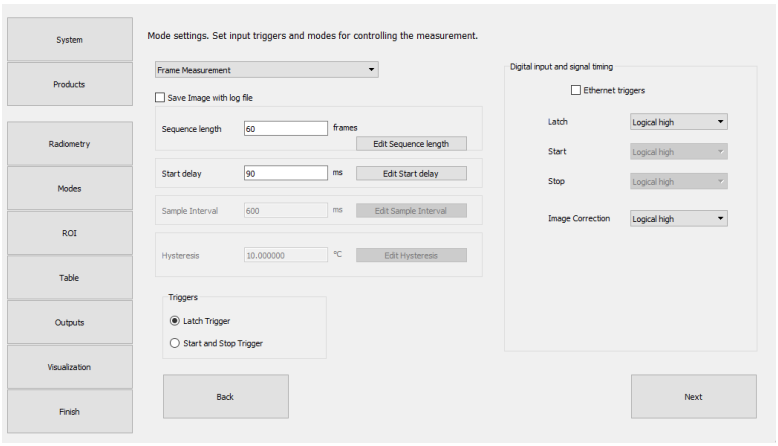


(c) : Product selection settings

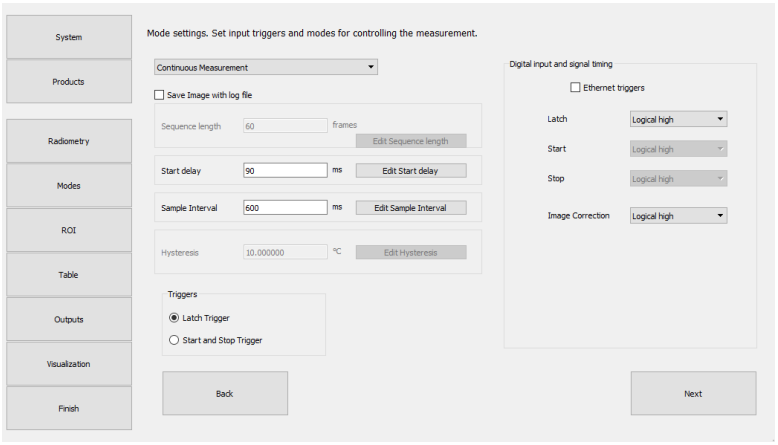
Figure B.2: Pictures form the General settings of the Administrator mode and Product selection settings



(a) : Non-Trigger measurement

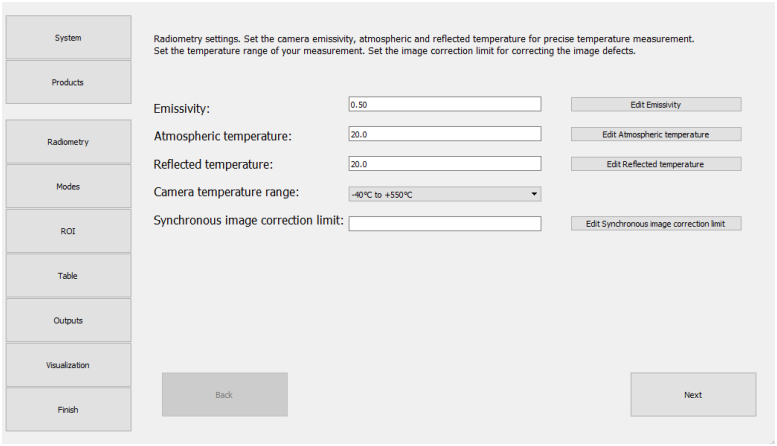


(b) : Frame measurement



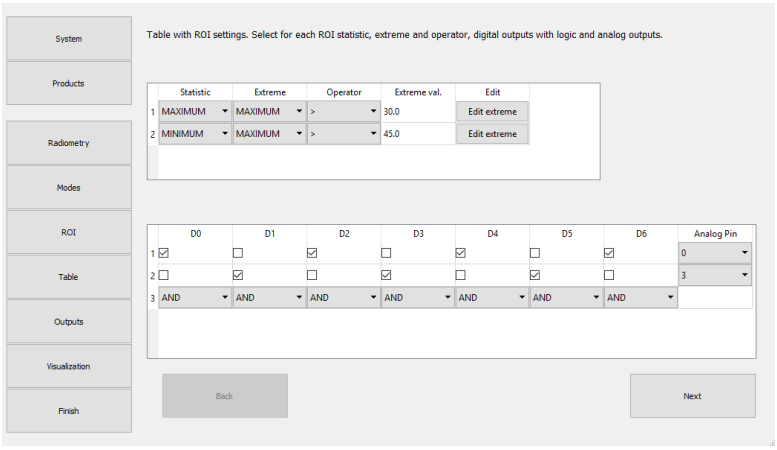
(c) : Continuous measurement

Figure B.3: Pictures form the Mode settings of the Administrator mode



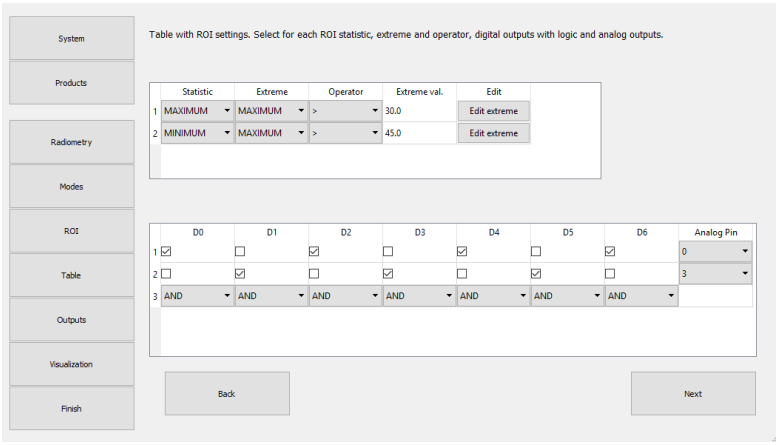
(a) : Radiometric settings

(b) : ROI settings

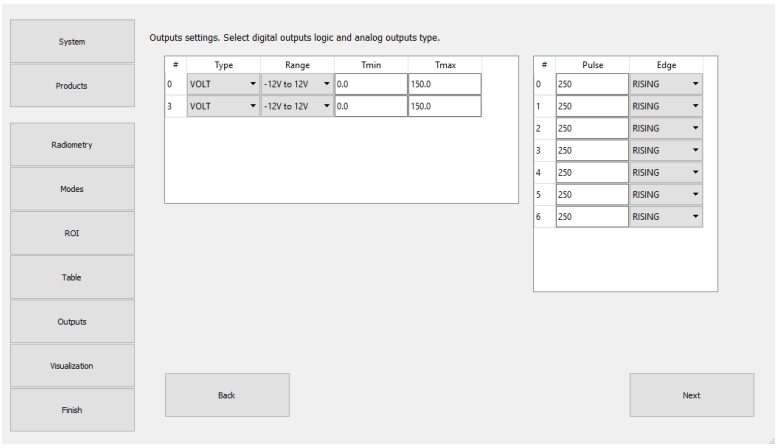


(c) : Statistics settings

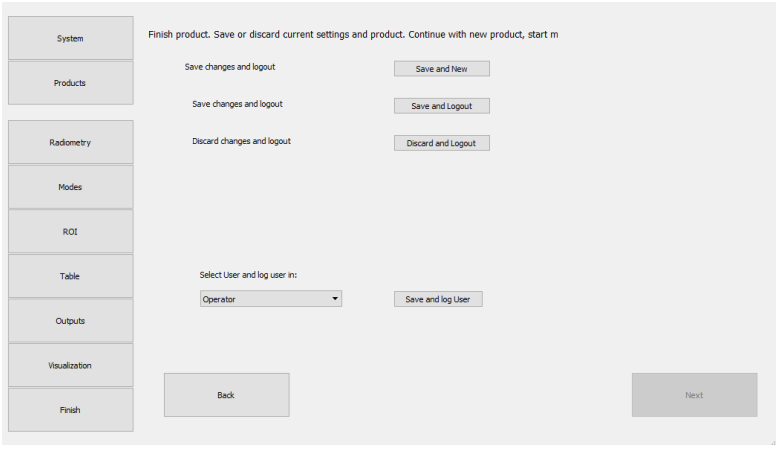
Figure B.4: Pictures from the configuration of selected Product by the Administrator



(a) : Output settings



(b) : Visual settings



(c) : Finalization of the settings

Figure B.5: Pictures from finishing of the configuration by the Administrator user

Appendix C

CD attachment

The thesis contains CD attachment containing all codes and developed libraries. Documentations, pictures and the thesis source code is also part of the content. On the CD, you can find `README.txt` file, containing information about the structure of the content.

Appendix D

Bibliography

[1] Article *Global Thermal Imaging Market worth over $10bn by 2024*, https://www.gminsights.com/pressrelease/thermal-imaging-market .

[2] FLIR, *Flir ThermoVision CM*, https://www.flir.com/products/thermovision-cm/ .

[3] InfraTec. *InfraTec. PRESS-CHECK*, https://www.infratec-infrared.com/thermography/industrial-automation/quality-control-for-press-hardening-press-check/

[4] Workswell s.r.o, *Workswell SMARTIS*, https://www.workswell-thermal-camera.com/thermal-imaging-camera-smartis-industrial/

[5] Friendly Elec, *NanoPC-T2 web page*, http://wiki.friendlyarm.com/wiki/index.php/NanoPC-T2 .

[6] MYiR, *Rico Board web page*, http://www.myirtech.com/list.asp?id=510 .

[7] Friendly Elec, *NanoPC-T3 web page*, http://wiki.friendlyarm.com/wiki/index.php/NanoPC-T3 .

[8] Friendly Elec, *EFlasher web page*, URL: http://wiki.friendlyarm.com/wiki/index.php/EFlasher

[9] Friendly Elec, *Google drive containing supported OS*, URL: https://drive.google.com/drive/folders/1hGZVao7vpj_0BtaN-_HZPs2GNr35E0if

[10] SourceForge, *Win32DiskImager download page*, URL: https://sourceforge.net/projects/win32diskimager/

[11] LXQt, *The LXQt web page*, URL: https://lxqt.org/

[12] Friendly Elec, *FriendlyCore S5P6818 source files*, URL: http://112.124.9.243/qtsdk-friendlyelec/s5p6818/

[13] Qt, *Qt web page*, URL: https://www.qt.io/

[14] Qt, *Qt Creator web page*, URL: https://www.qt.io/qt-features-libraries-apis-tools-and-ide/

[15] Beej's Guide to Network Programming, *A Simple Stream Server*, https://beej.us/guide/bgnet/html/multi/clientserver.html.

[16] Beej's Guide to Network Programming, *Copytight and Distribution*, URL https://beej.us/guide/bgnet/html/multi/intro.html#copyright

[17] FLIR ExifTool Forum, *Extract binary data from FLIR radiometric jpg*, URL: http://130.15.24.88/exiftool/forum/index.php/topic,4898.60.html

[18] Doxygen web page, URL: http://www.doxygen.nl/

[19] Friendly Elec, *LCD-HD101 touch panel*, URL: http://wiki.friendlyarm.com/wiki/index.php/LCD-HD101