VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

HOCHSCHULE FURTWANGEN UNIVERSITY, GERMANY

B.BRAUN MELSUNGEN AG, GERMANY

EVALUATION OF THE WI-FI TECHNIQUE FOR USE IN A NAVIGATED ORTHOPEDIC SURGERY

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE AUTHOR	Bc. JINDŘICH TRUHLÁŘ
VEDOUCÍ PRÁCE SUPERVISOR	prof. Ing. IVO PROVAZNÍK, Ph.D. Dipl. Inf. CHRISTIAN WEHRLE (BBraun) prof. Dr. rer. nat EDGAR JÄGER (HFU)

BRNO 2012

BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering and Communication

Department of Biomedical Engineering

Diploma thesis

Master's study field

Biomedical Engineering and Bioinformatics

Student: Bc. Jindřich Truhlář ***ID:*** 106103

Year of study: 2 ***Academic year:*** 2011/12

TITLE OF THESIS:

Evaluation of the Wi-Fi technique for use in a navigated orthopedic surgery

INSTRUCTION:

1) Carry out a research about the available wireless technologies and restrictions related to use in medical environment. Research should include problems of acceptance of WLAN in OR in different countries, medical accreditation, suggestion of service concept, etc. 2) Explore possible ways of wireless connection of devices used in OrthoPilot navigation system (navigation camera, wireless remote control, ultrasound, tablet/smartphone, second monitor, KIS). Test different hardware/software solutions for wireless image streaming and based on analysis choose the optimal one. 3) Perform configuration and tests with access point. 4)According to the chosen approach, build up a working system with several devices. 5) Discuss the behaviour and reliability of such system and suggest further possible development.

REFERENCE:

[1] MACKAY, S.;WRIGHT, E.;REYNDERS, D.;PARK, J.: Practical Industrial Data Networks: Design, Instalation and troubleshooting, Newnes, Oxford, 2004. ISBN 0-7506-5807-X
[2] O' ROURKE, B.: Wireless HD Video Technologies: WHDI, WirelessHD, and WiGig Try to Create a Market, In-Stat-an NPD Group Company, 2012. Product No: IN1205134MI

Assigment deadline: 6.2.2012 ***Submission deadline:*** 31.7.2012

Head of thesis: prof. Ing. Ivo Provazník, Ph.D.

Consultant:

prof. Ing. Ivo Provazník, Ph.D.

Subject Council chairman

Abstract

Following text focuses on use of wireless technologies in OrthoPilot navigation system developed by B.Braun company. Description of OrthoPilot software is followed by overview of available wireless technologies highlighting their both advantages and disadvantages. Practical part consists of two main parts, mostly dealing with electronic circuits. First part describes development process of camera-wireless printed circuit board which substitutes currently used RS-422 cable connection between PC and stereo camera. Part of this chapter covers programming in C++ in order to make interface compatible with the rest of current OrthoPilot software. Second bigger part deals with remote controller development using prototyping board mikroMedia for XMEGA. Besides electrical circuits design, chapter describes also software part - microcontroller programming in C language. Thesis is concluded by discussing system limitations and ideas for future development.

Keywords

OrthoPilot, wireless, WLAN, Bluetooth, stereo camera

TRUHLÁŘ, Jindřich *Evaluation of the Wi-Fi technique for use in a navigated orthopedic surgery*: master's thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Biomedical Engineering, 2012. 118 p. Supervised by Dipl. Inf. Christian Wehrle, prof. Dr. rer. nat Edgar Jäger, prof. Ing. Ivo Provazník, Ph.D.

Declaration

Acknowledgements

First and foremost I express my sincerely thanks to God, as I would never have been able to finish my thesis without His guidance. I´m especially grateful for being given a chance to write my thesis in abroad and to acquire many new experiences. My dearest thanks are given also to my international friends, especially to Ana Maria Cozar Infantes for their encouragement and patience throughout my thesis. When expressing my thanks I must mention my dearest parents and sisters for the family background, which is for me of primary importance. In my daily work I have been blessed with a friendly behavior of all my colleagues and time which I could spend in the Aesculap AG Company. Special thanks to my company supervisor Dipl. Inf. Christian Wehrle as well as prof. Dr. rer. nat Edgar Jäger for their academic supervision and useful advices.

Table of Contents

List of Figures

List of Tables

Abbreviations

ENT	Entwicklung Navigation Technologies
OR	Operation room
PCB	Printed Circuit Board
RC	Remote Controller
IR	Infra Red
NDI	Northern Digital Inc.
RB	Rigid Body
GND	Ground
RF	Radio Frequency
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
LAN	Local Area Network
BT	Bluetooth
UWB	Ultra Wide Band
FPS	Frames per Second
SPA	Serial Port Adapter
AC	Access Point
ISR	Interrupt Service Routine
VCP	Virtual COM Port

1 Introduction

1.1 Company portfolio

The following thesis was created in Aesculap® company, a division of B. Braun Melsungen AG in Germany, which is one of the very important suppliers of products and services for all surgical and core processes in Europe. Aesculap's headquarters are located in Tuttlingen (Germany) currently employing over 2 500 employees. The company was established in 1867 focusing on the production of surgical instruments. In 1976 B. Braun AG acquires a majority shareholding of Aesculap which leads to incorporation as a division of B.Braun in 1997. Nowadays the product range includes surgical instruments for open or minimally invasive approaches, implants (e.g. for orthopaedics, neurosurgery and spinal surgery), surgical sutures, sterile containers, motor and navigation systems as well as products for cardiology. [1]

Since 1994 department for Development of Navigational Technologies (ENT[1]) in cooperation with University of Grenoble in France focuses its work on navigation used during orthopaedic surgical interventions. In 1999 the company introduced the first generation of orthopaedic, CT-free navigation system OrthoPilot®. This was developed to achieve the main surgical goal of perfect implant alignment without having to perform preoperational examinations or take radiation-intensive and expensive CT or MRI scans. Since the beginning OrthoPilot is a helpful tool for physicians to perform precise bone execution and accurate implant positioning and last but not least to shorten intraoperative procedure time rapidly. Different software applications cover wide range of surgical interventions e.g. TKA (Total Knee Arthroplasty), THA (Total Hip Arthroplasty), HTO (High Tibial Osteotomy), ACL (Anterior Cruciate Ligament Replacement). [2]

Figure 1 Aesculap AG, Tuttlingen, Germany

[1] German abbreviation Entwicklung Navigation Technologies

1.2 Motivation

OrthoPilot is a system consisting of more components (PC unit, camera, footswitch, ultrasound device, second monitor), as will be described later in details, which are currently bound with cables and fixed to a massive trolley making the whole system heavy and hardly manipulative, predetermining the device to stay at one limited place/building only. The new idea and efforts in ENT department are to substitute wires by commonly available wireless technologies which are almost fully comparable in terms of transfer speeds and reliability to provide easy manipulation, and increase security in operation room (OR) by removing cables from the floor. System like this would further offer portability by allowing user to detach and mount particular components easily from the light-weight, accumulator-supplied trolley to another. This would be appreciated especially in small, countryside hospitals where one surgeon takes care about the whole region and simple transporting of most important and costly components (camera) to another city/village would increase availability of health care while lowering customer´s expenses rapidly. By omitting meters of cables and using wireless technologies instead, the whole system does no longer need to be mounted all together, but gives surgical team freedom of movement and possibility to place particular devices in OR accordingly to their wishes and needs and make their work more effective and convenient.

The whole project was covered by motto "OrthoPilot Goes Wireless" and is already in progress. Until now there has already been some development done but there is still a lot of questions which need to be answered, hardware/software to be repaired, improved or newly developed from a scratch. All these tasks became a target and challenges of my thesis and the results are being presented here.

Particularly my work at company dealt with:

- Theoretical study and research in order to get into and understand particular devices, principles, technologies etc.
- Testing different possibilities of wireless desktop cloning (image streaming from one computer to second computer/monitor).
- Coupling wirelessly camera with PC unit. Testing prototype of printed circuit board (PCB) and creating new one.
- First tests with Wi-Fi technology wireless modules. Writing source code for controlling the interface and implementing this into test application software.
- Development of remote control (RC) prototype using ATXMEGA microcontroller.

2 OrthoPilot

Following chapter describes advantages of using navigated surgery, OrthoPilot navigation system, its components, and basic principles of navigated surgery at all. Interfaces between particular components and communication standards are further described in detail as it is crucial for next understanding.

In traditional surgery, surgeons must manually measure bone lengths, angles, ligament tensions, and the like. The tools available to do so, though good, aren’t always exact. A mismeasurement, deviation in cutting angle, or inaccurate implant alignment can undermine any procedure and result in long-lasting pain for the patient and increased wear on the implant itself. In a recent international study[2] comparing 821 patients, roughly 3 of every 10 individuals who received implants through traditional methods were found to have a prosthesis, or implant, out of optimal alignment by more than 3 degrees or more. And 1 of 10 was found to be out of alignment by 5 degrees or more. On the other hand, patients who received knee implants in procedures supported by OrthoPilot enjoyed a 300% improvement in implant alignment accuracy. [3]

2.1 Status Quo

OrthoPilot *Next Generation FS101* is the newest model of OrthoPilot navigation systems successfully following its ancestor model *FS100* which became very popular and since 2003 more than 550 pieces have been sold and are being successfully used in clinical departments all over the world. In 2008 the *FS100* system was awarded a world prestige prize for best design “IF Product Design Award” in category “Medicine and Healthcare”. [5]

Next Generation FS101 was released in 2009 and has brought many upgrades especially easier manipulation, maintenance, user´s comfort and higher accuracy thanks to new stereo camera. System is adaptable to user´s needs offering mounting or dismounting additional modules (Ultrasound, System Control Unit).

[2] R.K. Miehlke, H. Kiefer, S. Kohler, J.Y. Jenny, W. Konermann, Navigation in Knee Arthroplasty, Abstract, Combined ERASS & ARO-Congress, 20-22 June 2002, Berlin, Germany – Abstract Book Nr. T 5.4

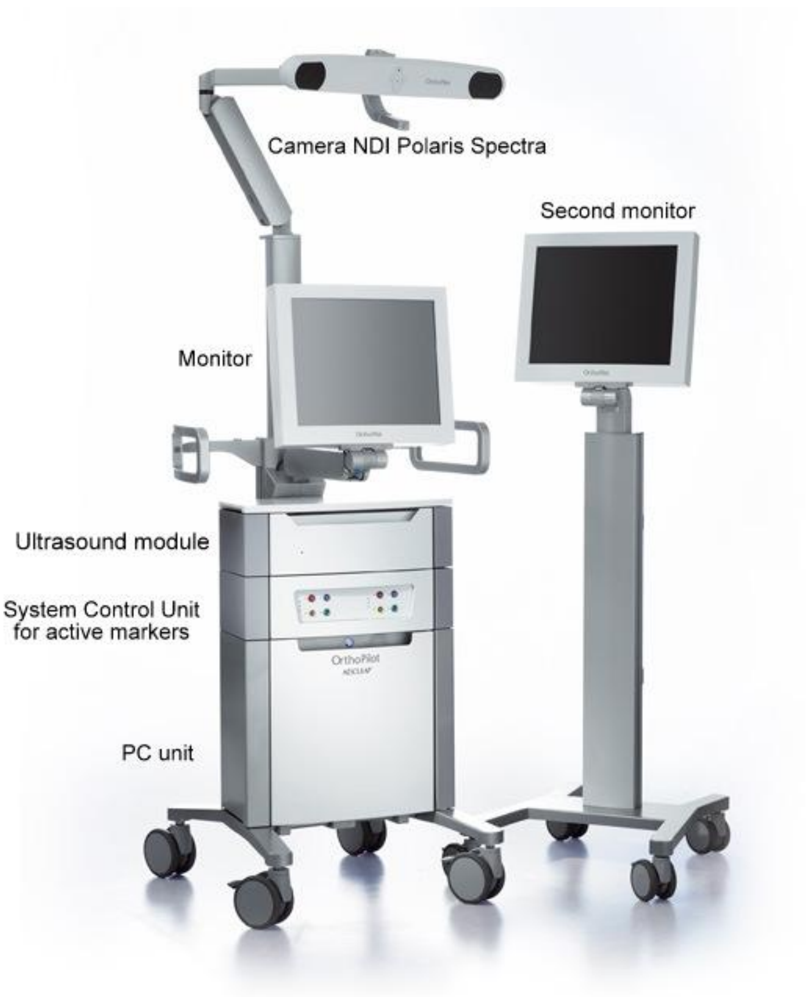

Camera NDI Polaris Spectra
Second monitor
Monitor
Ultrasound module
System Control Unit for active markers
PC unit

Figure 2: OrthoPilot Next Generation FS 101navigation system

2.1.1 Stereo camera

Markers can be precisely detected in distance up to 240 cm far from camera. [5] Camera[3] is the most important part of the whole system and its quality determines the accuracy of acquired position data and therefore the precision of the whole surgical intervention. Since very beginning OrthoPilot systems have used cameras from American company Northern Digital Inc. (NDI GmbH, Radolfzell, Germany) and the newest third version, OrthoPilot *Next Generation FS 101,* uses camera model NDI Polaris Spectra.

The most common structure of these video-optical systems used in navigated surgery (NDI cameras included) are two infrared (IR)-CCD cameras although there are also systems using light in visible spectrum. Based on the precisely defined distance between cameras and

[3] Commonly an equivalent term " localizer" is being used.

angle in between their axes using simple trigonometric rules, cameras can determine spatial position of the infrared-emitting or retro-reflective markers in the camera’s coordinate system (see Figure 3). To let the position sensor determine position and orientation of particular object, at least 3 markers are necessary as the plane is always given at least by three different points. These markers are affixed on special instruments, so called rigid bodies – RB. There are two types of markers: active (power supplied) and passive as will be described further in the next chapter.

In case of active markers, the power supply is procured by System Control Unit (SCU) which acts as a mediator between camera, markers and PC unit. Camera is connected to the computer via 14 pin Lemo *FGA.1B.314.CLL* (Lemo Elektronik GmbH, München, Germany) connector which provides 24V power supply and RS-422 (see Chapter 3.1.3) communication interface. Camera is currently coupled with PC unit over Host USB Converter in case of passive markers or SCU in case of active markers. These devices convert serial RS-422 communication to USB.

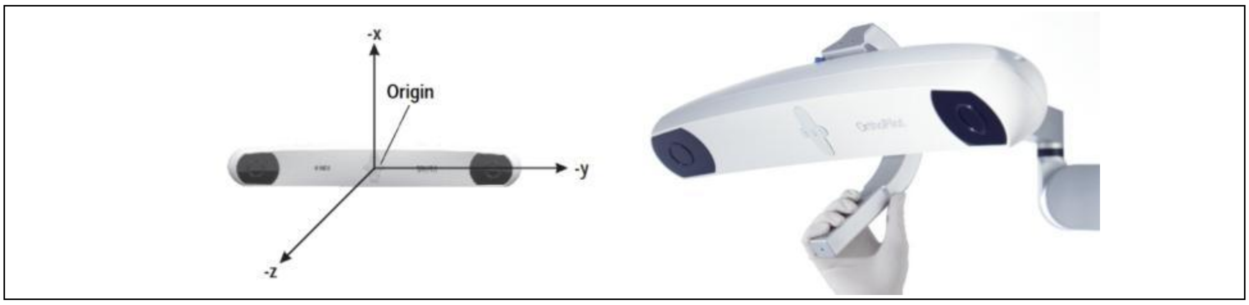


Figure 3: Stereo camera NDI Polaris Spectra; coordinate system (left)

Figure 4: System Control Unit (left), Host USB converter (right)

According to the manufacturer´s specification, *Polaris Spectra* detects a single marker with accuracy 0.25 mm RMS (Root Mean Square) in defined pyramid volume and 0.30 mm in extended pyramid volume as can be seen on Figure 5. [4] M

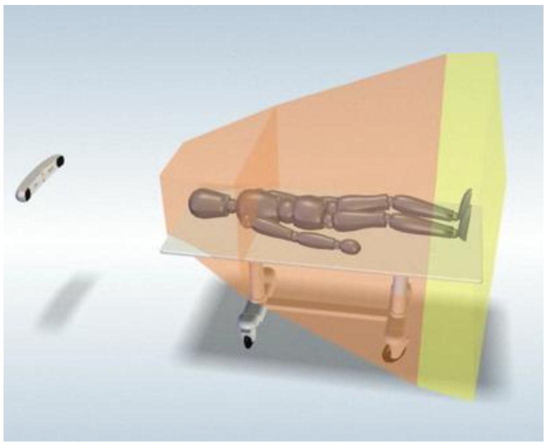


Figure 5: NDI Polaris Spectra Tracking Volume (yellow part = extended pyramid)

2.1.2 Rigid Body

Rigid bodies are instruments which consist of body and markers mounted on it and as already mentioned, markers can be either passive with small plastic spheres coated with special reflective layer on surface or active, with power supplied diodes emitting IR light. Overall pros and cons are [4]:

Passive

- ✓ Passive tools use spherical, retro-reflective markers that reflect infrared light emitted by the illuminators on the position sensor
- ✓ Tools are wireless
- ✓ Offers simple tool construction
- ✗ Requires each tool have a unique marker geometry resulting in larger tool sizes
- ✗ Passive spheres need to be replaced after single use
- ✗ Difficult to clean passive spheres intra-operatively

Active

- ✓ Active tools incorporate infrared-emitting markers that are activated by an electrical current
- ✓ Tools can be small since the same geometry can be used multiple times
- ✓ Markers can be cleaned easily
- ✓ Can incorporate switches and visible LEDs
- ✓ Tool Definition File can be programmed into tool
- ✗ More complex to build than passive tools
- ✗ Markers have limited life and are not replaceable
- ✗ Tools are wired to the system

Body itself was developed by Aesculap, is made of steel with high accuracy and every single piece is carefully checked because it is necessary that the coordinates of markers always point exactly to the tip of the RB otherwise the total inaccuracy would be increased. Based on the kind of intervention, up to 3 RB are used at the same time. To differentiate these, every rigid body has its own specific shape and color mark. It has become a standard that blue marked (●) RB is mounted to tibia, red (●) belongs to femur and yellow (●) is mounted on a pointer. The pointer is an instrument which is not fixed to any bone or structure, but is used in surgeon´s hand to palpate significant points on bones with the tip of the pointer. These points are important to determine geometry and sizes of the bones. In the case that the patient´s fat layer is too thick and the significant points are impossible to palpate, ultrasound is used instead.

The camera has to determine the coordinates of a tip of the instrument (approximately denoted in the figure below by green point) based on positions of the markers. To do so, camera has to know the exact geometry of the RB and distances between markers and tip of the RB. All these information are stored in “tool definition file” (.rom extension) and loaded into camera’s internal memory

All markers of particular RB must be visible for camera otherwise the camera will send an error message. The visibility status of every RB is denoted in the main application window in every step in top center part by colorful circles (green = visible, yellow = on the border of visibility, red = not visible)

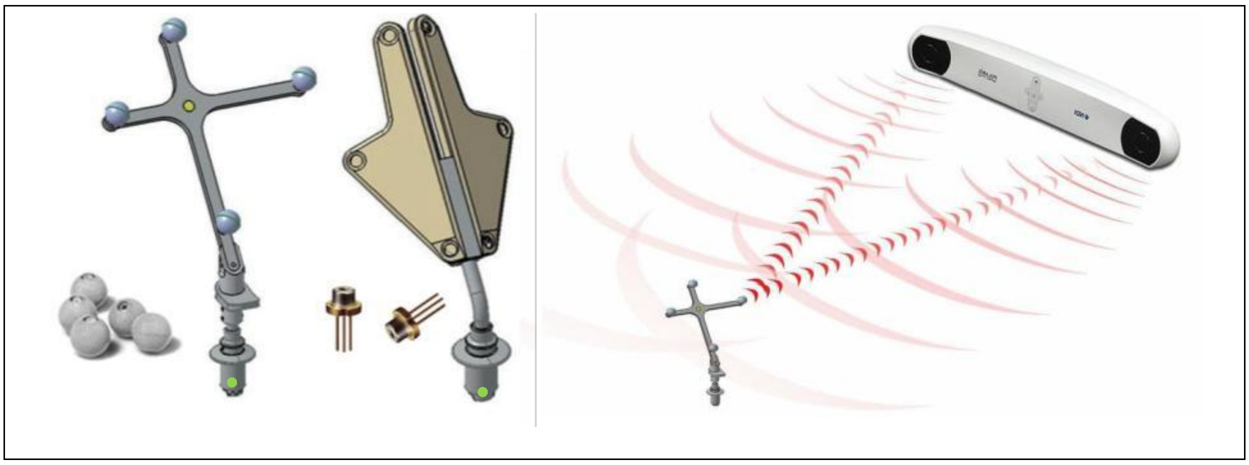


Figure 6: Passive RB with reflective spheres (left) and active RB with IR diodes (right)

2.1.3 Ultrasound module

If wanted OrthoPilot systems may be equipped with an ultrasound module, which allows surgeon to obtain significant bone points in a minimally-invasive way. This is used mostly when operating obese patients, where the fat layer does not allow palpating the bone structure directly on the skin surface or slightly deeper. In addition the usage of the ultrasound probe also decreases the risk of infection and generally shortens the healing process. In order to be able to define coordinates of the ROI (Region of Interest) using ultrasound, the coordinate system of the ultrasound probe must be synchronized with the coordinates of the whole system. To do so, there is a yellow RB mounted directly on the probe.

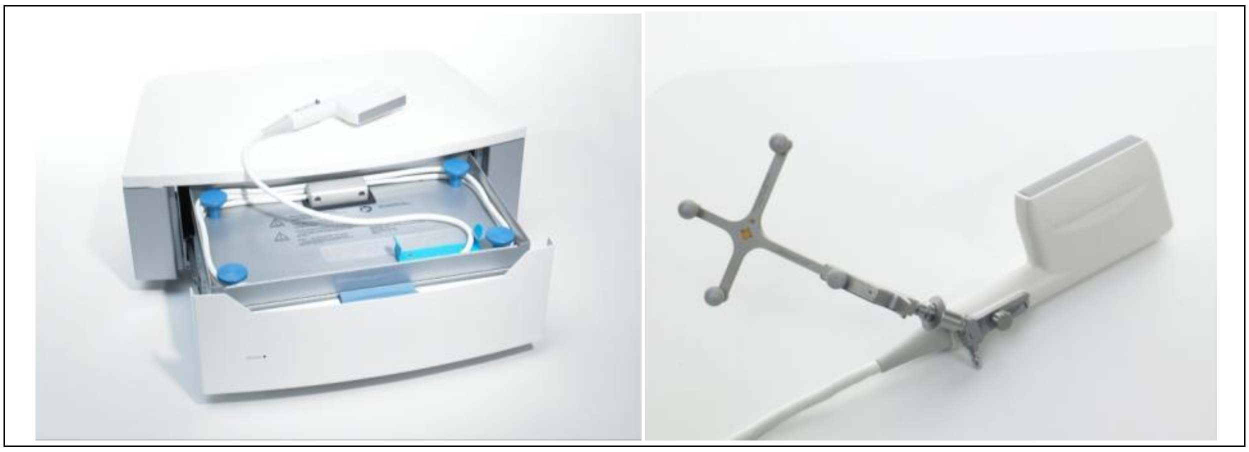


Figure 7: Ultrasound module and the ultrasound probe with yellow RB mounted

A lot effort is being put on development and exploring further possibilities of using ultrasound nowadays and especially how to reliably recognize and increase the recognition accuracy of the interesting spots in the ultrasound image.

2.1.4 Footswitch

To control the software application running on the screen, surgeon has two different possibilities: screen itself, which is equipped with a touch panel or footswitch. The whole application is controlled by two (three) buttons only, differentiating right, left, (both or middle) button and short and long pressing. At every step in the application the same action can process different event or the buttons can be disabled at the certain moment until required task (e.g. rotating tibia) is completed. This is always described by a button symbol on the screen. Mostly the button events are the same in the whole procedure e.g. right button pressed for a short time will collect the data, holding the right button for predefined minimal interval (approx. 1,5s) will proceed forward onto the next step/screen in the application. Shortly pressed on left button will perform step back in the application while holding the button will discard the collected data. Pressing both buttons simultaneously will save a screenshot (see example in Figure 8).

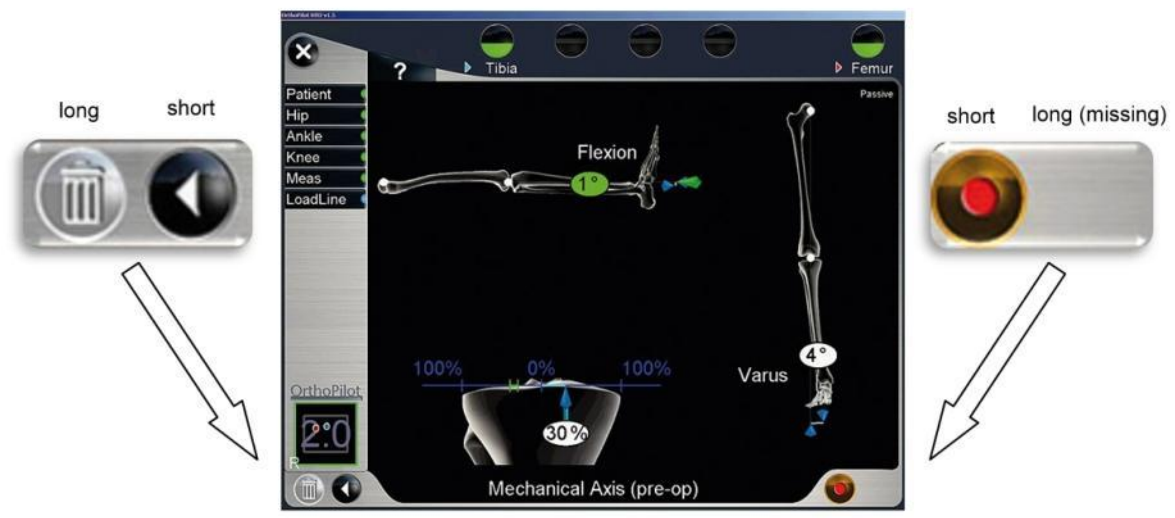


Figure 8: OrthoPilot HTO v 1.5 - different button events. Notice that right long action (moving step forward) as well as left long action (erasing collected data) is not available as the data were not collected yet. It is done by pressing right button.

OrthoPilot *Next Generation* system uses footswitch steute *MKF-2 MED G26* (steute Schaltgeräte GmbH & Co. KG, Löhne, Germany) which is, compared to previous versions, water-resistant that allows this to be washed and cleaned easily. [5] Footswitch is connected to the PC unit via USB cable and communicates with the application using predefined commands. These are in regular intervals send by footswitch and loaded to the application buffer, which is regularly read and updates the information on the screen. [6]

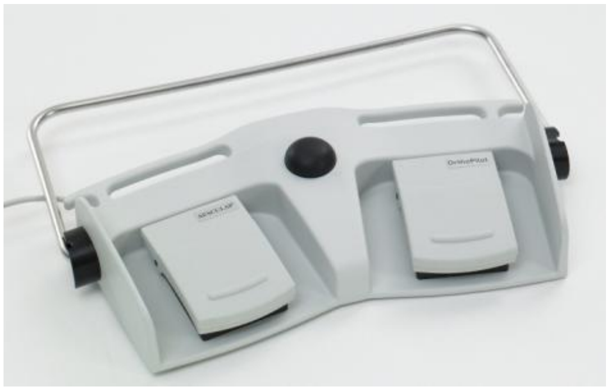


Figure 9: OrthoPilot Footswitch

2.1.5 PC unit and overall body

OrthoPilot system is equipped with embedded computer unit made by Penta company (PENTA GmbH, Deggendorf, Germany), which was developed especially for usage with OrthoPilot *Next Generation* systems. PC unit contains UPS[4] spare energy source which, in the case of electricity lost can support system with energy for approx. 5 min. This gives surgeon enough time to store and backup collected data. [5] Compared to previous version *FS 100, FS 101 Next Generation* uses touch screen monitor instead of the standard keyboard and mouse which allows easier manipulation and maintenance.

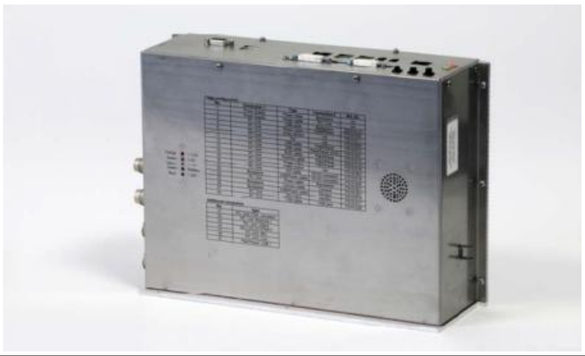


Figure 10: PENTA embedded PC unit

Although it is not totally easy, OrthoPilot system is portable and for these purposes can be collapsed in much smaller sizes of about 80cm x 140 cm x 70 cm. Total weight is 65 kg. PC unit is equipped with second video output (standard DVI[5] female connector), which allows user to connect to second monitor or beamer either during an operation or outside the OR for educational or another purposes. After the operation is over, reports and data must be stored or uploaded into Clinical Information System (CIS). This must be done manually with USB-stick or burned CD, which nowadays seems to be little inconvenient and out-of-date way.

2.1.6 Software

Last but not least software is a very important part of the whole system. Generally it can be divided in two parts. First of them contains classes for initialization and communication with peripheral devices and cares about the technical background while the second one can be called as ‘medical’ one. This includes all the values and calculations for defining proper angles, distances and evaluating them to tell the surgeon where to cut the bone and place an implant. This part contains many pictures and animations to inform surgeon

[4] Uninterruptible Power Supply (Source)

[5] Digital Visual Interface

about what he does respectively about what OrthoPilot wants him/her to do. Application is coded in C++ programming language. Until now there are many different OrthoPilot applications available according to different interventions e.g. ACL, TKA, THA.

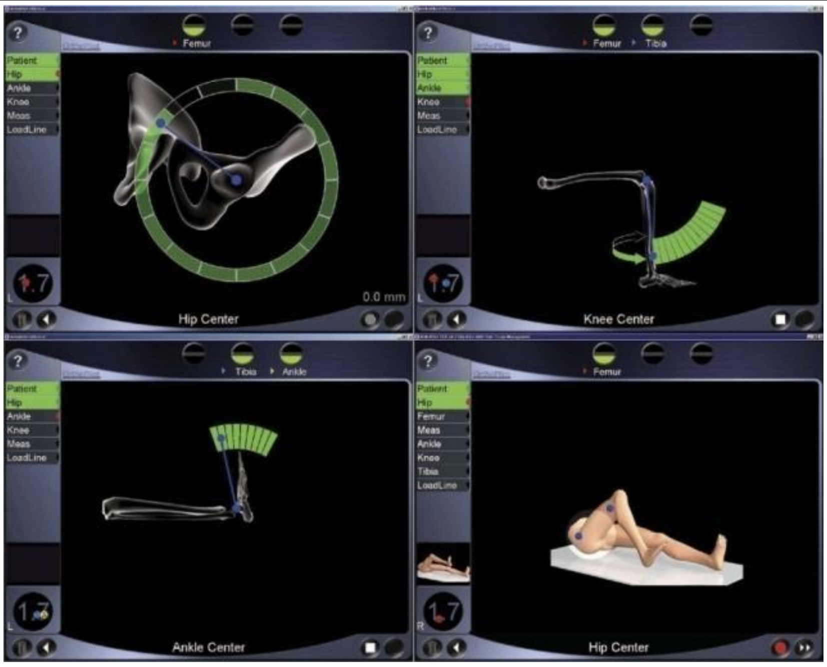


Figure 11: Printscreen samples - OrthoPilot HTO v 1.4

2.2 Mathematical background

Although it is not absolutely necessary for understanding following text, when describing navigation system, it is suitable to describe basic principles and mathematical theory in background. Related to our system, most important are geometrical operations which allow us to transform one coordinate system into another, respectively to describe relationship between these two in order to define position of a point in a new coordinate system. This happens by means of two basic operations – translation and rotation.

Note: In whole following chapter, by term ‘coordinate system’, 3D orthogonal coordinate system is meant.

2.2.1 Translation vector

Translation vector describes the movement of a point/object/coordinate system in 3D space. Every point in this space is defined by its position vector $\rightarrow p$ consisting of three elements p_x, p_y, p_z. In order to translate this point translation vector $\rightarrow t$ (t_x, t_y, t_z) is added. To transpose whole object or coordinate system, this is applied to every point in this object/system. Mathematically translation is described as follows: [7]

$$\vec{p}' = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Equation 1: Translation [7]

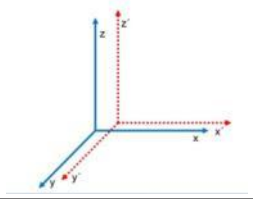


Figure 12: Translation

2.2.2 Rotation matrix

To rotate point by particular angle around any axe (or axes) we must multiply its position vector →*p* with rotation matrix *R*. Rotation around each of three axes is described by following matrixes $R_{x,}$ R_y and R_z . [7]

$$\overrightarrow{p}\,' = R * \overrightarrow{p}$$

Equation 2: Rotation [7]

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha_x) & sin(\alpha_x) \\ 0 & -sin(\alpha_x) & cos(\alpha_x) \end{bmatrix}$$

Equation 3: Rotation X-axe [7]

$$R_y = \begin{bmatrix} cos(\alpha_y) & 0 & sin(\alpha_y) \\ 0 & 1 & 0 \\ -sin(\alpha_y\ ) & 0 & cos(\alpha_y) \end{bmatrix}$$

Equation 4: Rotation Y-axe [7]

$$R_z = \begin{bmatrix} cos(\alpha_z) & -sin(\alpha_z) & 0 \\ sin(\alpha_z) & cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 5: Rotation Z-axe [7]

By combination of these three matrixes we get complete rotation matrix *R*.

$$R = \begin{bmatrix} cos(\alpha_y) * cos(\alpha_z) & -cos(\alpha_y) * sin(\alpha_z) & sin(\alpha_y) \\ sin(\alpha_x\ ) * sin(\alpha_y) * cos(\alpha_z) + cos(\alpha_x) * sin(\alpha_z) & -sin(\alpha_x) * sin(\alpha_y) * sin(\alpha_z) + cos(\alpha_x) * cos(\alpha_z) & -sin(\alpha_x) * cos(\alpha_y) \\ -cos(\alpha_x) * sin(\alpha_y) * cos(\alpha_z) + sin(\alpha_x) * sin(\alpha_z) & cos(\alpha_x) * sin(\alpha_y) * sin(\alpha_z) + sin(\alpha_x) * cos(\alpha_z) & cos(\alpha_x) * cos(\alpha_y) \end{bmatrix}$$

Equation 6: Rotation matrix [7]

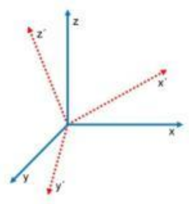


Figure 13: Rotation

2.2.3 Transformation matrix

Transformation matrix is the combination of both previous: translation and rotation, and is always sufficient to describe relationship between coordinate system A and coordinate system B (A→B). Transformation matrix looks as follows:

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 7: Transformation matrix [7]

Indexed letters r_{11}-r_{33} denote elements of rotation matrix (3x3) and in the last column translation vector $\to t$ (t_x, t_y, t_z) (1x3) is added. Last row denotes vector of distortions and scaling and is filled by zeros and one when no distortion or scaling is applied, that is both A and B are orthogonal coordinate systems. [7]

By means of transformation matrixes the position of a marker (respectively tool tip) is passed from the camera (camera coordinate system) to the application running on PC which represents another coordinate system.

2.3 Future vision

As was already mentioned in the introduction part, current work and research of Aesculap developers focuses on development of new generation OrthoPilot system when one of the most significant features should be implementing wireless technologies to substitute current cable connections to make system more adaptable, portable, easy-to-handle etc. Overall concept of the future system looks as follows:

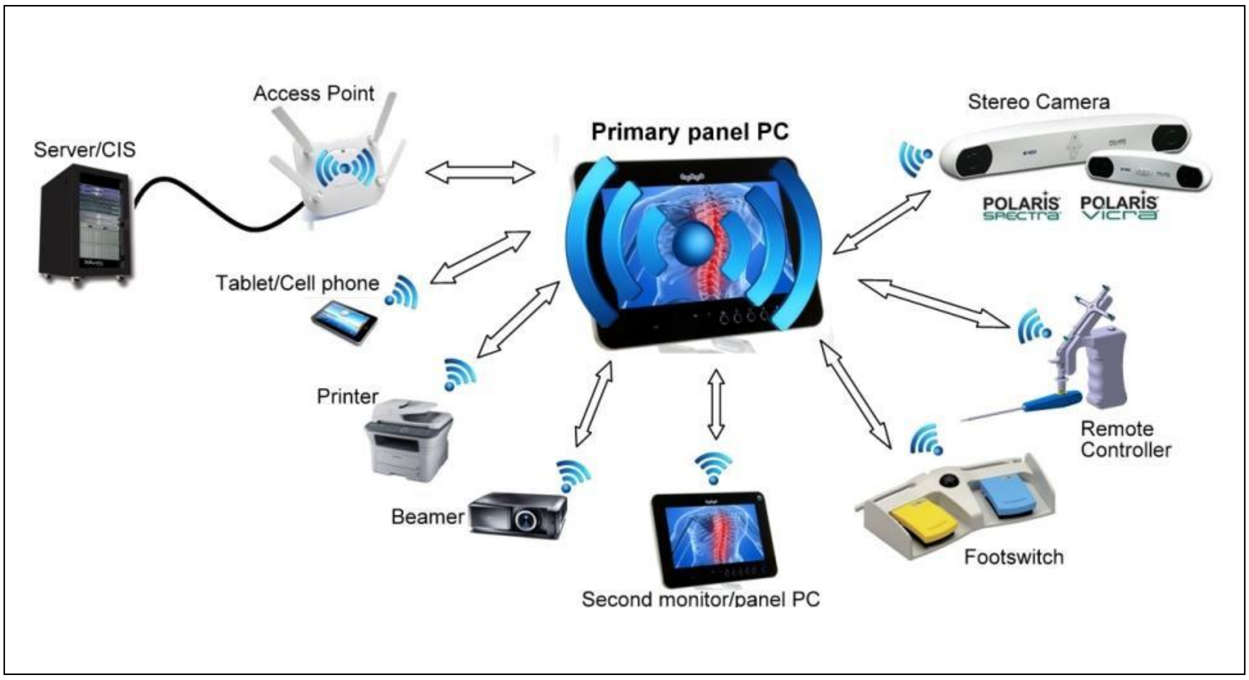


Figure 14: OrthoPilot future wireless concept

2.3.1 Panel PC

Previously used PC unit is substituted by panel PC equipped with touch screen. Combining computer with monitor into one device will saves space, decrease weight and energy consumption. On this panel PC the OrthoPilot software runs and the screen is wirelessly cloned (streamed) onto second monitor/panel. Both displays are detachable and can be mounted on any position as required by the surgeon, limited only by electricity plug.

2.3.2 Footswitch, Remote Controller

Both devices allow user to remotely control application running on the main panel PC by pressing backward and/or forward button. The whole remote controller with its buttons is mounted directly on instrument (pointer), which in some moments can be more convenient than searching for the footswitch on the floor. Remote controller also displays visibility of the RB by means of LEDs or small built-in-display. Because instruments are used directly in operation field, the surgeon is informed immediately without any need to watch any of screens. Further vibration and sound alerts are added to inform surgeon about processed action.

Both footswitch and remote controller communicate with the panel PC wirelessly and the power supply is procured by accumulators with appropriate dimensions and sufficient capacity.

2.3.3 Camera

Camera is no more limited by the distance from the main PC, because wireless interface is used instead of cable and therefore camera and panel PC can be mounted separately. This makes manipulation with the camera trolley easier and offers freedom in placing components in the OR. Camera position depends only on power supply (18-32 V), which can be possibly integrated into trolley’s base. As the second effect, the weight of the accumulator placed in bottom makes the trolley stable.

2.3.4 Connection to CIS

OR is equipped with an access point (AP) providing secured WLAN (Wireless Local Area Network). Usage of two different frequency bands (2,4 and 5 GHz) can eventually separate the environment between intra-OR and extra-OR to increase security (sensitive data protection), reliability, avoiding hacking attacks etc. The second mentioned would be used for connecting OrthoPilot to CIS directly, substituting USB-sticks or CD for transferring post-operation reports.

2.3.5 Other Peripherals

Wireless network or another direct wireless connection serves as an interface also for other devices such as tablet PC, beamer or printer.

3 Interfaces

Components of the OrthoPilot navigation system are bounded (are planned to be bounded) using different communication technologies. Following sub-chapters describe each of these currently used (serial) and intended-to-use (wireless) standards.

3.1 Serial communication

Generally serial communication means transferring of digital data while the term "serial" tells us that single bits of a byte are transferred one by one, that means at each moment only one bit is sent/received. Although this is slower than parallel communication, which allows transferring an entire byte at once due to separate line for each bit, it can be used to transfer data over much longer distances e.g. IEEE 488 standard for parallel communication states that the distance between any two devices can be no longer than 2m. In comparison serial communication can extend as much as 1200m. [8] There are two basic types of serial communications, synchronous and asynchronous. With synchronous communications, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. That is, each character that is sent is either actual data or an idle character. Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communications to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters. In most of applications and devices, including ours, asynchronous form is used. [9]

3.1.1 UART: Universal Asynchronous Receiver Transmitter

UART is a basic element (also can be considered to be a communication standard), a piece of computer hardware which translates data between parallel and serial form in order to make them ready for the next usage – commonly with RS-232, RS-422 or RS-485 standards. The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signaling levels. External signals may be of many different forms (e.g. voltage levels of RS-232 standard). [10] Because asynchronous communication lacks any time synchronization between sender and receiver, the recognition of the beginning and the end of the transmission must happen in different way.

- ***Communication by bits***

Serial devices don´t use bits only to convey data (data bits), but also to establish and control communication by means of special purpose bits: Start bit and Stop bit. At the beginning of every communication start bit must be sent in order to inform receiver that the sender is about to start sending data bits. Start bit is always represented by logical “0”, which causes rising edge on the line and a signal for receiver. Start bit is followed by selectable number of data bits (5, 6, 7, 8) while the selected number limits the data size which can be sent. With 8 data bits, all ASCII[6] values can be sent (0-255), while 7 data bits don´t allow us to send ASCII value greater than 127 etc. After sending the whole byte, the frame must be terminated by stop bit. This is represented by logical “1” and returns line into the previous state as it was before transmission. Number of stop bits is usually set to 1 or 2 which also determines the minimal pause between two transmissions [9]

✗ *Bi-directional communication*

Bi-directional (full-duplex) feature means that sending and receiving of data can happen simultaneously while there is separate line for each operation. In fact three lines (transcieve - TxD, receive - RxD, common signal ground) would be enough, but additionally there are some more added to control the communication. [9]

✗ *Parity bit*

Beside the start and stop bits which cares about synchronization, an additional parity bit can be transmitted along with data to inform about potential error in transmission. Parity can be chosen as even, odd, or can be disabled at all. When even or odd parity is used, before the data is sent total number of “1” data bits is counted and according to the result (odd or even) the parity bit is set to “0” or “1” in order to increase the number of “1” bits or leave as it is, because the final number already matches selected parity. On the side of receiver, this must have the same settings, the sum of “1” bits (from 8 data bits and parity bit) is checked and compared to selected parity. If there was any single bit error, the parity will not match the selected parity which is the signal for the user that there was an error in transmission. Parity control checking is very rudimental. Evidence may be for example: [9]

- ✓ If even number of bits is in error, the parity bit will not reflect any error at all
- ✓ Not matching the parity inform us about an error, but not about which bit (even number of bits) is faulty

[6] American Standard Code for Information Interchange - as computers can only understand numbers, an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort (delete, escape). Standard characters have their decimal value from 0 to 127, extended ASCII table codes less common chars (values 128-255). ASCII Table can be found in Appendix A.

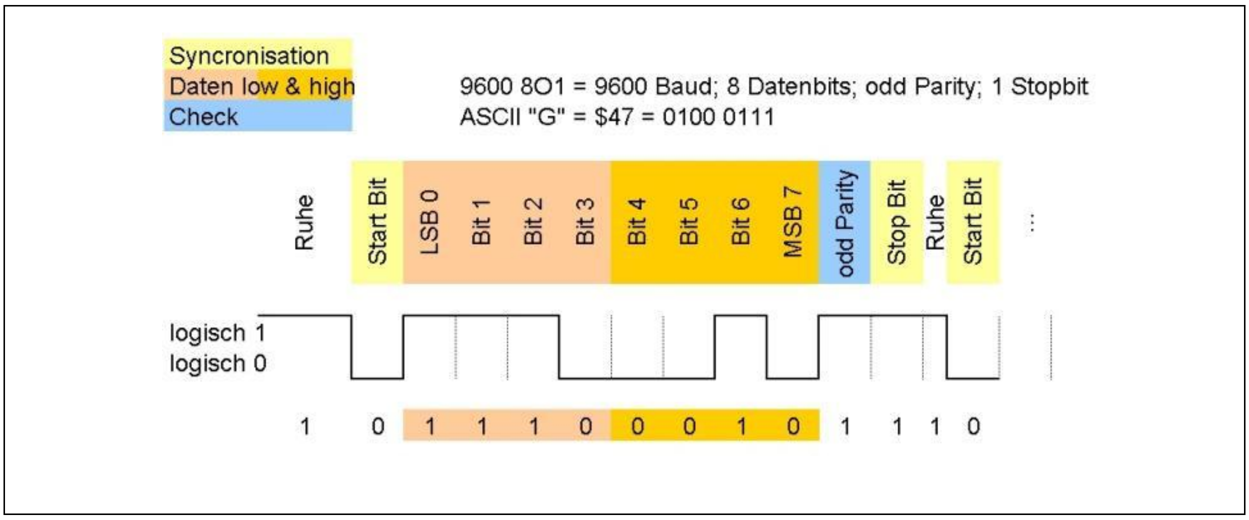


Figure 15: UART: Frame structure. Example of line state when "G" character is sent [10]

3.1.2 RS-232

RS-232 (EIA-232) stands for Recommend Standard number 232 which specifies unique electrical as well as mechanical properties (e.g. voltage levels, wiring, connector shape and dimensions, transfer speeds) of the serial interface. Since 1962 most of the computers have been equipped with male D-type 9-pin (DB9) connector as defined by this standard. RS-232 has become very popular way for connecting computers (called as DTE[7]) to peripheral devices e.g. modems, sensors, machines (called as DCE[8]) but since last years it stays in the shadow of another popular standard – USB, which overtook its priority, especially thanks to much higher transfer speeds. Very often RS-232 interface is called also communication port or COM port.

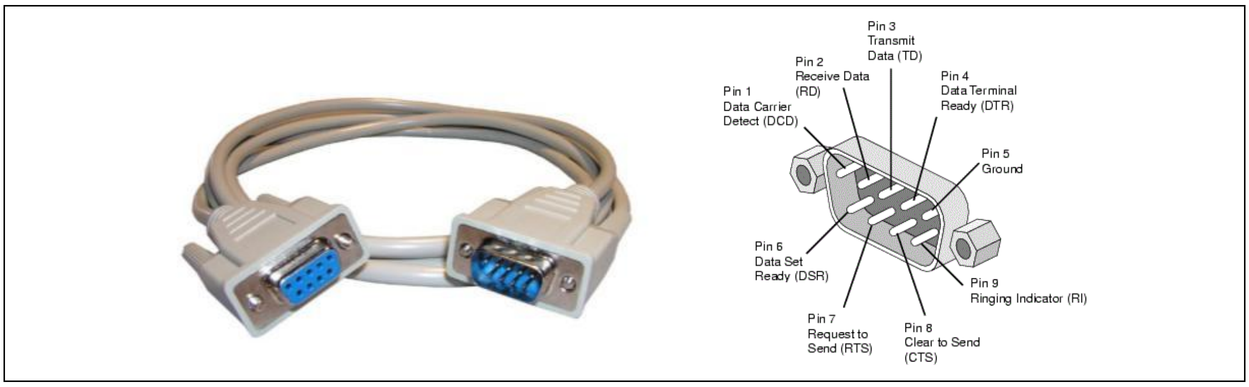


Figure 16: RS-232, male and female DB9 serial connector [11] Pin description [12]

[7] Data Terminal Equipment

[8] Data Circuit-terminating Equipment

As already mentioned 3 lines are sufficient for bi-directional communication but beside TxD and RxD line, lines for controlling signals are added. Here comes an overview of all lines/pins in DB9 connector and their purpose: [10]

- **GND**: common ground level
- **TxD**: Transmit Data. This line carries data from the computer to the serial device
- **RxD**: Receive Data. This line carries data from the serial device to the computer
- **DTR**: Data Terminal Ready is used by the computer to signal that it is ready to communicate with the serial device. In other words, DTR indicates that the computer is ON.
- **DSR**: Data Set Ready similarly to DTR, is an indication from the Dataset that it is ON.
- **RTS**: Ready To Send is used to request clearance to send data to a serial device. In other words computer asks the device "Are you ready to receive my data?"
- **CTS**: Clear To Send is used by the serial device to acknowledge the computer's RTS Signal/question. In most situations, RTS and CTS are constantly on throughout the communication session.
- **DCD**: Data Carrier Detect. Is used very rarely.
- **RI**: Ring Indicator. Is used very rarely.

Regarding the logic levels, RS-232 standard is inverted compared to UART. For data transmission lines the "negative logic" is used. To send /receive logical "0" the line voltage must be between positive +3V to +15. Negative voltage -3V to -15V stands for logical "1". Voltage between -3V to +3V is not valid for RS-232 standard. On the contrary control signals (CTS, RTS, DSR, DTR) are inverted in terms of logic ("positive logic"). This means that active state on these lines is represented by positive voltage while inactive state is denoted by negative. [10]

CTS and RTS are also called as flow control or hand shake. This is commonly one of the settings in RS232 serial communication which decides whether the flow control will be used or not.

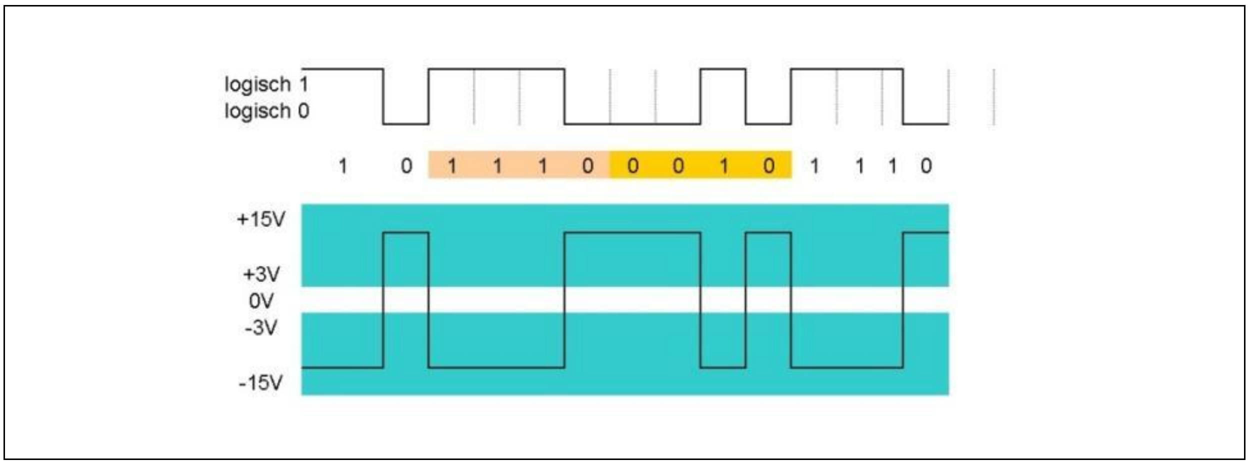


Figure 17: UART (above) and RS-232 (bottom) logic levels [10]

RS-232 supports speeds up to 1,5 Mbit/s (MBd/s[9]) but this number decreases rapidly with growing distance. However it is possible to connect almost 1 km distant devices when special low-impedance cable is used. In this case the communication speed reaches only approx. 2 kbit/s. [13]

3.1.3 RS-422

RS-422 is a technical standard that specifies electrical characteristics of serial connection. In compare to RS-232, this uses differential pair of lines for each signal which allows interconnecting two devices in much higher distances (up to 1200 m) using higher transfer speed (up to 10 Mbit/s[10]). RS-422 may bind two devices directly or is also often used for extending the range of RS-232 communications. [12] As the standard defines only electrical characteristics, there is no fixed norm defining connector or pin numbers. Therefore RS-422 often uses the same 9-pin connector as is used in RS-232 communication.

The cable consist of altogether 4 pairs (±TxD, ±RxD, ±RTS, ±CTS) while for each pair, two wires are established. First line, denoted as “A” or “-” carries inverted signal and the second “B” or “+” carries non-inverted signal. On the side of receiver both signals are subtracted (A-B) which means that the final voltage will be 2* original signal level. This technique of differential pairs helps rapidly to protect signal against noise which influences both lines simultaneously with same additive signal. While the carrying signal is added, the noise signal is subtracted as can be seen on the figure below.

[9] Bd stands for 1 Baud. Unit defines number of pulses per second. In digital (binary) systems 1 Bd = 1 bit.

[10] This number is defined for distance 12 m

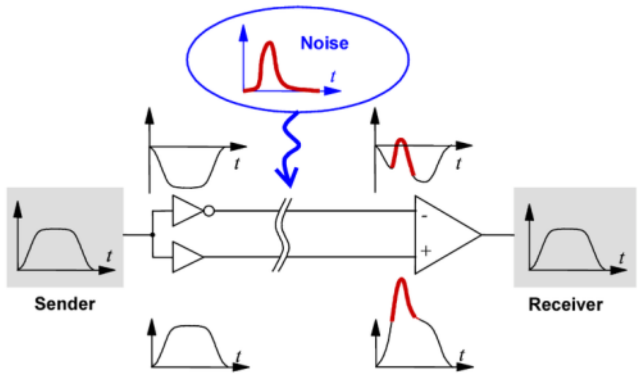


Figure 18: Differential signaling, noise protection [15]

RS-422 operates within ±200 mV to ±10 V while using negative logic. Therefore if the difference of signal lines (A-B) is greater than +200 mV, receiver will recognize logical “0” and if the difference is lower than -200 mV, logical “1” is read.

It is recommended that the differential lines are terminated with a terminating resistor (standard values approx. 100 Ω) on the side of receiver to avoid reflecting of RF (radio-frequency) back which could be source of interference. [15]

	RS232	**RS422**
Differential	no	yes
Receiver input sensitivity	±3 V	±200 mV
Receiver input range	±15 V	±10 V
Max distance (acc. standard)	15m	1200m
Max speed at 12 m	20 kbs	10 Mbs
Max speed at 1200 m	1 kbs	100 kbs
Network topology	point-to-point	multidrop

Table 1: RS-232, RS-422 comparison [16]

3.1.4 Logic levels

When the receiver receives certain voltage value and recognizes the “0” or “1” state according to values defined by standards (RS-232, RS.422), this information usually has to be passed further into digital circuit. Logic levels are predefined states (finite number of states) used within digital circuit (voltage ↔logic) which allow two electronic parts to understand each other. In order to do so, it is necessary that these two use the same logic. The range of voltage levels that represents each logical state depends on the logic family being used. Most common logic families are TTL (Transistor-Transistor-Logic) and CMOS (Complementary metal–oxide–semiconductor) which is used in two forms: 5V and 3,3V. The information about which logic family does the particular IC (Integrated Circuit) use is always specified in the components documentation/datasheet provided by manufacturer. Following figure describes the TTL and CMOS voltage levels for high and low state.

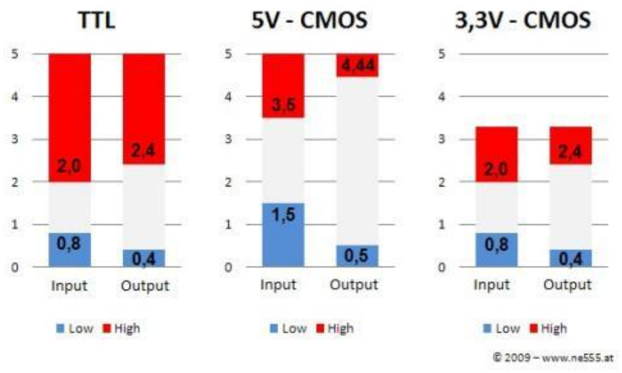


Figure 19: TTL and CMOS logic family voltage levels [17]

Besides defining the voltage levels it is necessary to decide whether high state of line represents logical "0" or "1" which may be sometimes little confusing. As it was already mentioned when describing serial interfaces, there are two different logics:

- Positive/active high (high = 1, low = 0)
- Negative/active low (high = 0, low = 1)

3.1.5 USB: Universal Serial Bus

USB is another interface, which actually does not match definition for serial communication as it does not send data bit by bit but in packets of particular size (8-256 Bytes). Since establishment in 1995 it has become very popular thanks to its universality and much higher transfer speeds compared to RS-232 or other standards. All communication is based on one differential pair (D+, D- line) which is twisted to decrease influence of electromagnetic noise signals. Besides these, USB standard describes 2 other lines for supplying external devices with power (+5V, GND). A device is allowed to draw current up to 500 mA from a port in USB 2.0, for newest standard USB 3.0 up to 900mA. If this is not sufficient, device can combine USB power supply with external one or to depend fully on external supplement. Compared to other standards, USB interface consist of one Master and more possible Slave devices. Connected devices (slaves) cannot send any data unless they are requested by master. The transmission happens via so called "frames" which last exactly 1 ms. Master can communicate with more slaves at the same moment by putting packets for all these slaves together into one frame. In the same frame both slow (low-speed) and fast (full-speed) can coexist. Communication is managed and controlled exclusively by master, which means that slaves must synchronize themselves according to data flow.

Data rates differ according to the version. Until now version USB 1.0, 2.0 and 3.0 was introduced. USB 1.0 differentiate two different data rates *low-speed* (1,5 Mbit/s) and *full-speed* (12 Mbit/s). The main difference between those is that in *low-speed* form it takes 8 times longer to transmit bit than for *full-speed*. USB 2.0, which was introduced in 2008, supports *high-speed* data rate (480 Mbit/s) and is fully backward compatible with USB 1.0 version. USB 3.0 is getting very popular nowadays as it supports data rates up to 5Gbit/s. Compared to previous versions, 4 additional lines were added. Again 3.0 version is fully backward compatible. [18]

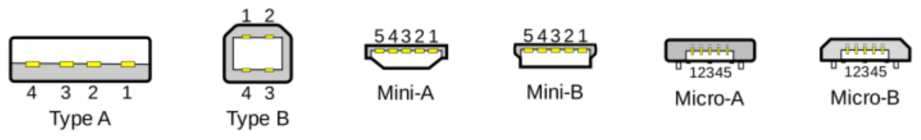


Figure 20: USB Standard, Mini, Micro plugs [18]

3.2 Wireless communication

Since last thirty years, wireless technologies have experienced a phenomenal growth and their fast development (due to new possibilities of integrated circuits elements) and growing popularity extended their usage from small individually developed applications into worldwide accepted standards and implementation in plenty of different fields (e.g. industrial, scientific, private, medical) and common applications. Wireless technologies are currently one of the fastest growing areas of the electronics industry. However it was not only fast development which has contributed to their popularity and spreading. To do so, basic legislative changes in licensing and allowance of radio frequency bands needed to be set in place. It was common that provider needed to have a license to use particular frequency band in order to prevent any interference which would have might be caused by other users. Important change was made in 1985 when the Federal Communications Commission (FCC) in the USA opened up some small frequency bands of the spectrum for license-free applications. The bands that were used were the 900 MHz, 2.4 GHz, and 5.8 GHz and were called as Industrial Scientific Medical (ISM) bands. These portions of the radio spectrum were reserved internationally for the use of RF energy for industrial, scientific and medical purposes other than communications. An example of applications in these bands includes microwave ovens, medical diathermy machines, Bluetooth devices, cordless telephones etc. Because these devices emit lot energy and could possibly interfere with other devices which have to work reliably, they were assigned particular frequency bands. On the other hand the norm says that if the communication device wants to operate in these (ISM) bands, it must tolerate any possible interference caused by another operating device and that the user has no regulatory protection if the communication is disrupted. Although the original intention (noncommunication equipment), in recent years ISM bands have been used mostly for fast developing short-range, low power communications systems .[18][19]

Following paragraphs describe every particular technology (which possibly might be used with OrthoPilot system) in detail.

3.2.1 WLAN / IEEE 802.11

IEEE[11] 802.11 is a working group as well as collection of standards defining wireless networks which are based on the previous standard 802.3 (also known as Ethernet or LAN) and because 802.11 may be simply called wireless Ethernet similar term WLAN (Wireless Local Area Network) was introduced and is primarily used. WLAN is often substituted by term Wi-Fi which is nothing but trade mark for products based on IEEE 802.11x standards. As the 802.11 is continuation of 802.3, these standards are fully compatible and also in praxis WLAN is often used as an extension/connection to/between LAN(s). Letter "x" mentioned above stands for different modifications/versions of this standard, which were developed mainly to satisfy consumers calling for higher transfer speeds and security. First version was introduced in 1997 (802.11) and supported speeds up to 2 MBit/s. Because operating range (2,4 GHz) is allocated in unlicensed ISM band, standard became very popular and started to spread widely.

✘ ***Transfer Speed (Throughput)***

As the speed is depending on many factors (environment, number of network participants and their distance) you can almost never reach the values specified by manufacturer and the difference can be surprisingly big. Therefore it is fair (but is not very common) always to provide both brutto and netto values. Standard 802.11n offers such higher data rates through doubled channel bandwidth (40 MHz) capable of throughput 65 MBit/s and MIMO (Multiple Input Multiple Output) technology, which uses up to four antennas/streams at the same time, each capable of 150 MBit/s (in total 4*150 = 600 MBit/s). Besides these two features also the simultaneous use of both 2,4 and 5 GHz bands contributes to higher data rates. You can see an overview of all versions and their transfer speeds in a table below.

[11] Institute of Electrical and Electronics Engineers: non-profit professional association headquartered in New York City that is dedicated to advancing technological innovation and excellence.

Standard	802.11	802.11b	802.11b+	802.11g	802.11a/h/j	802.11n
Frequency band	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	5 GHz	2,4 + 5 GHz
Compatible with	802.11b	802.11b+/g	802.11b/g	802.11b/b+	-	802.11b/g
Transfer speed (Brutto)	1 - 2 MBit/s	5,5 - 11 MBit/s	5,5 - 11 MBit/s	6 - 54 MBit/s	6 - 54 MBit/s	150 - 600 MBit/s
Transfer speed (Netto)	0,5 - 1 MBit/s	1 - 5 MBit/s	1 - 5 MBit/s	Up to 22 MBit/s	Up to 22 MBit/s	Up to 150 MBit/s
Indoor range (max)	40 m					
Outdoor range (max)	100 m				2 km	
Emitted power (max)	100 mW				200 mW - 1 W	

Table 2: IEEE 802.11standard overview [22]

✘ *Frequency Bands, Channels*

Regarding the frequency bands 2,4 GHz stands for range of frequencies between 2,4 and 2,4835 GHz as well as 5 GHz denotes 5,15-5,35 and 5,47-5,725 GHz. Both of these bands are allocated in unlicensed bands and therefore have to be resistant or protected against interference which occurs due to many RF-emitting devices operating on the same frequencies. One of the protective mechanisms is channel selection. Each band (2,4; 5 GHz) is divided into certain number of channels with defined bandwidth. Standards operating in 2,4 GHz frequency band use 13 (in Europe; in US 11, in Japan 14; see Appendix B) channels with bandwidth 22 MHz while each is capable of data rates 54 Mbit/s. Each channel overlaps an adjacent one but non-overlapping channels can be found (e.g. Channels 1, 6 and 11; 2 and 7 etc.). When choosing frequency channels for wireless stations in the vicinity of each other, it is recommended to choose frequency channels that are several channels apart from each other (e.g. Channels 1 and 6).

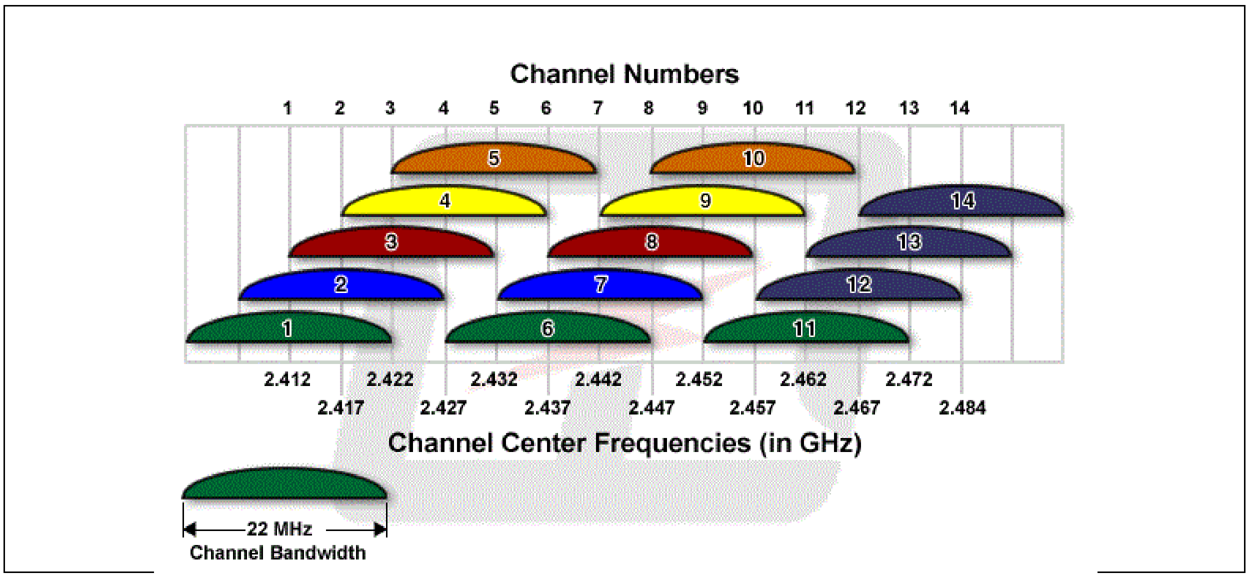


Figure 21: 2,4 GHz Channelization Scheme [23]

In comparison standard 802.11n operating in both 2,4 and 5 GHz band offers an option to double the bandwidth per channel to 40 MHz bandwidth, which results in slightly more than double data rates. The 5 GHz ISM band is divided up into sub-bands called U-NII bands (Unlicensed National Information Infrastructure) and are usually named U-NII-1, U-NII-2, U-NII-2e, and U-NII-3 where U-NII-3 is not freely available worldwide. [24] In Europe, special requirement for Dynamic Frequency Selection (DFS) and Transmit Power Control (TPC) capabilities are set as well as limitation for channels between 5,150-5,250 GHz to be used only indoors. In total there is 19 non-overlapping channels (20 MHz bandwidth) allowed in Europe (see Appendix C). [22]

✘ *Security*

Compare to LANs which are usually established within one building or complex of buildings, WLANs operate in an open space. For this reason it is much easier to break into or listen to running communications and therefore some safety features have been developed to secure data transmissions. Primary goal is to deny access into a network to any unauthorized user, which is done by encryption and authentication. Encryption technology scrambles messages which are send over the network so that they cannot be easily read by humans. All today´s Wi-Fi equipments already supports at least some forms of protection, but because the settings must be same in the whole network, the user is always forced to use “lowest common denominator” settings. Examples of wireless encryption and authentication methods are:

- ✓ WEP64/128 (Wired Equivalent Privacy) - today not sufficient
- ✓ WPA/WPA 2 (Wi-Fi Protected Access) - preshared key
- ✓ TKIP (Temporal Key Integrity Protocol) – used by WPA/WPA2
- ✓ AES (Advanced Encryption Standard)- substitutes TKIP, used by WPA/WPA2

3.2.2 Bluetooth

Bluetooth™

Bluetooth (BT) is a wireless technology described by IEEE 802.15.1 standard which comes from 1998 when it was firstly introduced by Ericsson, IBM, Intel and Nokia gathering together into Special Interest Group (SIG). Today more that thousand companies belong into this Bluetooth-SIG.

Bluetooth technology was initially developed to provide voice and data connections within small personal networks of battery powered devices such as hands-free, mobile phone interconnecting. Due to these targets, since the beginning Bluetooth developers have been focused on low power consumption, small footprint dimensions and cost-efficiency. Bluetooth operates in unlicensed 2,4 GHz ISM band (2,402-2,480 GHz) and defines in total 79 channels with 1 MHz bandwidth. To handle coexistence and avoid interference with WLANs in the same frequency band Bluetooth introduced very unique and efficient technology called Adaptive Frequency Hopping Spread Spectrum (AFHSS) abbreviated simply as frequency hopping. Technology works on principle of switching carrier channel according to pseudorandom sequence known to both transmitter and receiver. The channel is switched with frequency 1600 changes per second. This gives Bluetooth very high stability (protection against interference) as well as security features. On the other hand Bluetooth is not such a big source of interference for the other systems operating on the same frequencies as it transmits at such a low power (1-100 mW depending on the distance, 10 times lower than WLANs). [22][25][26]

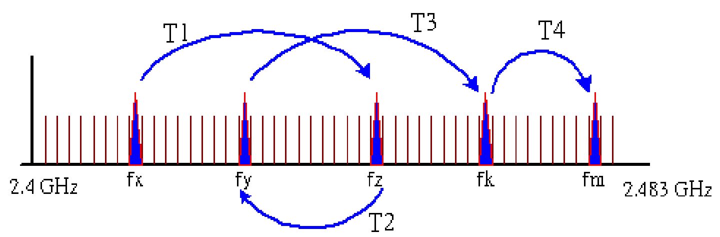


Figure 22: Bluetooth - Adaptive Frequency Hopping (T-time interval)

Most of devices used today use BT standard 2.0 or 2.1 which is capable of data transmissions within 10-1000 m with data rates 1 MBit/s brutto, ≈700 kBit/s netto. With EDR (Enhanced Data Rate) technology the speed can reach up to 2,1 MBit/s brutto, ≈ 1,5 MBit/s. BT uses 128-bit encryption to secure transmitted data. [26]

Recently the new Bluetooth Low Energy technology was introduced, bringing many interesting improvements. It is primarily intended for applications requiring episodic or periodic transfer of small amounts of data, while in between, most of the time the device is found in sleep mode with almost no consumption (average 1 μA). On the other hand in comparison with standard BT technology the data rate reaches only approx 100 kBit/s. [26]

Many Bluetooth enabled medical products are on the market today including defibrillators, pulse oximeters, weight scales, blood glucose adapters and ECG monitoring devices.

3.2.3 UWB: Ultra Wideband Technology

WiMedia ALLIANCE

UWB is a class of RF technologies that uses a very wide bandwidth to transmit signals via a wireless link. The most common UWB technology is based on WiMedia Alliance recommendations. WiMedia's UWB technology is a radio standard for high-speed, wireless connectivity with ISO certificate. UWB offers a combination of high data throughput rates (up to 480 MBit/s) and low energy consumption utilizing bands within the frequency range of 3.1 – 10.6 GHz.

WiMedia defines specifications of a UWB based Physical (PHY) and Media Access Control (MAC) layer. Other technologies including Wireless USB, Bluetooth 3.0 and other wireless designs can use these two layers and take advantage of the larger bandwidth of this technology. The combination of the PHY and MAC layers is called "common radio platform". The spectrum is divided into 14 bands and 6 band groups (BG) while each band group consists of 3 bands as described in Figure 24.

WiMedia Alliance also specifies a Multi-band Orthogonal Frequency Division Multiplexing (OFDM[12]) with over 110 sub-carriers per channel (4.125 MHz sub-carrier bandwidth), a channel bandwidth of 528 MHz and very low broadcast power that allows same-channel coexistence with narrower band devices such as 802.11a/b/g/n. Thanks to the much higher bandwidth UWB features much higher throughput, coupled with a very low RF output power. UWB offers a communication range of up to 10 meters. Figure 25: Comparison of narrowband (NB), Spread Spectrum (SS) and Ultra Wideband (UWB) signal concepts [29]

[12] OFDM: a method of encoding digital data on multiple carrier frequencies

demonstrates the wide bandwidth of UWB as compared with more traditional narrowband and spread spectrum signals (used by 802.11, Bluetooth). [29][31]

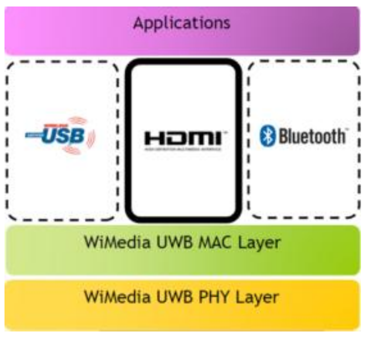


Figure 23: WiMedia UWB PHY and MAC layer [29]

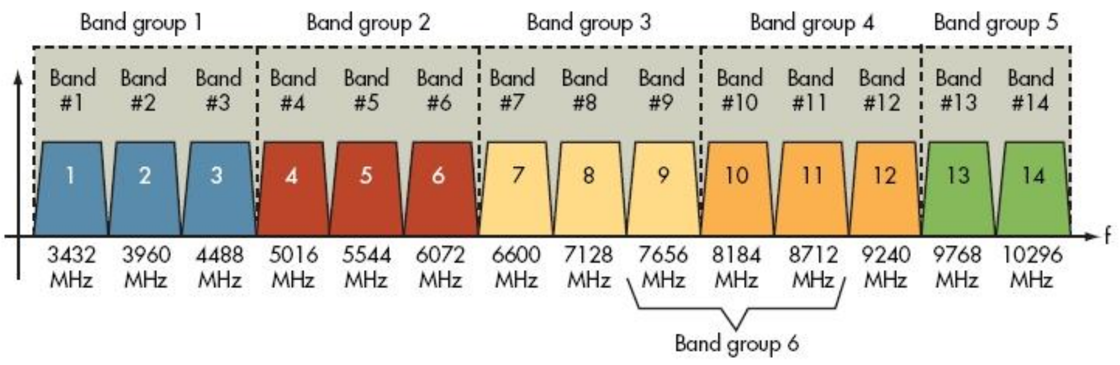


Figure 24: WiMedia UWB band allocation [30]

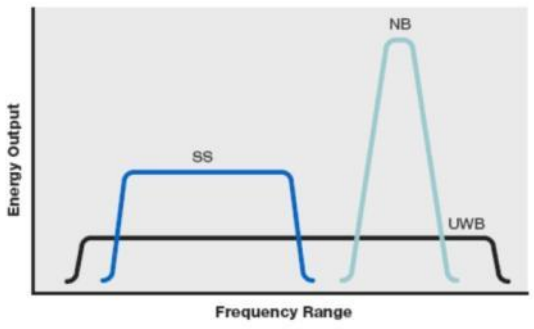


Figure 25: Comparison of narrowband (NB), Spread Spectrum (SS) and Ultra Wideband (UWB) signal concepts [29]

Recently, FDA [13] and other similar national organizations have approved Ultra Wideband (UWB) based wireless video devices for use in the OR. This is possible for their

[13] Food and Drug Administration: agency of the United States Department of Health and Human Services. The FDA is responsible for protecting and promoting public health through the regulation and supervision.

low output power, which was proved that does not interfere with other wireless devices in the common spectrum. [29]

3.2.4 ZigBee / IEEE 802.15.4

ZigBee is a specification for a products based on 802.15.4 standard, which is often used in mesh networks[14]. Its biggest advantage is in extremely low power consumption and low costs. ZigBee operates in the ISM bands; 868 MHz in Europe, 915 MHz in the USA and Australia, and 2,4 GHz in most jurisdictions worldwide. Data transfer rates vary from 20 to 250 kBits/s. Transmission distances are remarkable for a low-power solution, ranging from 10 to 75 meters, depending on power output (generally 1 mW) and environmental conditions. The technologies are mostly used in applications such as energy monitoring, process and building automation. The mesh network functionality makes it capable to cover wide areas when there are no requirements on low latency. [27][28]

3.2.5 EnOcean

The EnOcean technology is an energy harvesting (self powered) wireless technology primarily used in home sensor applications or industrial automation systems. It features extremely low power consumption, low output power and as it operates in frequency ranges 868 MHz and 315 MHz it does not interfere with other devices such as WLANs or Bluetooth. EnOcean offers one - way and bidirectional communication within range from 30 m (indoor) to 300 m (outdoor). Its supports data rates 125 kBit/s sending relatively small packets (14 Bytes). Modules based on EnOcean technology combine micro energy converters with low power electronics and enable wireless communications between battery-less wireless sensors, switches, controllers and gateways. In March 2012, the EnOcean wireless standard was ratified as the international standard ISO/IEC 14543-3-10. [32]

[14] Mesh networking (topology) is a type of networking where each node must not only capture and disseminate its own data, but also serve as a relay for other nodes, that is, it must collaborate to propagate the data in the network.

4 WLAN in OR

Bringing new medical product from company´s test room into medical field always requires a lot of effort and time. It is not only development part which needs to be done. Before first product´s release also a lot of administration, negotiating and testing in approved laboratories must be undertaken to fulfill all requirements given by international and local standards. For wireless devices the way is maybe even longer as the wireless technologies in healthcare are quite a new topic and have not been examined sufficiently compared to other medical electronic (ME) devices. Therefore very strict demands (maybe more than necessary) are required. The main requirements are related to personnel safety, image quality, operating distance, radiation level, and co-existence with other RF equipment. In the past many wireless technologies have failed to get a permission due to bandwidth limitations, interference with other OR equipment or simply were not robust enough to provide a reliable connection. [29]

4.1 Standards and Guidelines

All the requirements on ME devices are specified by these two packages, while the difference is in their law relevancy.

- Guidelines are drafts and explanations to guide company developers in setting standards or directives to produce device able of being released.
- Standards are, in terms of law, on the top of importance. These precisely define limit values, dimensions, materials, steps of measuring and testing process, which are (going to be) applied on every device applying for accreditation.

The priorities of all the requirements are firstly protection of human beings against influences of RF signals as the high output power of any device could be harmful and secondly protection of devices in order not to interfere with the others, respectively not to decrease their ability to communicate or work properly. An ability of a device to share the same electromagnetic environment without disrupting other devices and causing harmful emission is called EMC (Electromagnetic Compatibility). EMC is determined by following issues, characteristics of a device [33]:

- Electromagnetic Interference (EMI)
- Susceptibility / Immunity : ability to perform reliably in presence of external disturbances; susceptible device is more likely to be influenced while immune device is more stable
- Emission: unexpected generation of electromagnetic energy/field
- Coupling: way of transferring energy from source to victim device/person e.g. inductive, capacitive, radiative

Worldwide there are many organizations defining EMC standards from different aspects. For instance just those of maximal liability.

- International Electrotechnical Commission (IEC)
- International Organization for Standardization (ISO)
- Comité Européen de Normalisation (CEN)- Europe [15]
- Comité Européen de Normalisation Electrotechniques (CENELEC)- Europe [14]
- European Telecommunications Standards Institute (ETSI)- Europe [14]
- The Federal Communications Commission (FCC)- US

Regarding OrthoPilot project, apart from those guidelines and standards which were applied in previous generations, newly the directives describing use of wireless technologies and their influence are (are going to be) added. Every wirelessly coupled device is regulated and standardized by many directives. Requirements and liability/validity of a particular directive may vary throughout the countries, but as the OrthoPilot has always been intended for a world market, it will have to follow all these [34]:

- **ISO 14971:** ***Medical devices - Application of risk management to medical devices***
  Specifies a process for a manufacturer to identify the hazards associated with medical devices to estimate and evaluate the associated risks, to control these risks, and to monitor the effectiveness of the controls. [35]
- **IEC 60601:** ***Medical electrical Equipment***
  International standard was published in 1977. Current 3rd edition consists of a general standard, about 10 collateral standards (numbered 60601-1-X) defining the requirements for certain aspects of safety (Protection for diagnostic use of X-rays), and about 60 particular standards (numbered 60601-2-X) defining specific devices (MR scanners, Ultrasounds etc.). These standards have been accepted by many countries as a national standard. For OrthoPilot purposes general standard 60601-1 and collateral standard 60601-1-2 are of biggest importance. [36][37]

[15] Alltogeher called European Standardisation Organisations (ESOs). These produces European Standards (norms) - EN

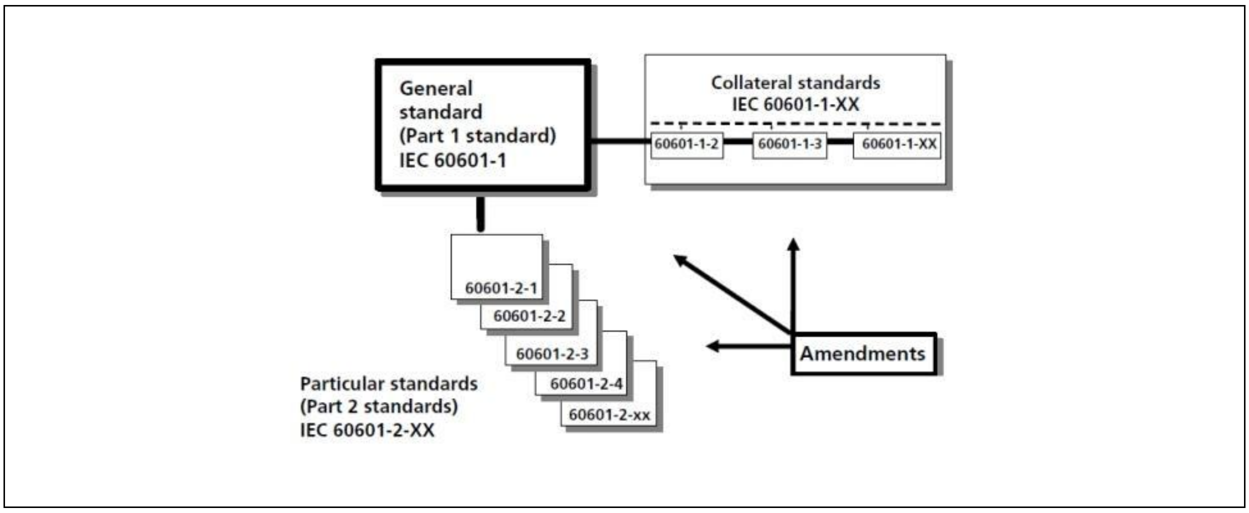

General standard (Part 1 standard) IEC 60601-1
Collateral standards IEC 60601-1-XX
60601-1-2
60601-1-3
60601-1-XX
60601-2-1
60601-2-2
60601-2-3
60601-2-4
60601-2-xx
Particular standards (Part 2 standards) IEC 60601-2-XX
Amendments

Figure 26: IEC 60601-1 and its collateral and particular standards [37]

- ✓ 60601-1:2005: General requirements for basic safety and essential performance
- ✓ 60601-1-2:2010: Electromagnetic Compatibility

✗ **IEC 62304: 2006: *Medical device software - Software life cycle processes***

This standard applies to the development and maintenance of medical device software and defines life cycle requirements. [38]

✗ **IEC 8000: 2010: *Application of risk management for IT-networks incorporating medical devices***

✗ **FCC Part 15: *Radio frequency devices***

US standard contains rules and regulations divided into subparts regarding unlicensed transmissions. Important ones for OrthoPilot products are marked with * [39]:

- ✓ FCC Part 15 Subpart A: General terms and definitions*
- ✓ FCC Part 15 Subpart B: Unintentional radiators
- ✓ FCC Part 15 Subpart C: Intentional radiators (include WLAN, Bluetooth)*
- ✓ FCC Part 15 Subpart E: Unlicensed NII (U-NII) devices*
- ✓ FCC Part 15 Subpart F: Ultra-wideband (UWB) operations*

✗ **Directive 1999/5/EC (R&TTE) + Guideline**

Defines the rules for the placing Radio and Telecommunications Terminal Equipment on the market. It covers all radio equipment and all equipment intended to be connected to public telecommunications networks. [41] Standards were developed mostly by ETSI and CENELEC. [42]

✗ **FDA Draft Guidance for Industry and FDA Staff: *Radio-Frequency Wireless Technology in Medical Devices***

FDA has released this draft guidance to help companies in medical industry to design, develop and evaluate their RF wireless medical devices and to get oriented in national (US) and international standards as well as regulatory requirements. This document also discusses the following considerations specific for RF wireless medical devices which should be considered in addition to general medical device requirements:

- ✓ design and development activities
- ✓ risk management activities
- ✓ testing
- ✓ labeling
- ✓ purchasing controls
- ✓ acceptance activities
- ✓ servicing
- ✓ corrective and preventive action (CAPA) considerations

The whole draft should be considered as a recommendation only as it does not establish any enforceable responsibilities. [43]

- ✗ **FDA Draft Guidance for Industry and FDA Staff :** ***Mobile Medical Applications***
  As there are plenty of mobile applications being developed nowadays this draft clarifies the types of mobile medical applications (apps) to which the FDA intends to apply its authority. It should be considered only as a recommendation which is good to follow in order to get an authority. [44]
- ✗ **ISO 13849-1: 2006:** ***Safety of machinery - Safety-related parts of control systems***
  Provides safety requirements and guidance on the principles for the design and integration of safety-related parts of control systems including the design of software
- ✗ **AAMI TIR 18:1997:** ***Guidance on Electromagnetic Compatibility of Medical Devices for Clinical/Biomedical Engineers***
  Technical Information Report was published by the Association for the Advancement of Medical Instrumentation (AAMI) to provide guidelines for management of EMI in healthcare environment as well as management of wireless technologies. [45]
- ✗ **IEEE/ANSI C63.18: 2007:** ***Recommended Practice for an On-Site, Ad Hoc Test Method for Estimating Electromagnetic Immunity of Medical Devices to Radiated Radio-Frequency (RF) Emissions from RF Transmitters***
  Document published by IEEE and ANSI (American National Standards Institute) serves as a guide for healthcare organizations and recommends the steps to be taken to identify possible EMI of devices already used or intended to use in medical environment. [45]

4.2 Acceptance in different countries

Not only culture but also the laws and regulations differ between two particular countries. Therefore the company which wants to sell its products abroad must be aware of differences in requirements and include this information into design and development process as well as marketing strategy.

As an example we can mention acceptance of 5 GHz channels in US. When a German company decides to sell product using wireless standard 802.11b (Wi-Fi) in US, it must be careful of not using channels 12 and 13 as these are not allowed in North America. For this reason either different versions of product (hardware) or flexible software settings must be available. List bellow summarizes important aspects and questions which should not be forgotten:

- ✗ Which laws and standards are required in particular country?
- ✗ Which frequency bands/channels are allowed? Any special conditions?
- ✗ Are different product/software versions needed?
- ✗ How must be the device labeled?
- ✗ Which documentation and in which language(s) is needed?

To introduce a product into a particular country, always test report according to specified (mostly any international) standard is required. Beside this, some countries, such as South Korea, Japan or China require local testing and therefore 1 or more sample product. In European countries, EU Declaration of Conformity (DoC) is needed. This document (certificate) as well as CE[16] (European Conformity) logo is one of the common ways of proving, that the product meets all directives written in the body of the document.

[16] CE: abbreviation of French: Conformité Européenne

5 Wireless desktop cloning

This part of my thesis deals with possibilities of wireless desktop cloning, in other words, wireless transmission (streaming) of image data. My task in this topic was to explore possible solutions including hardware and software variants.

5.1 Compression

When we consider screen of 1280x720 pixel resolution, 32 bit depth; 1 frame has the size of ≈ 30 Mbit (1280*720*32 = 29491200). For transmitting at least 25 frames per second (FPS), technology supporting throughputs at least 750 Mbit/s would be needed. Actually none of the wireless technologies mentioned earlier fulfill this requirement, but even though it is possible thanks to existence of compression methods. These can be divided according to basic characteristics: [46]

- ***Spatial vs. Temporal Compression***

Spatial compression applies algorithm to the whole frame (picture), without any relations to other frames. Every complete frame is compressed and useless data are removed. On the other hand, temporal compression searches only for the changes between subsequent frames (images), which decrease amount of data to be stored / transferred. Some compression methods combine both of these concepts (e.g. every fifth frame is compressed completely, others carry only the changes).

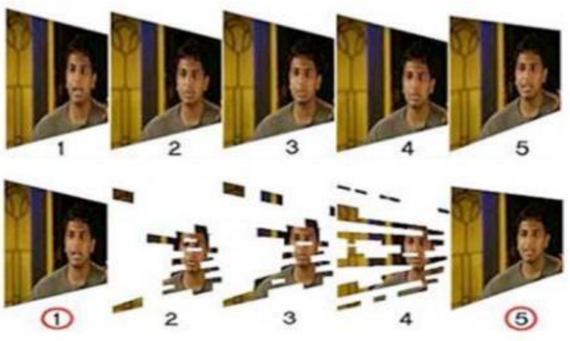


Figure 27: Spatial vs. temporal compression (JPEG-top, MPEG-bottom) [46]

- ***Lossy vs. Non-Lossy Compression***

When lossy compression is used, some of the image information is lost. Even that this looks like limitation, in respect to transferred data volume, lossy compression has its advantage. Finally all of the loosy compression methods usually allow user to set level of lost information and until certain value, compression in the final image isn't noticeable.

Here comes small overview and description of the most popular compression methods: [46]

- ***JPEG (Joint Photographic Expert Group)***

Spatial compression algorithm mostly focused on single frame compression (photographs) or very low frame-rate video. Until compression factor 10 (10:1), no significant loss of quality may be observed. Higher factors (up to 40) cause clearly visible artifacts.

- ***M-JPEG (Motion JPEG)***

Variant of JPEG, but intended for video sequences. JPEG frames are played in fast subsequence to give the motion illusion. Is not a temporal compression, every single frame is compressed completely.

- ***MPEG-2 (Moving Pictures Expert Group)***

Standard temporal compression algorithm for broadcasting quality video. Up to factor 30 (30:1) may be used to maintain excellent quality

- ***H.264, MPEG-4, AVC (Advanced Video Coding)***

The newest standards offering 50% better efficiency compared to MPEG-2 while keeping the same image quality (ratios up to 60:1). Standards are universal, designed to suit most of currently used applications (low/high resolutions and low/high bandwidths).

5.2 MaxiVista

After previous research, some of software/hardware solutions were chosen and bought/borrowed in order to be tested.

5.2.1 Testing

MaxiVista is a software solution for multi monitor-system. To connect to other PC(s), TCP/IP[17] protocol is used. Physically the connection is done using Firewire (1394) cable, LAN, WLAN or USB. Software offers three modes: extended screen (allows user to extend up to 3 other computers), mirroring (current version supports only one computer, next release promises more) and remote control. After installing the main application (MaxiVista Server) it is necessary to copy generated file (MaxiVista Viewer) to other computers. Both Server/Viewer can be run manually or automatically when copied to Window´s start-up

[17] Transmission Control Protocol/Internet Protocol

folder. Furthermore Viewer can be run as a service, which allows user to log into locked secondary PC from the primary PC in remote control mode. It is recommended to use "Performance optimizer" in option menu to automatically set the best possible performance. When testing before and after, the big difference in speed and fluency could have been observed.

Firstly MaxiVista was tested using 1394 connection, which is capable of speed up to 400 Mbps. IP addresses were assigned automatically as well as MaxiVista's "Network" setting was let to be automatic (default). Connecting to the Viewer program was always error free when running both server/viewer on start-up but was not able to connect if the viewer was installed as a service. To test the quality, sample HD video (720p/24fps) was chosen. The video was played using Windows Media Player 9.0. On the primary PC the video run at 23,8 fps while in extended mode (video window dragged to second monitor) only 16,5 and in mirroring mode 18,5 fps. Because MaxiVista transmits only changes in the image and not the complete image at each frame, network utilization never exceeded 5 %. Secondly the connection was made using 100 Mbps LAN cable (crossed!!!) with similar results: 23fps primary PC, 17 fps extended screen, 20 fps in mirroring mode. Network utilization was about 12 %. The IP addresses had to be chosen manually (192.168.0.1. and 192.168.0.2) because the server could not find the viewer when they were automatically assigned. Nevertheless MaxiVista "Network" setting remained automatic. The most important part was testing MaxiVista software when connecting two PCs wirelessly, because this is the kind of connection which would be eventually used. For wireless connection pair of two WLAN USB sticks was used. Although the sticks provide speed up to 300 Mbps using the standard 802.11n, under Windows XP it was not possible to set up an ad-hoc network faster than 54 Mbps. The sample video was displayed at 13,3 fps in extended mode and 14 fps in mirroring mode. Network utilization was approximately 20%.

Next task was to test the ability to transmit screen displaying OrthoPilot software. Versions TKA 4.4 and HTO-3D 2.0 was used. It is noted by producer of MaxiVista that the program doesn't support hardware acceleration such as DirectX or OpenGL, which finally showed up to be a problem in our case. The newer version HTO-3D 2.0 worked error free and very reliably with or without graphic card support, fluently and without noticeable delay in both extended screen and mirroring modes. On the contrary in TKA version strange behavior was observed when some elements of the screen were not displayed properly. The result was better when turning off hardware acceleration, but never perfect as can be seen on the figures below. The behavior does not depend on the connection used. Probably these bugs could be solved by recoding/fixing the program, but still the absence of hardware acceleration is partially limiting. MaxiVista always redraws (twice) the screen whenever the mode is switched or started. This behavior does not look very professional for device intended for OR.

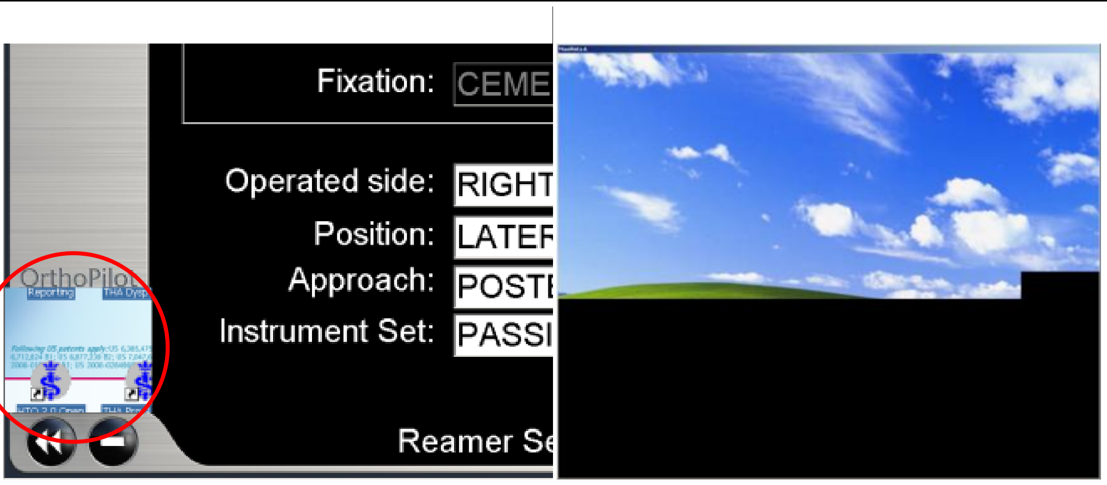


Figure 28: MaxiVista software: left: detail showing problem in TKA application with displaying certain components (hardware acceleration), right: flickering/loading the screen when the program starts

5.2.2 Conclusion

The MaxiVista solution is limited by the necessity of having second PC running operation system and also controlling the program by tray icon on the task bar is not very comfortable. Sometimes software behaves in unexpected way and although everything is set well, program will not connect to viewer until restarted.

Although the transmission speed is sufficient for our, not so demanding purposes, the manipulation, user friendliness and reliability is limited, especially due to missing hardware acceleration support. The MaxiVista fulfilled my expectation of 99 $ device which can be used for home purposes but for use in medical field does not look very convincing.

5.3 RIOXO RX-VN10

5.3.1 Testing

Device RIOXO RX-VN10 is produced by Barox Kommunikation AG and offers a hardware solution for HD video transmission over IP net. The easiest system consists of pair of devices (transmitter/receiver) which are interconnected over LAN/WAN network, but connecting multiple receivers to one source is also possible. Both devices are the same and different function is assigned only by configuration. The data stream is encoded at the one side using JPEG 2000 compression and decoded by receiver on the other. As a video input/output DVI-I connector is used and HDMI and VGA (input only) connection is also possible by means of simple converters. The connections between encoders and decoders are defined by simple IP-addressing. To change these settings, devices are accessible using a web browser on the certain IP address.

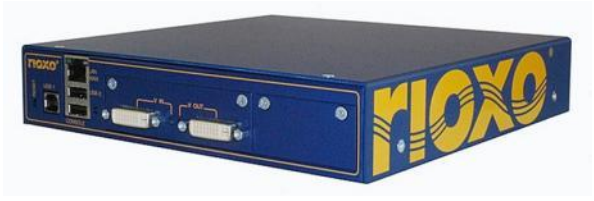


Figure 29: RIOXO RX-VN10 encoder/decoder

In the experiment, devices were connected to a wireless router (here used only as a switch) which was connected to the LAN card of the computer. Video source (output B from the graphic card) was connected to transmitter module and the output from receiver was wired to a second monitor, both using DVI cable. Using web browser both devices were accessed, allowing us to change any setting at any time. Default IP addresses were used (192.168.1.201 for transmitter and 192.168.1.200 for receiver).

Although the devices were connected and set up properly, sometimes there was no stream received on the receiver's side until either modules or computer was restarted. In some cases ("Force HDCP" setting unmarked) green screen was suddenly displayed on the second monitor.

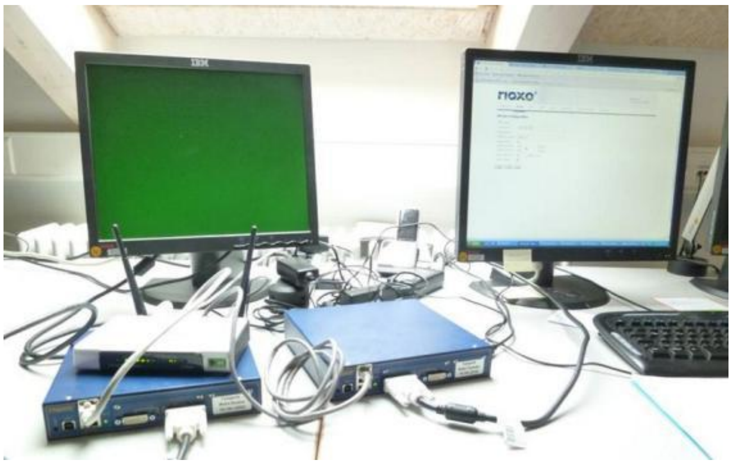


Figure 30: Unpredictable behaviour-green screen

The settings allow user to select max. data rate from 1 to 800 Mbit/s but from 90 Mbit/s the secondary display starts to flicker or in higher bit rates (approx >150) does not display anything. The bit rate has a big influence on the quality of the streamed image as can be seen on the figures below. When tested on OrthoPilot software TKA 4.3, 90 Mbit/s was sufficient but on the other hand most of common applications (MS Excel, web browser etc.) were displayed very poorly at this particular bit rate. Deviations were observed only on static images. As for HD video (720p/24fps) streaming transmission performed very well and fluently.

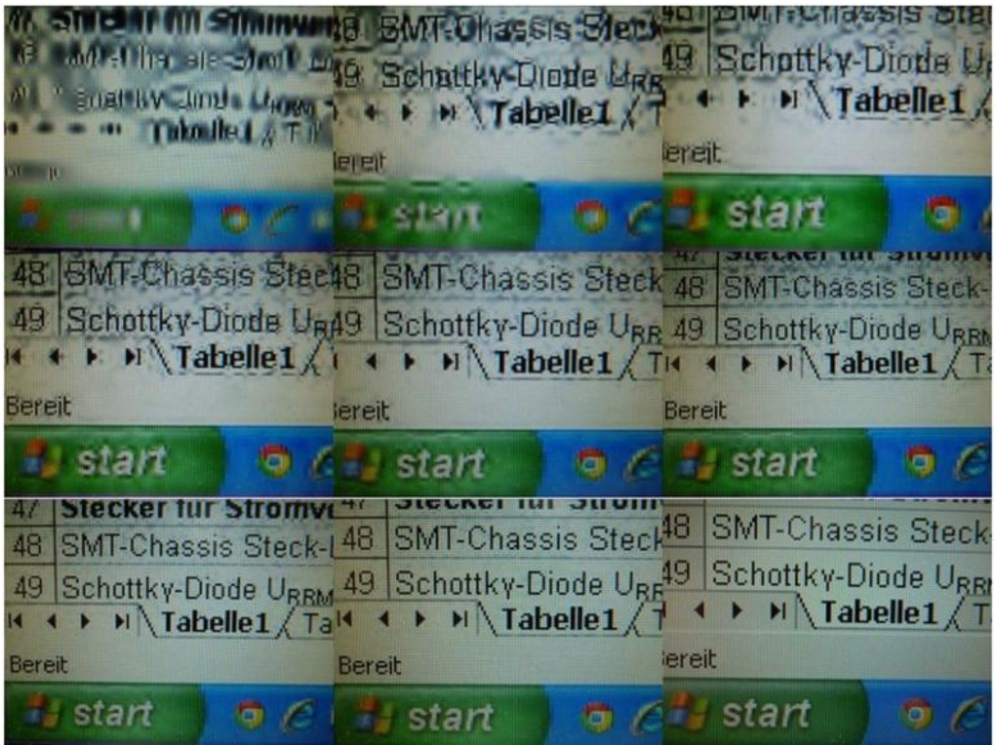


Figure 31: Quality of image depending on bit rate (from left top): 10, 30, 50, 70, 90, 10,130, 150 Mbps, original

Finally it was found out that the flickering happens due to router, because it supports only 10/100 Mbit Ethernet. When the pair of devices were connected directly (using proper CAT5 or higher speed LAN cable) flickering stopped and the connection was stable even at the highest possible bit rate of 800 Mbps. As the manufacturer mentions, there is certain latency (3 FPS + network delay) which can be observed for example on mouse cursor movement. It is delayed and also a bit slower. This behavior does not depend on the bit rate setting. It is very difficult and inconvenient to connect the devices wirelessly. To do so, two access points are needed (each on one side), each of them connected to a device over LAN cable. For sufficient performance the standard 802.11n Wi-Fi would need to be used, because speed of 56 Mbps of standards 802.11b/g is not enough. Last but not least the size and shape is also limiting as well as missing medical accreditation. Complete overview can be seen in the table below.

PROS	CONS
Output directly from the graphic card -> source is always identical with original (e.g. no hardware acceleration limitations)	Price cca. 2000 €
TKA 4.3 works fine even at 90 Mbps	No medical accreditation
Multicasting (more receiver monitors)	Noticeable delay
	In lower bit rates image noticeably blurred
	Design, size
	Difficulties in wireless connection- connection to device only over LAN cable
	Unpredictable behavior, unreliability

Table 3: RIOXO pros/cons overview

5.3.2 Conclusion

Although the Rioxo solution offers interesting features such as streaming to multiple monitors over long distances using existing LAN/WAN, it is not very suitable for use in OR. Every device used in medical field needs to be reliable and easy to use but after testing, this cannot be claimed about this device. Compared to NDS Zero Wire device for the same price and with medical accreditation the device lacks a lot of crucial features.

5.4 NDS ZeroWire Duo

5.4.1 Testing

Test was done using pair of devices (Rx/Tx) ZeroWire Duo High-Definition Transmit and Receive system (made by NDS Surgical Imaging) while transmitter was connected to PC equipped with a monitor and receiver was connected directly to the second monitor. All connections were done using DVI-D port. For testing the quality of transmission, two HD video samples in format 1080p and 720p, both 30 fps, were used. Test was carried out in the experiment room with plenty of electromagnetic devices and metallic objects. Up to the tested distance of approx. 7 m the transmission of 720p sample worked without any noticeable errors when the devices were facing each other in the same height of about 0,5 m above ground. When moving receiver or transmitter in the different positions small disturbances visible in the displayed video occurred. Stream stopped to be fluent only in some extreme cases (positions): Tx/Rx facing the ground, metallic object right in front of one of the device. The transmission was perfect even in not recommended positions (without any moving!) of devices e.g. different height, facing different direction, horizontal/vertical position.

Figure 32: NDS ZeroWire Duo Transmitter Receiver system

For full HD sample (1080p) no credible result can be claimed, because the video in this quality could not be displayed fluently even on the transmitting side due to the insufficient hardware quality. Anyway no bigger disturbances were observed compared to the previous 720p resolution.

5.4.2 Conclusion

The results were surprisingly satisfactory and seem to match the parameters given by manufacturer. If all the requirements are followed, the device works very reliable even in high resolutions. Both transmitter and receiver have to be connected via DVI-D connector (digital I/O). It was found out that device could not operate via VGA (analog) port although the VGA-DVI-I adapter was used. One small disadvantage might be that the devices transmit only image data, but also keyboard or mouse events data could possibly be transmitted along. Big advantage is, that the system is already fully approved for use in medical field. For the intended use in wireless OrthoPilot navigation system the device seems to be suitable.

6 WLAN, Bluetooth module

Following chapter describes wireless modules, which were chosen to substitute current cable connections. According to the research made by another co-worker in the past, ready-to-embed modules from company connectBlue (connectBlue AB, Malmö, Sweden) were selected, mainly for their features which fulfill the requirements of OrthoPilot application.

6.1 Requirements [13]

- **Minimal data rate 115 kBit/s**
  As the modules will be integrated into communication between camera→PC and, remote controller→ PC, there are no reasonable requirements for higher data rates because the speed is limited by the camera itself to 115 Kbit/s. This is actually very sufficient because the transferred data are of very small sizes. Commands and answers (transformation matrices) which are sent and received in ASCII coding are of approx length 100 characters that means sizes of approx. 100 bytes. In a case of remote controller the data sizes are not specified exactly, as far as there is no final product, but the device is supposed to communicate via simple commands e.g. command 0x40 (in hexadecimal interpretation → 01000000 binary → 1 byte) when forward button is pressed. Therefore the amount of data will be even smaller and the minimal speed 115 Kbit/s offered by module will be certainly sufficient.
- **Internal antenna possibility**
  Internal antenna is an obligatory feature as it can save costs and allows the wireless device reach smaller dimensions and convenient shapes. Because both camera and wireless devices will be used only within OR (range of max 10 m), there is no need to implement external antenna to increase signal strength. Nevertheless in case of using internal antenna there is a requirement for the housing of the device to be made of RF transparent material. connectBlue offers every module in two different modifications: with internal antenna (e.g. OWS451i) and with output pin for external antenna (OWS451x).
- **Safety features**
  Module should support at least some basic encryption methods to secure transmitted data. Hacking into communication and listening to data stream wouldn´t be so dangerous because the coordinates or button actions are not much valuable information. Much worse is a case when the transmitted data would be modified because the success of the whole intervention depends primarily on the accuracy and correctness of acquired coordinates.
- **Driver compatibility with *Windows XP* and *Windows 7***

Current OrthoPilot version operates under *Windows XP* but the next generation is supposed to use *Windows 7*. Nevertheless the backward compatibility must be ensured.

- **Availability at least 5-7 years after purchase**

  This is very important point, because Aesculap acts more or less as a mediator which uses external suppliers´ products to build its own systems. Therefore to be able to give service guarantee of seven years to customer, Aesculap has to be sure that the spare parts from external supplier will be available for at least the same time or to store spare parts in the stock which would be risky and financially the worst solution. connectBlue promises that their products will be available even after 5 years and that new versions/products will be backwards compatible with developed hardware and software.
- **SPP (Serial Port Profile)**

  SPP defines the protocols and procedures for a module using Bluetooth for serial (e.g. RS232) cable emulation. Such a device is afterwards presented to host computer as a virtual serial (COM) port object even though there is no physical cable connection. Any application may be run on either device, using the virtual serial port as if there was a real serial port. This feature is required because the camera uses serial connection and all OrthoPilot software has been coded using serial (COM) port classes. Serial cable emulation is the easiest way how to establish wireless connection without need to change software. connectBlue call their modules Serial Port Adapters (SPA) as they all act as a converters between serial and Bluetooth/Wi-Fi technology.
- **Possible with medical accreditation**

  Complete OrthoPilot system before being launched must have got a medical accreditation confirming that the device fulfills all requirements (see Standards and Guidelines 4.1). This process is much faster, cost-saving and generally easier when the individual components have already their own medical accreditations. connectBlue is one of few companies producing wireless modules which delivers its products with medical accreditation.
- **Low power consumption**

  Future generation of OrthoPilot is supposed to use accumulator to supply system with energy. Therefore any device must be optimized with respect to energy consumption as much as possible. Energy saving is important especially in case of remote controller, which will be supplied by a small (approx 1000 mAh) rechargeable battery.
- **Simple**

OrthoPilot requires so that every component is as simple as possible. According to everyone's experience, simple devices feature higher reliability and behave (usually) in the expected and required way which is crucial aspect of every medical device.

- **Communication must be established automatically**
  Once the modules' settings are given, connection of one module to another or to the existing network must be established automatically and as fast as possible. System must be sure that after certain time interval after switching the device ON, the connection will be available.

Another advantage of connectBlues products is that both modules Bluetooth and Wi-Fi are of the same sizes and the way of connecting and pin's purposes (apart from little differences) are in both cases the same. Further both technologies use the same software (Serial Port Adapter Toolbox) for configuring the modules and many commands have in common. All this is an advantage, because the performance and functionality can be easily and fairly compared using only one prototype device and afterwards can be definitively decided for one or another wireless technology. Additionally connectBlue company releases in regular intervals firmware upgrades, which can be installed into module's EEPROM[18] (Electrically Erasable Programmable Read-Only Memory) to fix bugs or/and add new features. To load firmware into memory, special software (Flash Loader) from connectBlue is used.

In the following subchapters modules which were used and tested will be described in closer details. Modules' description is focused only on the features/commands which are related to module's function in the OrthoPilot system. Complete description is available in product's basic description and datasheets at connectBlue web pages (e.g. [46], [48]). Priority is given to description of WLAN module as these modules were used and tested in OrthoPilot application for the first time. Bluetooth modules have already been tested and described by previous trainees, e.g. [13]

6.2 OBS433 and OWS451 Serial Port Adapter

Both modules *OBS433* (Bluetooth) and *OWS451* (WLAN) support all requirements specified above. All software for communication is embedded in the module so that the user just sends raw serial data which are converted by module in Bluetooth modulated signal or TCP/UDP[19] packets (WLAN). On the other side, another Bluetooth/WLAN device receives

[18] Memory used in computers and other electronic devices to store small amounts of data that must be saved when device is switched off. EEPROM is realized as arrays of floating-gate transistors.

[19] Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) are protocols of the transport layer for the establishment of host-to-host communications.

data and converts them back in serial. Module is connected to the host device (PCB) via solder pads on the bottom side (see Figure 34). Two rows of pads on the edge (J6) are used for direct soldering on the PCB (were not used) while rows J2 and J3 are used for removable connection using board-to-board connector Samtec *FSI-120-03-G-D-M-AB* (Samtec USA, New Albany, USA) and two M2 screws.

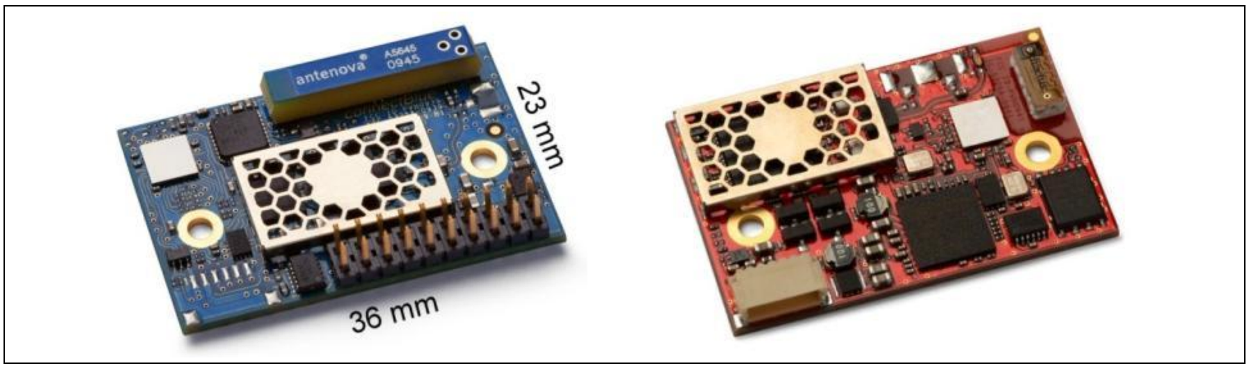


Figure 33: connectBlue Bluetooth OBS433i-02 (left) and WLAN OWS451i-04 (right) modules [46][48]

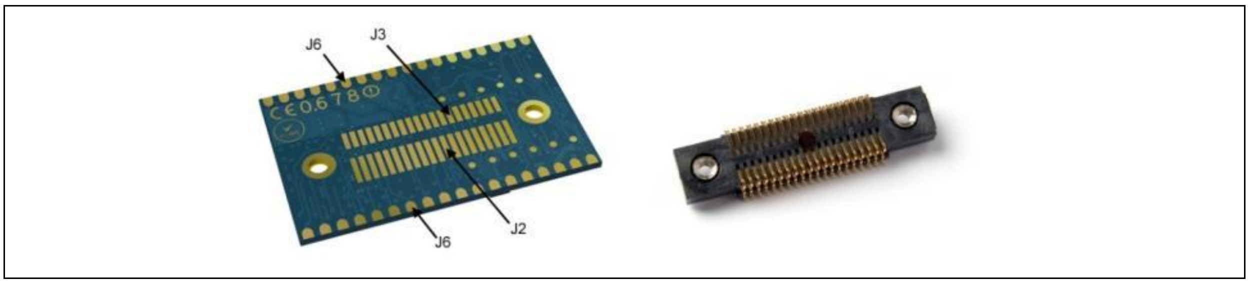


Figure 34: Backside solder pads (left); double row connector Samtec (right) [48]

Modules support standards Bluetooth 2.1+EDR or IEEE 802.11a/b/g/n (n-single stream) and baud rates up to 1,8 Mbit/s - serial communication with host computer/device over UART logic levels. Module is connected to PC over special Serial to USB converter which is available from connectBlue (see Figure 35).

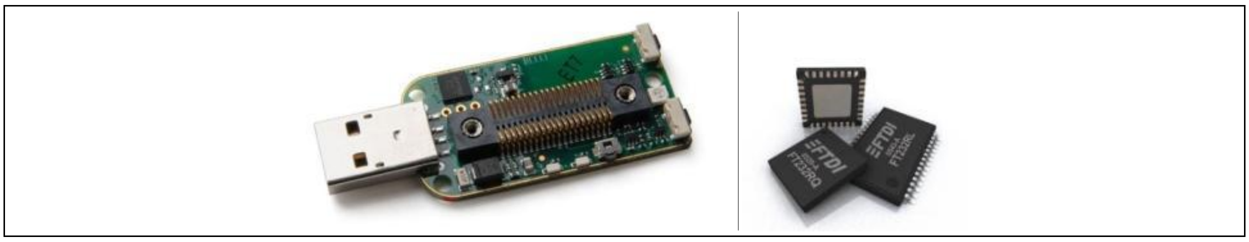


Figure 35: connectBlue USB to serial converter (left) based on FTDI chip FT232R (right) [48]

Because this is meant only for configuring and upgrading modules and not for use in applications for permanent data transfer (does not even have medical accreditation) Aesculap has developed its own converter based on the structure and schematic of the commercial one.

“Heart” of the converter is hidden in a small IC called FTDI chip (Future Technology Devices International ltd., Glasgow, United Kingdom). This company is generally known for their microchip solutions for converting serial peripherals to USB. The converter with the module mounted on it (using Samtec connector) is presented to a PC as a virtual COM port and any software can send and receive data from this port as if there was serial cable connected.

Modules can operate in two basic modes: AT[20] mode and Data mode. Data mode is default mode after each reset and is used for normal data transmission between PC and peripheral device. AT mode is closely related to AT commands and both are described in the following subchapter

6.3 AT mode/AT commands

Note: Following text subjects to WLAN module, although many commands are similar with BT modules. Every AT command must be terminated with <CR> control character (“Carriage return”; in C and many others programming languages “\r”) which returns on the beginning of a line of text. Escape sequence is sent without <CR> character!!!

Purpose of AT mode is to allow user to configure module and reset its parameters. Any settings, values and properties of a module can be set/changed using appropriate AT command. These all are specified in details in connectBlue Wireless LAN SPA AT Commands document [49] and the selection of those related to my thesis in **Chyba! Nenalezen zdroj odkazů.**.

6.3.1 Escape sequence, AT*ADDM

In order to switch from data mode to AT mode special escape sequence must be sent to the module. Default escape sequence consists of three forward slashes “///” but the escape character can be changed (e.g. “@@@”, “+++“). Another condition is the time intervals of relaxed state, when there is no data transmission on the line, which must precede and follow the escape sequence. The default intervals are set to 1100 ms. To move from AT mode back to data mode, Enter Data Mode command AT*ADDM<CR> must be sent (Figure 46).

[20] AT = attention. AT commands also called Hayes commands is a specific command language originally developed for Hayes Smartmodem in 1981 to distinguish between modem data transfer and configuration data intended for the modem itself.

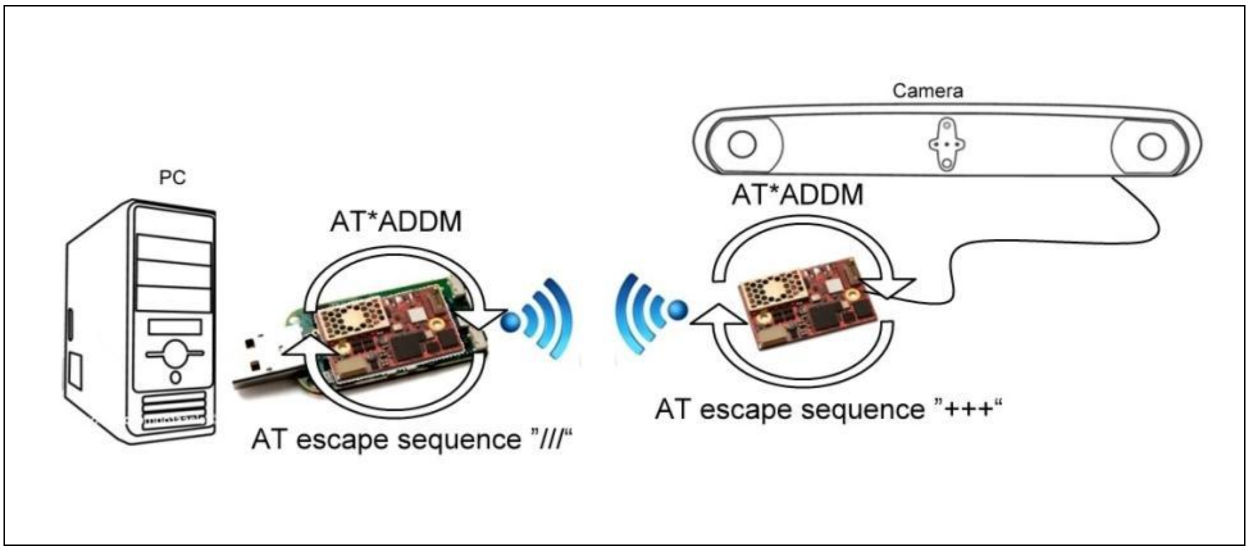
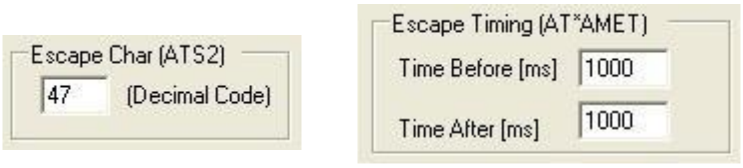


*Figure 36: Escape sequence, AT*ADDM, configuration over air*


Escape Char (ATS2)
47 (Decimal Code)

Escape Timing (AT*AMET)
Time Before [ms] 1000
Time After [ms] 1000

Figure 37: SPA Toolbox: Escape sequence setting

When the module enters AT mode, by default it disconnects all remote peers and connects them as soon as it is back in data mode. To keep peers connected during AT mode S-register 4006 must be set to "1", command ATS4006 = 1<CR>.

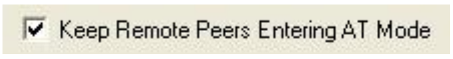
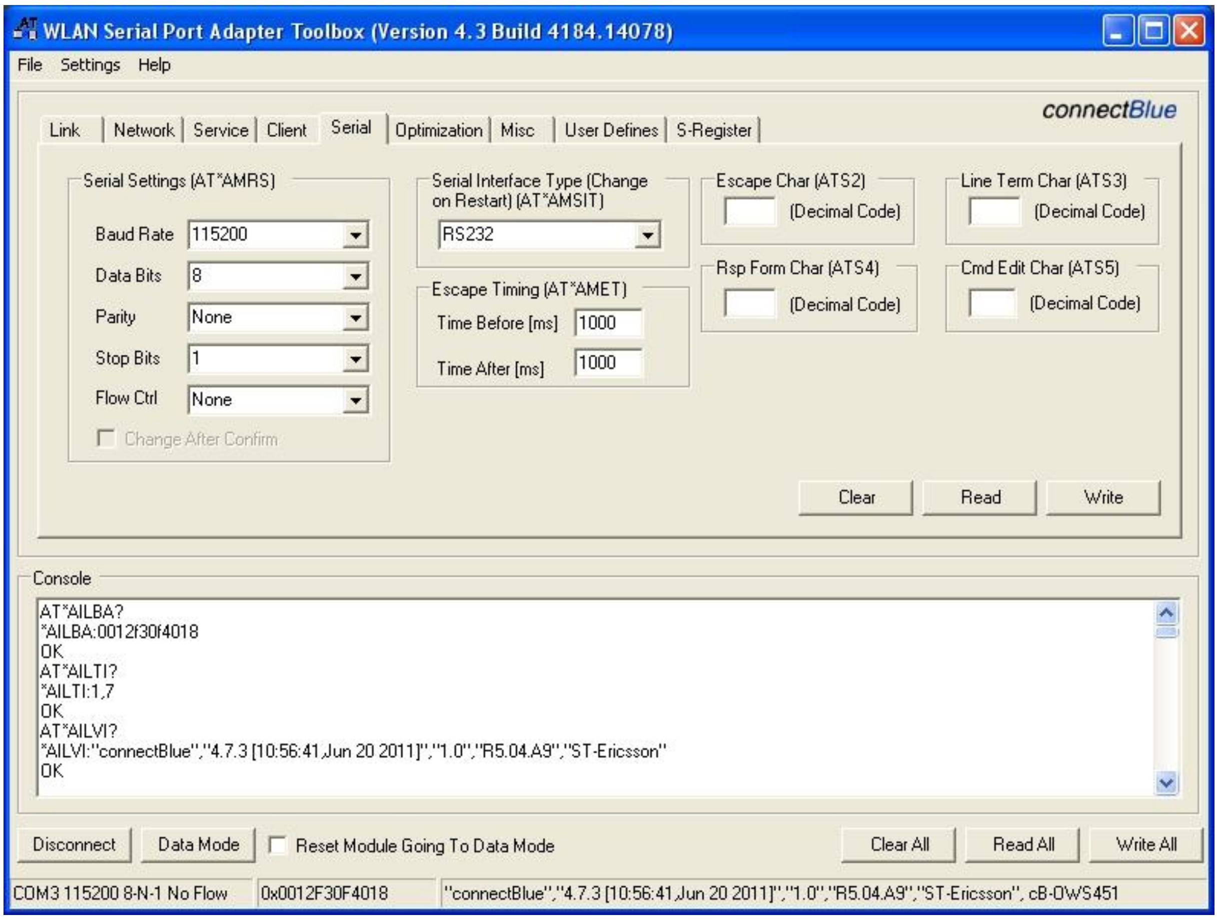

[✓] Keep Remote Peers Entering AT Mode

Figure 38: SPA Toolbox: Keep remote peers

To configure parameters which will be fixed and will not be changed during the application progress, connectBlue offers more convenient way - Serial Port Adapter Toolbox. The software allows user to select required property and its parameters from GUI (Graphical User Interface) without need to remember and write every single command. Anyway SPA Toolbox sends exactly the same commands as can be finally seen in the console which is part of the application. The same configuring may be done over any serial terminal e.g. HTerm, Realterm and of course over any application working with serial data transmission which needs to reset parameters anytime in progress.



Serial Settings (AT*AMRS)
Baud Rate: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Ctrl: None
Change After Confirm

Serial Interface Type (Change on Restart) (AT*AMSIT): RS232

Escape Timing (AT*AMET)
Time Before [ms]: 1000
Time After [ms]: 1000

Escape Char (ATS2) (Decimal Code)
Line Term Char (ATS3) (Decimal Code)
Rsp Form Char (ATS4) (Decimal Code)
Cmd Edit Char (ATS5) (Decimal Code)

```
AT*AILBA?
*AILBA:0012f30f4018
OK
AT*AILTI?
*AILTI:1,7
OK
AT*AILVI?
*AILVI:"connectBlue","4.7.3 [10:56:41,Jun 20 2011]","1.0","R5.04.A9","ST-Ericsson"
OK
```

Reset Module Going To Data Mode

```
COM3 115200 8-N-1 No Flow | 0x0012F30F4018 | "connectBlue","4.7.3 [10:56:41,Jun 20 2011]","1.0","R5.04.A9","ST-Ericsson", cB-OWS451
```

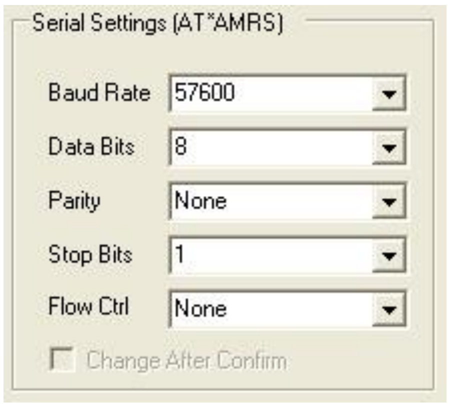
Figure 39: SPA Toolbox: sample

In the following text the most important commands related to my application will be described. Some of them are described in closer details because they needed to be changed while the application is running and therefore exact structure of the command is needed as it is implemented into the source code. Other properties of the module were set firmly and don´t change their setting all time long. As was mentioned above, all these settings have also their own AT commands but these are not described as the command structure is not necessarily needed.

6.3.2 Serial settings

As was already mentioned, it is necessary to synchronize serial settings (baud rate, data bits, parity, stop bits, flow control) in order to connect to serial device respectively to be able to correctly read received data. Because modules don´t support auto baud rate, this must be manually set. This can be done when device finds itself in AT mode by sending command AT*AMRS RS232 Settings with required parameters (see **Chyba! Nenalezen zdroj odkazů.**).

For example command **AT*AMRS=8,1,1,1,1,0,1** will set module to baud rate = 57600, number of data bits = 8, number of stop bits = 1, parity = no parity and flow control = CTS/RTS. The penultimate parameter must be always set to zero, because the function of this parameter is reserved for future upgrades. For the changes to take effect, the module must be restarted. The last “1” tells the module to store the settings after restart. Mentioned example is also default factory serial setting.



Serial Settings (AT*AMRS)

Baud Rate	57600
Data Bits	8
Parity	None
Stop Bits	1
Flow Ctrl	None

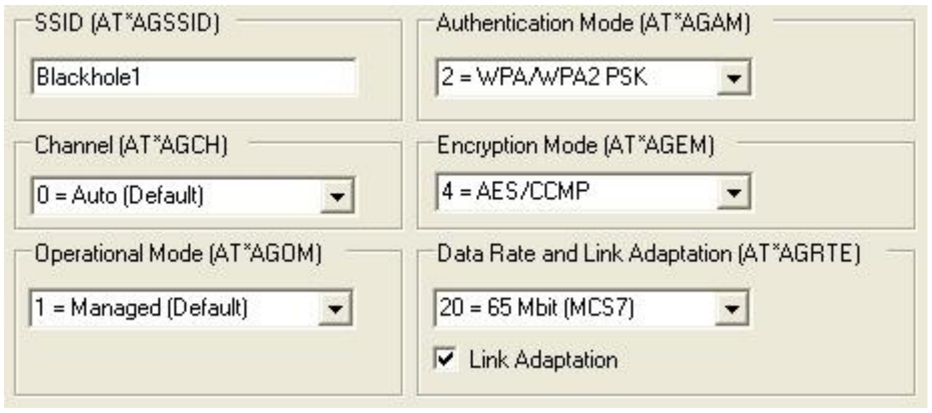
☐ Change After Confirm

Figure 40: SPA Toolbox: Serial settings

This command is similar to BT modification except of penultimate parameter - this offers to change serial settings immediately (parameter = 1) after the command is confirmed without any need to restart the module.

6.3.3 WLAN settings

SPA can operate in two basic modes: Infrastructure (called as “Managed”) and ad-hoc. The infrastructure mode uses an existing network and the modules connect to the access point (AC) which creates this network. By ad-hoc mode, modules create their own network and manage it themselves. If a module wanting to connect to an ad-hoc network cannot find an existing network with the predefined SSID (Service Set Identifier), it will create its own network and start broadcasting this. Any devices looking for the same network will find the network and can properly connect to it. As each device keeps the network up, devices can join and leave freely while the remaining devices keep managing the network. Each device in the network has its position in hierarchy and specific behavior.


SSID (AT*AGSSID): Blackhole1

Authentication Mode (AT*AGAM): 2 = WPA/WPA2 PSK

Channel (AT*AGCH): 0 = Auto (Default)

Encryption Mode (AT*AGEM): 4 = AES/CCMP

Operational Mode (AT*AGOM): 1 = Managed (Default)

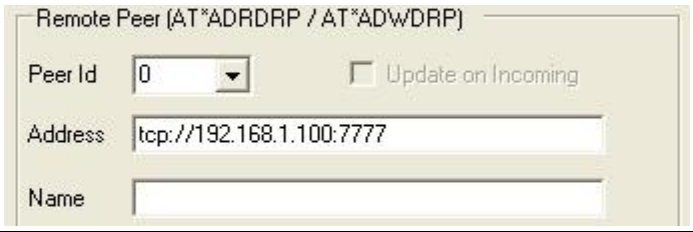
Data Rate and Link Adaptation (AT*AGRTE): 20 = 65 Mbit (MCS7)

[✓] Link Adaptation

Figure 41: SPA Toolbox: WLAN settings

Peers

Every sender/receiver is referred to as a peer. In the module case there are two types of peers: **local peer** = serial connection and **remote peer** = wireless connection, another device in the network. By specifying the remote peers in the SPA's settings, module can establish communication channel to this remote device. Remote peers are addressed using URL (Uniform Resource Locator) using similar string structure as internet server addressing e.g. http://www.connectblue.com. Compared to internet, in local networks instead of HTTP protocol, protocol TCP or UDP are used followed by IP address and port number. Then the example may look as follows: **tcp://192.168.1.100:7777**


Remote Peer (AT*ADRDRP / AT*ADWDRP)

Peer Id: 0

[ ] Update on Incoming

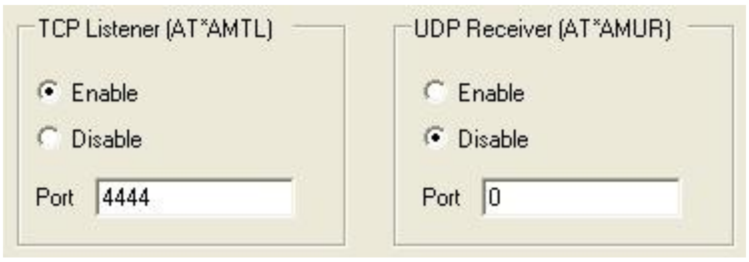
Address: tcp://192.168.1.100:7777

Name:

Figure 42: SPA Toolbox: IP setting

Listeners

SPA can be configured to act as a listener, listen and wait for incoming connection on a specified port. If the TCP listener detects incoming connection, it will negotiate TCP handshake and set up TCP connection.


TCP Listener (AT*AMTL)

[✓ Enable] Disable

Port 4444

UDP Receiver (AT*AMUR)

Enable [✓ Disable]

Port 0

Figure 43: SPA Toolbox: TCP/UDP Listener

✘ *Data transferring*

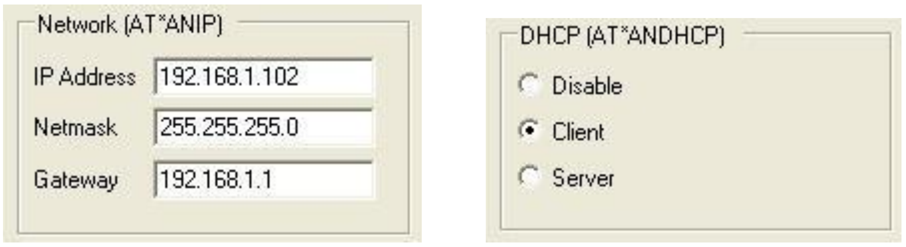
SPA can receive data in two different ways: data from serial interface (data intended to be sent to network) and data received from the network. Here it is necessary to mention one important fact. Data received by serial interface will be distributed to all remote peers which the modules connects to, while the data from remote peer(s) will be sent only to serial device (PC). There is no difference between data flow between TCP peer and TCP listener because the channel is full duplex. This means that once the connection is established (one module listens on specified port, second module creates a peer on this port) the communication will perform in both directions.

On the contrary, UDP connection (were not used in my application) does not create full duplex channel, which means that if you create UDP peer, module will not listen to this peer but will just send the data which has received from serial line. On the other hand module configured as UDP listener will only listen to incoming data which will pass to serial interface, but it will not send any data anywhere.

Problem occurs when the application requires information about from which module (peer) the answer comes from or when the module wants to send data only to certain peer without being received by the others. Regarding this troubles Bluetooth module has an advantage through the support of so called Extended Data Mode which is in WLAN variant not yet supported (promised in future firmware upgrades). When the BT modules communicate, “Master/Slave” system is used instead of peer/listener. Data from master (always only one) are sent to all slaves (up to 7 slaves) which are connected. On the other hand when the slave answers, the answer is received only by master. Communication between slaves is not possible. This structure is called Wireless Multidrop scheme. Extended data mode allows user to control individually each active link by sending special identifier along the data packets.

✘ *IP configuration*

Every device in the network is uniquely identified by its IP address. IP address (also netmask and gateway) of a module is stored in the module's IP stack and can be configured in two different ways: either via AT command AT*ANIP (SPA Toolbox) to the fixed value or by enabling DHCP server, which will automatically assign IP addresses to all participants (these are set to DHCP clients). DHCP server must also have IP address, which needs to be set manually. DHCP solution is a convenient way if the DHCP server acts as a listener, waiting for the others to connect. As soon as clients are connected to DHCP server, this assigns IP addresses to all connected clients. From the client's point of view, connecting to remote peer (DHCP server) is done using fixed IP address of the server. This must be known and configured for all clients. At maximum 7 clients may be connected to DHCP server.



Network (AT*ANIP)

IP Address 192.168.1.102

Netmask 255.255.255.0

Gateway 192.168.1.1

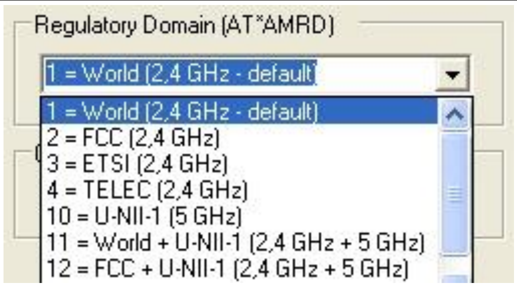
DHCP (AT*ANDHCP)

Disable [✓ Client] Server

Figure 44: SPA Toolbox: IP configuration

- ***Channel settings***

Module allows user to configure also frequency band (2,4 or 5 GHz or both) and appropriate channel. Frequency bands are preset according to destination's valid regulatory domain e.g. Channel can be chosen either automatically (selection 0 in drop-down menu) or manually. This gives user a tool, if he knows about any other network which uses the same channel, to change the channel and avoid interference or worse throughput. Known issue I that bands UNII 2 and UNII 2 extended are not allowed in ad-hoc mode.



Regulatory Domain (AT*AMRD)

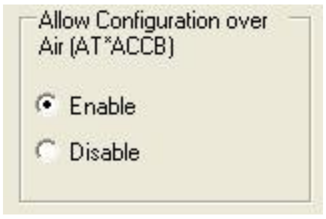
1 = World (2,4 GHz - default)

1 = World (2,4 GHz - default)
2 = FCC (2,4 GHz)
3 = ETSI (2,4 GHz)
4 = TELEC (2,4 GHz)
10 = U-NII-1 (5 GHz)
11 = World + U-NII-1 (2,4 GHz + 5 GHz)
12 = FCC + U-NII-1 (2,4 GHz + 5 GHz)

Figure 45: SPA Toolbox: Regulatory domain

6.3.4 Configuration over WLAN

By default, AT mode may be entered only via serial connection but using command AT*ACCB enables feature which makes it possible to enter module's AT mode remotely, over air. To make it possible, the communication must be already established, so that it is possible to send and receive any data between the modules and each module must have its own unique escape sequence. All the commands are sent from the PC running application and this is all the time connected to virtual COM port (to module with escape sequence "///", according to the Figure 36). When any of the connected devices receives command which matches their escape sequence, this module will switch to AT mode. Configuring a module over air is a big advantage because usually only one module is connected directly to PC while the others are mounted on peripheral devices (camera, remote controller) and collecting data. In some cases it is necessary to change settings (e.g. IP address, baud rate…) while the application on the PC is running and there it is no possibility to dismount module from the remote device and connect it to PC directly to reconfigure it. In such a structure it is necessary that each module has its own unique escape sequence.

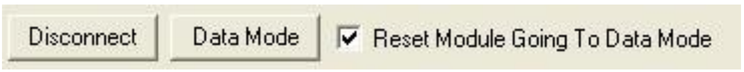

Allow Configuration over Air (AT*ACCB)

(● Enable) ○ Disable

Figure 46: SPA Toolbox: Configuration over air

6.3.5 Restarting module

As mentioned before, for the changes to take effect it is not enough just to switch back to data mode (AT*ADDM) but module must be reset. In order to reset the module, command AT*AMWS (see **Chyba! Nenalezen zdroj odkazů.**) is used or if the configuration is done via SPA Toolbox, field "Reset module going to data" mode must be checked. AT*AMWS command also returns module back to data mode as it is default mode after reset.



☑ Reset Module Going To Data Mode

Figure 47: SPA Toolbox: Resetting module

7 Wireless coupling PC ↔ Camera

With this chapter, description of my practical work begins. As described in the Motivation part and later on in Chapter 2.3, wireless connection of camera with the PC unit is one of the main steps in development of the new system. Currently camera communicates with computer over cable - RS-422 interface, which is afterwards converted by Serial USB converter into USB communication (apart from input signal, RS-422, the principle is identical with connectBlue UART to USB converter showed in Figure 35).

It was already explained that the cable is substituted by wireless modules, which uses their own communication protocol (802.11, Bluetooth). Therefore important from our point of view is only input (output) signal into one module and output (input) signal from the second one and make these compatible with camera input (output) signals. Interface on the computer side is procured by serial USB converter (Figure 35, Figure 48).

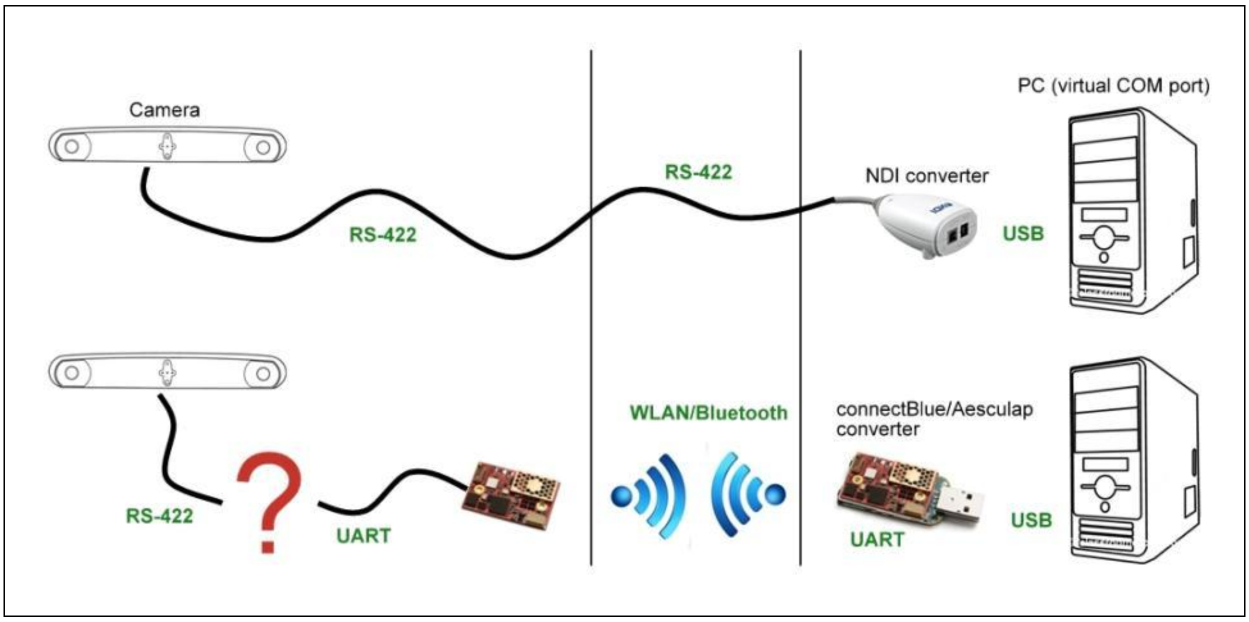


Figure 48: Camera-PC interface. Top row presents current state, in bottom can be seen intended configuration. Signal types are denoted by green color. Red question mark stands for developed device-subject of this chapter and of my practical work

Wireless module ↔ Camera interface development was based on the work of the previous student, who made the first prototype as shown on the picture in Appendix E. Main disadvantages of use of such a device are at first its dimensions and the fact that the cables were not removed at all, because camera must be bound with that "blue box" via wire. Further the motivation to build a new prototype was an attempt to increase the speed of transmission.

On the test application, the difference of FPS (frames per second[21]) between cable: wireless solution was in ratio 30: 11 FPS. [13]

Development of a new prototype device (PCB) *Camera-Wireless rev1.1* consisted of several steps. At the beginning it was necessary to define general requirements:

7.1 General requirements

- Device is connected to camera directly over embedded LEMO male connector
- Device dimensions including housing max. 60 x 35 x 20 mm
- Device converts RS-422 signal into UART levels
- Device is power supplied with 18-32V DC source and passes voltage to camera
- Step down voltage converter converts input voltage to 3,3V for nets with IC components
- Device supports both Bluetooth and WLAN connectBlue modules
- Module is equipped with internal antenna
- Module is removable (connection to PCB via Samtec connector)
- Module’s max dimensions 36 x 23 x 4,1 mm
- Housing is made of RF visible material
- Device informs about its current status by means of blue LED (Light Emitting Diode)
- Device will not be used in sterile environment
- Support of CTS/RTS flow control

7.2 Software environment – Altium Designer

On the market there are many different software products offering PC-based electronics design (PCB development) such as *EAGLE* (Easily Applicable Graphical Layout Editor), *ZenitPCB, FreePCB* or *Altium Designer* etc. The last mentioned has been used for projecting new PCB. Here comes a brief description of *Altium Designer Rev. 10* (Altium Limited global headquarters, Sydney, Australia):

- ***Schematic editor***

Together with PCB editor these are basics of every electronics designer. Schematic editor allows designing nets and connections using components from libraries, adding notes, labels etc. Every component has its own schematic and footprint in PCB editor. Single

[21] In our case FPS = number of request-answer cycles - how many times per second the camera is requested for data and replies (transformation matrix) for command

components can be either found in the existing libraries (in Altium folder or downloaded from the Internet) or must be designed by the user.

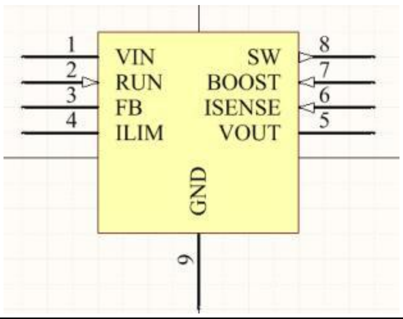


Figure 49: Component's schematic example (LT3663 Voltage converter)

PCB editor

When opening new PCB document, all components (their footprints) from schematics are automatically added into the working space and can be further processed. User can set shape of board, cutouts, holes for screws, number of layers, thick of dielectric layer, width of lines, auto-tracking options etc. To control the correctness of wiring, user may define “PCB rules”, which check automatically the distances between specified objects/lines/layers, overlaying objects, minimal line width etc. according to the value specified in the rule and notice user about rule violations in Design Rule Verification Report.

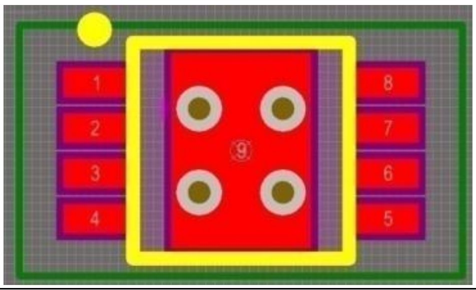


Figure 50: PCB component footprint example (LT3663 Voltage converter)

In the example above (Figure 49) you can see the footprint of the step-down converter. Red layer for instance stands for silver pads, which the component will be soldered to; gray-brown circles in the middle are the holes through the board to overtake the line from the top to bottom side in order to avoid shortcuts. The yellow line is top overlay layer, which will be printed (in color) on the board to denote the name, shape and orientation (dot on the component must match dot on the board) for soldering the component.

When designing own components, 3D body can be added (in existing libraries the 3D model usually already exists) to view the future board appearance in 3D model.

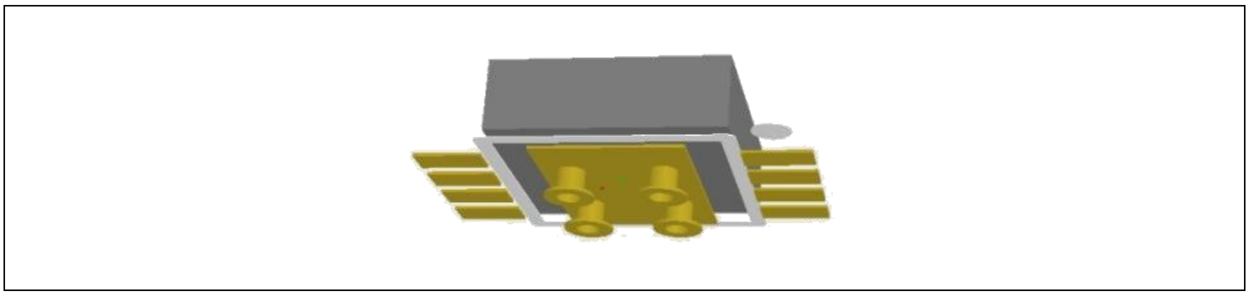


Figure 51: Component's 3D model (viewed from bottom)

7.3 Schematic development

First it was necessary to select appropriate electronic components and design schematic plan. The whole schematic can be divided into two bigger circuits:

7.3.1 Transceiver circuit

✘ ***Transceiver***

Crucial element in the whole device is RS-422 Transceiver. This is an IC which combines both transmitter and receiver in one body. Transmitter part (in IC called Driver) receives data in CMOS/TTL logic levels (Chapter 3.1.4) and converts this signal into two differential signals: A/- inverted and B/+ non-inverted as described in Chapter 3.1.3. On the side of receiver the differential signals are subtracted and the result is sent on the receiver's output, again in TTL/CMOS logic levels. Typical operating circuit may be seen in Figure 52 with pin explanation in Table 4. [50]

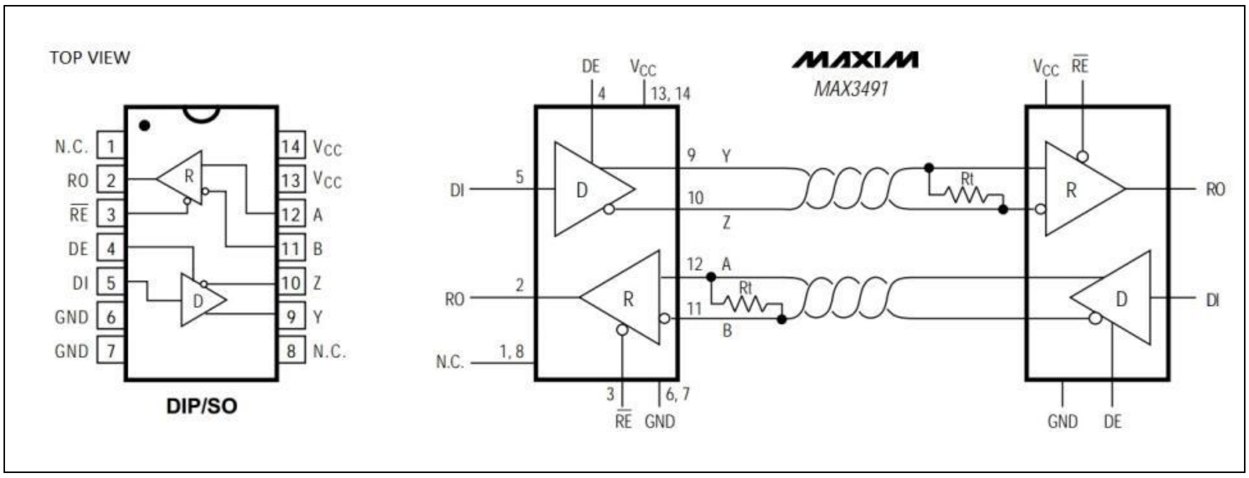


Figure 52: MAX 3491, typical circuit [50]

Pin number	Name	Function
1,8	N.C.	Not connected
2	RO	Receiver Output. If A > B by 200mV, RO will be high; if A < B by 200mV, RO will be low.
3	RE	Receiver Output Enable. RO is enabled when RE is low
4	DE	Driver Output Enable. The driver outputs are enabled when DE is high
5	DI	Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low.
6,7	GND	Ground
9	Y	Non-inverting Driver Output
10	Z	Inverting Driver Output
11	B	Inverting Receiver Input
12	A	Non-inverting Receiver Input
13,14	VCC	Positive Supply: 3,0V ≤ VCC ≤ 3,6V

Table 4: MAX 3491, pins functions [50]

For our application *MAX3491ESD* (Maxim Integrated Products, Sunnyvale, USA) transceivers were used. Compared to transceiver used in the first prototype, *MAX3491ESD* offers four times higher speeds - up to 10 Mbit/s. [50] . Camera uses 4 differential pairs of lines, 2 input pairs (RxD, CTS) and 2 output pairs (TxD, RTS). Because each transceiver contains one driver and one receiver, two transceivers are needed. MAX3491ESD supports and produces the same voltage ranges for logic states (CMOS 3,3V/TTL) as wireless module which means that no additional voltage level converters are needed and the pins 2 (RxD, CTS) and 5 (TxD, RTS) can be directly connected to pins of the wireless module.

✘ *Terminating resistor*

As can be seen in the Figure 52, on the receiver’s side there are resistors between differential lines. These, so called terminating resistors prevent signal of being reflected back after the signal hits big receiver’s input resistance. Rt = 120 Ω is the commonly used RS422/485 termination value. In our developed schematic, terminating resistors are denoted R5, R6 (see cutout in the figure below or Appendix F for complete schematic).

✘ *Pull-Down resistor*

It is also useful to use pull-down resistors (R1, R2, R3, R4) to pull the signal levels towards ground (GND) voltage when the line is (would be) in inactive (undefined) state or broken. This is a safety solution, which assures that the incoming data will not be misunderstood. The value of pull up/down resistors is commonly about 10 kΩ. All of these resistors are placed only at the side of receivers. On the other two pairs of lines, resistors are (probably) placed in the camera's electronics.

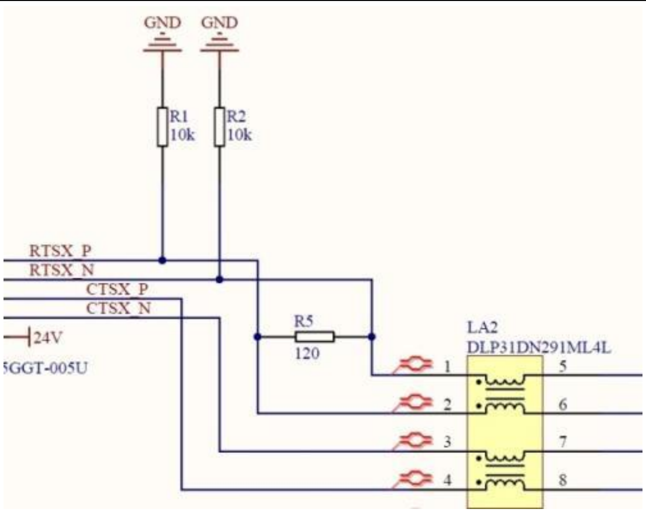


Figure 53: Schematic cutout: Terminating resistor (R5), Pull-down resistors (R1, R2), Common mode choke coil (LA2)

✘ *Common mode choke coil*

Blocks LA1 and LA2 are so called common mode choke coils[22] and serve as prevention against electrical noise caused by EMI. Twisted differential lines have their currents in different phase which leads to the fact that the sum of their magnetic fields is equal to 0. This results in zero impedance in the coil and passing differential signals through. On the other hand common-mode currents sum up their magnetic fields resulting in high impedance and attenuation of common-mode signals. [51]

✘ *Bypass capacitor*

Additionally pins VCC (input voltage) and GND of each transceiver are connected over bypass capacitor (C4, C5), which serves as a low-pass filter. Removing high frequencies from the input voltage makes IC's outputs smoother, suppressing edge spikes and making signal symmetrical at high-to-low and low-to-high transitions. Generally it is recommended to include one bypass capacitor for each IC, while the value is usually between 10-100 nF. [52]

[22] In Czech translation „tlumivka"

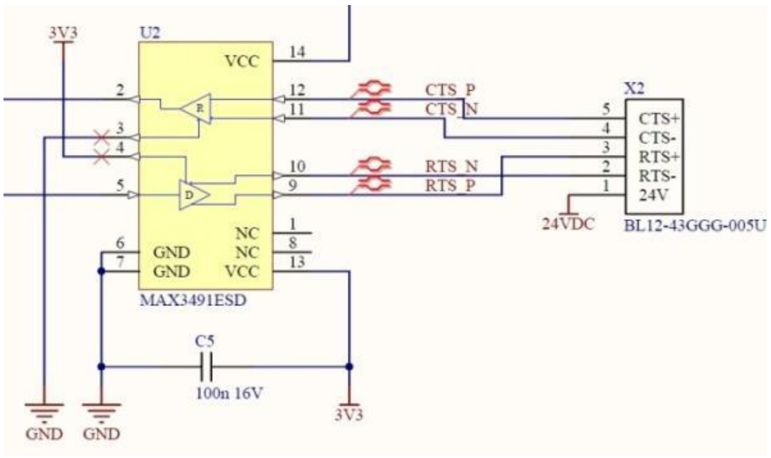


Figure 54: Schematic cutout: MAX 3491 wiring; bypass capacitor C5

Example of the logic output with and without bypass capacitor can be seen in the figure below.

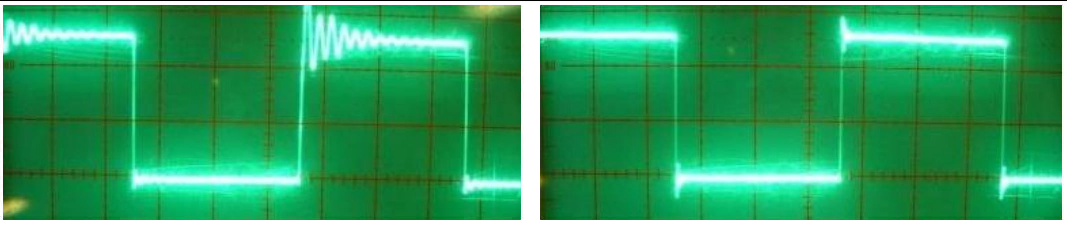


Figure 55: Bypass capacitor effect: circuit currents without (left)/with (right) mounted bypass cap. [52]

Lemo connector

As was already mentioned, camera is equipped with female connector on the rear side of the camera body. Pins and the signal which they carry are described in table below.

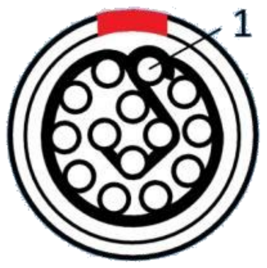
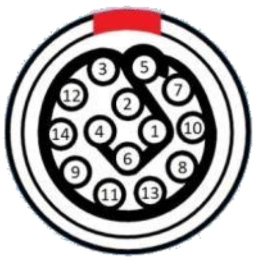
	Pin	Signal	Pin	Signal
	1	Power	8	TxD+
	2	Power	9	TxD-
	3	RxD+	10	Power
	4	RxD-	11	RTS+
	5	GND	12	RTS-
	6	GND	13	CTS+
	7	GND	14	CTS-

Table 5: Camera side connector: pin description (numbering along the curve, viewed from the rear of connector) [53]

To fulfill the requirement, compatible Lemo *FAA.1B.314.CLL* 14-pin connector, which is soldered directly to PCB, was chosen. In the schematics (and layout), numbering does not match the one in the table above and may be little confusing. See the difference in the table below.



Pin	Signal	Pin	Signal
3	Power	14	TxD+
5	Power	12	TxD-
10	RxD+	7	Power
8	RxD-	2	RTS+
9	GND	1	RTS-
11	GND	6	CTS+
13	GND	4	CTS-

Table 6: PCB soldered connector: pin description (viewed from the top of connector)

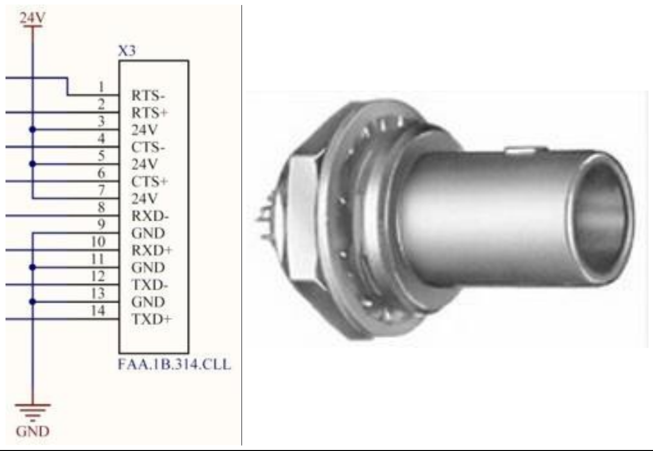


Figure 56: Lemo FAA.1B.314.CLL: connector wiring (left), connector (right)

Female/Male header connector

Because the female connector in the camera is placed very deeply below the surface of camera housing and the insufficient length of the connector pins wouldn't allow direct soldering to PCB, another "mediator" PCB must have been designed. This is connected to main PCB by female/male single row connectors placed on both PCBs (will be described closer in Chapter 7.4). It is necessary to realize that camera connector cannot be dismounted from the housing due to the fact that the housing wall is positioned between connector's outer extended ring and washer and nut on the other side followed by firmly soldered PCB. Therefore detachable connector part additionally allows making changes or repairing main PCB if needed independently on connector's PCB.

In the schematic, the header row connectors are denoted X1, X2 and consist in total of 10 pins (4 pairs of differential lines + GND + VCC). This connection is also the place where the differential lines changes their relative labels e.g. TxD output lines of module change their labels to RxD input lines of the camera. Analogically RxD → TxD, CTS → RTS, RTS → CTS. In the sense of inverted and non-inverted signals, the lines don't change.

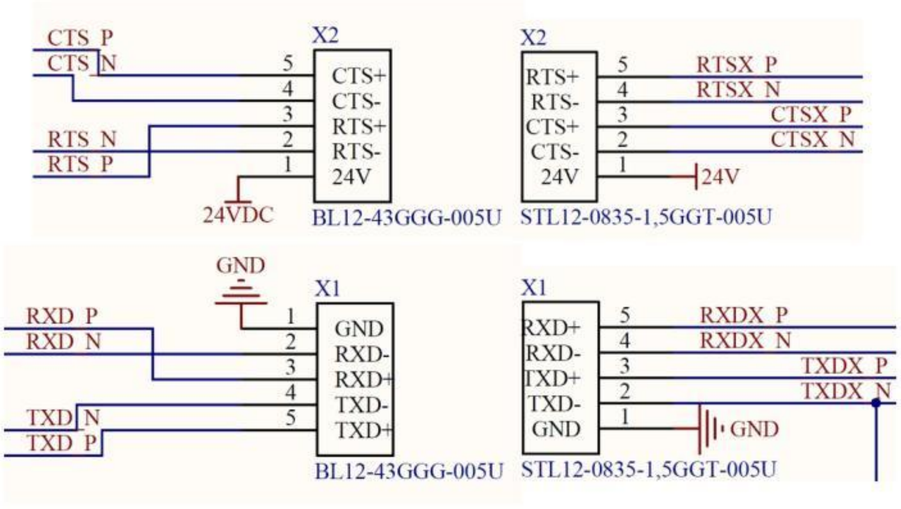


Figure 57: Schematic cutout: Row header connectors – connection between PCBs.

Samtec double row connector

Connector provides connection between PCB and wireless module. From total 40 pins organized in two rows (J2 (1-20), J3 (1, 20)) only 11 are used: TxD, RxD, CTS, RTS, 2 x 3,3V, 4 x GND, Blue LED status signal (see figure below).

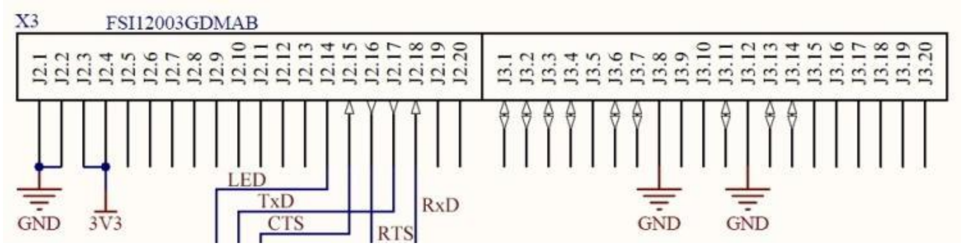


Figure 58: Schematic cutout: Samtec double row connector wiring

Blue LED

Light Emitting Diode informs about the state of module (ON when connected, BLINKING on data transmission). Controlling signal comes from the module and in active state is low (max. 0,4V), which causes voltage drop over the diode and makes it light. Resistor value (R) was calculated from formula:

$$R = \frac{V_{supply} - V_{forward}}{I_{forward}},$$

Equation 8: LED resistor value calculation where V_{supply} is the voltage in the net (3,3V), $V_{forward}$ and $I_{forward}$ are LED specific values specified in LED data sheet (2,85V and 5mA). Minimal calculated resistor value is R = 100Ω, so finally slightly higher value 120 Ω was chosen.

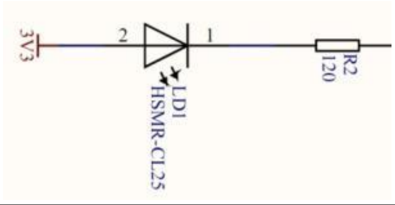


Figure 59: Schematic cutout: LED connection

7.3.2 Voltage converter circuit

Second subpart of the whole circuit consist of step-down voltage converter, because the voltage of 18-32V (needed for camera) is too high for use in IC circuits, which were described above. Converter Linear Technology *LT3663EMS8E-3.3#PBF* (Linear Technology Corporate Headquarters, Milpitas, California, US) was chosen for ability to support wide range of input voltages (7,5 – 36V), fixed output voltage (3,3V), programmable output current (0,4 – 1,2A) and last but not least its small dimensions compared to concurrency products. Pins' wiring was done according to typical application example in component's data sheet [54]. To connect the cable, power supply chassis socket Lumberg *1612 14* (Lumberg Connect GmbH, Schalkmühle, Germany) was used.

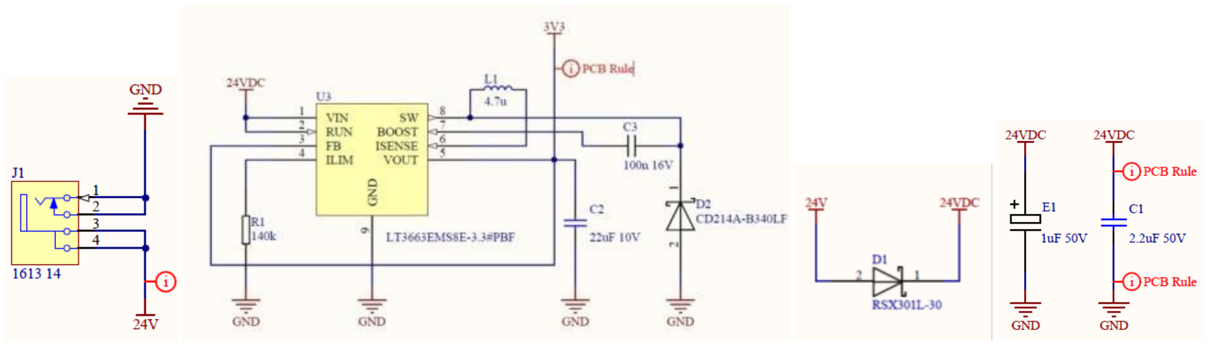


Figure 60: Schematic cutout: Power supply circuitry

Before approaching to PCB development the whole circuit was built on a "bread board" to test correctness of wiring and proper functionality.

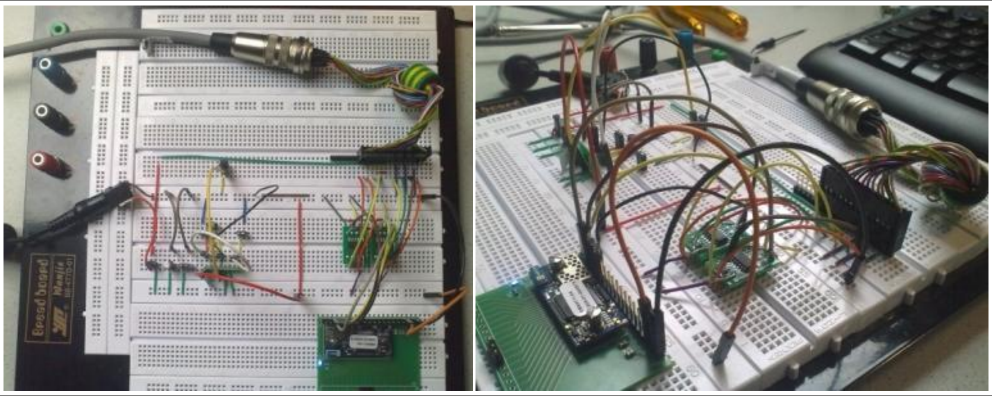


Figure 61: Circuit on a testing Bread board

7.4 PCB layout development

In PCB development it was necessary firstly to design the shape and decide how many layers the board will have. As was mentioned earlier, device consists of two PCB which were made as two separated projects:

- ***Camera_Wireless_1.1***

This bigger PCB carries most of the element and will be mounted into the housing. Therefore drills (holes) in mechanical layer for screws as well as holes for mounting Samtec connector and Lumberg power chassis socket cannot be forgotten. Housing itself was not included in my work as it will be designed after the PCB is made and all components are soldered by a colleague in ENT department. PCB dimensions are 55,5 x 27 x 0,8 mm and it contains two layers, while the bottom layer is made as a GND plane –"polygon", which means that the whole bottom layer has the common 0V voltage and other nets are cutout into this polygon. This is an advantage when lot of components must be connected to GND net. Then it is enough just to place "vias" at any position (holes connecting the layers) to connect to GND. The difference between top and bottom layer is shown below. Complete PCBs together with their 3D visualization can be found in Appendix H and Appendix I.

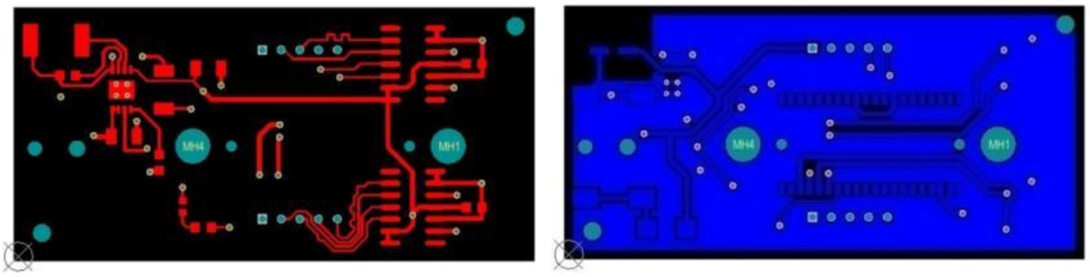


Figure 62: Top layer (left); Bottom layer polygon (right); Mechanical layer drills (dark cyan color)

- ***Camera_Wireless_1.1_small***

Smaller board as can be seen in Appendix H has dimensions 22 x 14 x 0,8 mm and will be placed on the top of the one described above, connected over 2 male/female header 5-pin connectors. This board will be firmly fixed together with Lemo connector to the housing. For imagination of the whole device and placement of one board to another, 3D model is shown in the figure below, with dimensions included.

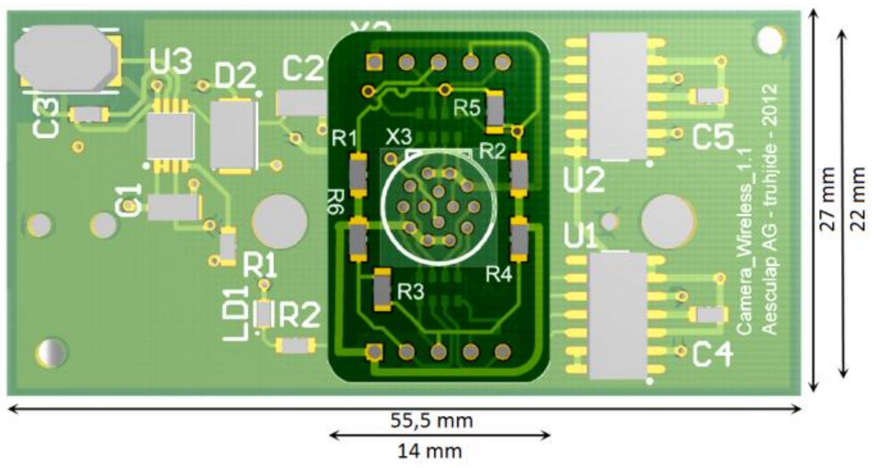


Figure 63: Position of smaller PCB with respect to main PCB (viewed from top) & dimensions

7.5 Software development

Changing interface between PC and camera required not only hardware changes but also software modifications. For communication with the camera, Aesculap developers coded classes for receiving and sending data from COM port (respectively VCP: virtual COM port). All source codes are written in C++ programming language.

7.5.1 Camera API commands

PC communicates with camera using specific commands which are described in detail in camera's guides. For instance, here are the basic two commands used further in my source code.

- `RESET<SPACE><CR>`
  replies: `RESETBE6F, ERROR`

Command can be sent anytime to reset camera. After each reset camera starts-up with default settings: baud rate 9600, 8 data bit, no parity, 1 stop bit, no handshake (CTS, RTS)

- `COMM<SPACE>50001<CR>`
  replies: `OKAY896, ERROR`

Command changes communication settings according to the following pattern:

Baud Rate	Data Bits	Parity	Stop Bits	Handshake
0 (9600)	0 (8 bits)	0 (none)	0 (1 bit)	0 (off)
1 (14400)	1 (7 bits)	1 (odd)	1 (2 bit)	1 (on)
2 (19200)		2 (even)		
3 (38400)				
4 (57600)				
5 (115200)				

Table 7: Camera communication parameters

Other commands are included in extensive source codes of OrthoPilot application (which has been developed for many years by professionals) but for our purposes are not important. Those commands for instance start tracking mode, load .rom file (see Chapter 2.1.2), obtain tool information, request transformation matrix etc.

7.5.2 Camera initialization

Commands described above are important because by changing the hardware interface, application wouldn't work with the previous codes anymore and some small changes had to be made. To understand the source code, which will be introduced later, let's explain the differences with help of application diagrams:

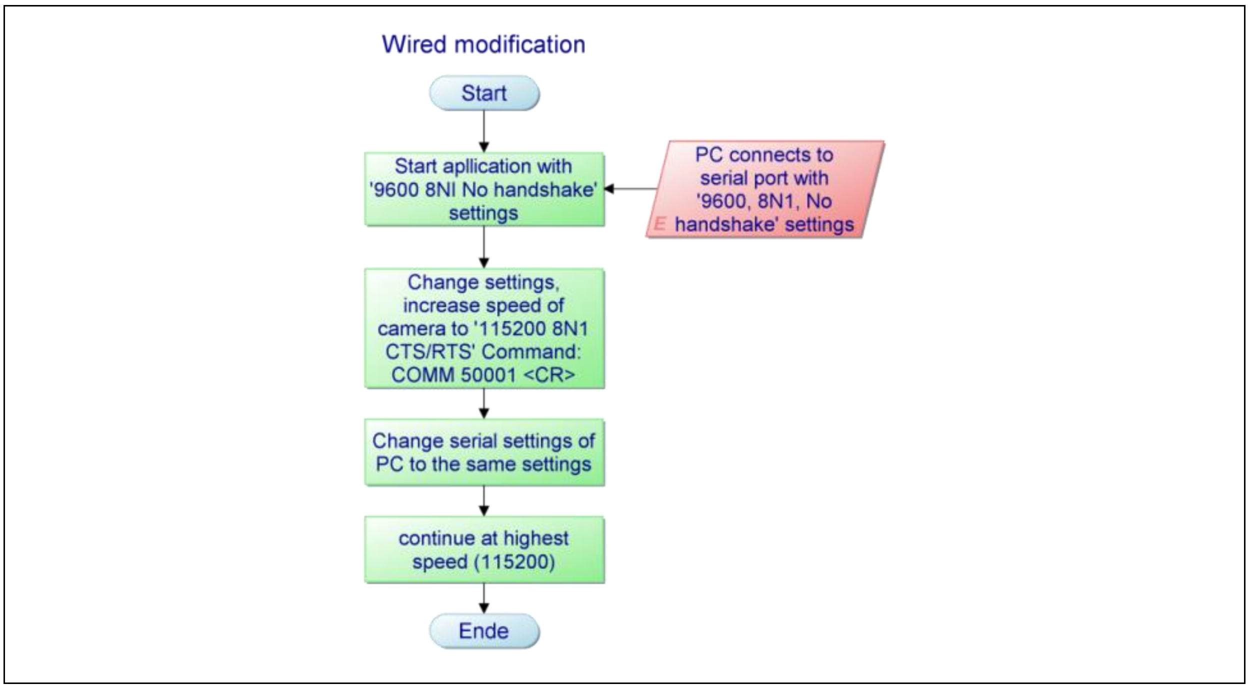

Wired modification
Start
Start apllication with '9600 8NI No handshake' settings
PC connects to serial port with '9600, 8N1, No handshake' settings
Change settings, increase speed of camera to '115200 8N1 CTS/RTS' Command: COMM 50001 <CR>
Change serial settings of PC to the same settings
continue at highest speed (115200)
Ende

Figure 64: Camera initialization cable modification[23]

[23] Abbreviation "9600 8N1" stands for: baud rate 9600, 8 data bits, No parity, 1 stop bit.

Communication parameters cannot be hard-coded because after every reset the camera starts-up with default settings and as they are not sufficient in terms of speed[24] these need to be changed to higher baud rates (max. 115200).

In the cable variant, it is necessary just to initialize communication at low speed (9600) and then send camera command `COMM 50001` in order to increase the speed to maximal possible speed 115200 baud. At the end PC resets its port setting (9600→115200) to synchronize with camera.

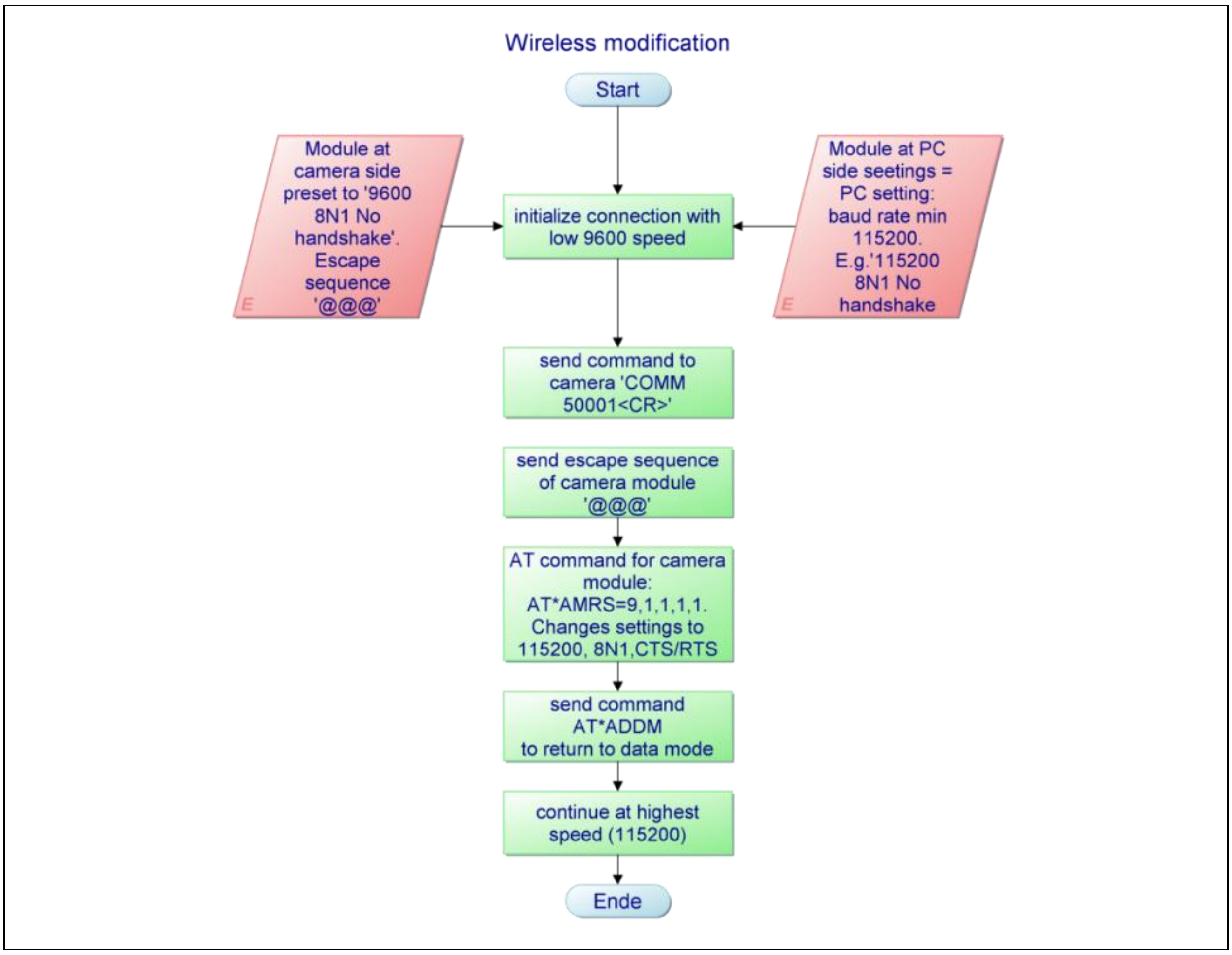

Wireless modification
Start
Module at camera side preset to '9600 8N1 No handshake'. Escape sequence '@@@'
initialize connection with low 9600 speed
Module at PC side seetings = PC setting: baud rate min 115200. E.g.'115200 8N1 No handshake
send command to camera 'COMM 50001<CR>'
send escape sequence of camera module '@@@'
AT command for camera module: AT*AMRS=9,1,1,1,1. Changes settings to 115200, 8N1,CTS/RTS
send command AT*ADDM to return to data mode
continue at highest speed (115200)
Ende

Figure 65: Camera initialization wireless module modification

In comparison wireless modification requires synchronization of communication parameters on two interfaces:

[24] Baud rate 9600 (9,6 kbit ≈ 1,2 kByte) would theoretically allow to repeat "request-answer" cycle only about 6 times per second: **request data size + answer data size ≈ 200 Byte ≈ 0,2 kByte**; **1,2kByte/0,2kByte = 6 cycles/s (fps).** At baud rate 115200 the theoretical value is about ten times higher (≈ 60 fps).

- ***PC ↔ PC wireless module***

Interface setting between PC and its module stays constant during the whole application process and can be set independently of second interface between module and camera. In order not to slow down the transmission, the speed of this interface should be at least of the value between camera and module, which means 115200 baud (maximal speed of camera) or more. The PC port must have the same settings (are set in the application using Aesculap, self-developed serial port classes).

- ***Camera ↔ camera wireless module***

Parameters' setting on the camera side is more complicated and configuration over air (Chapter 6.3.4) must be used. Initially the camera module is in DATA mode with settings (9600, 8N1, no handshake) which match settings of camera. In order to increase interface's speed, firstly the command for camera must be sent (`COMM 50001`) followed by escape sequence specific for this module sent from the PC to enter AT mode a reconfigure serial settings. After the changes take effect, module in DATA mode can communicate with camera again.

7.5.3 Software coding for use with WLAN module

The work has been based on code of another co-worker [13], who created the function for substituting cable with BT module. Although BT and WLAN modules look to work in the same way, slight differences didn't allow WLAN modules to be used with existing BT function. Therefore small changes/adding's to the code must have been done to make WLAN module work with the rest of the code. As mentioned, application is coded in C++ language and for programming Microsoft Visual Studio 2008 software was used. The complete code can be found on the CD which accompanies the written thesis. Original [13] and newly added code lines are differentiated by colors.

When OrthoPilot application starts, configuration data are loaded from the certain file. Here the information about which COM port should be used is stored. Because the application must differentiate whether the COM port uses serial cable or wireless module, it was set in, that cable connection will always be COM 1 and COM 3 stand for wireless module.

```
bool CCommunicationInterfaceRS232::CheckBluetooth()
{
	if (strPortName =="COM3" )
		return true;
	else
		return false;
}
```

If the condition below is false, the program will skip following function. If the condition is true, application will set up firmly COM 3 communication parameters (PC-module interface).

```
else if (bBluetooth)
            dcbDevConBlock.BaudRate = 921600; //set virtual BT comm port  speed
```

In the next step function , application enters AT mode (of a remote module) and distinguishes whether the wireless module is BT or WLAN by sending command, which is valid only for WLAN module. BT module would eventually reply with an error message.

```
BT_ATMODE_ON="+++"; //escape sequence of camera BT/WLAN module
Sleep(1100);                     //interval of relaxed state on line
WriteToPort(BT_ATMODE_ON);
Sleep(1100);                     //interval of relaxed state on line

WriteToPort("AT*ANDHCP?\r"); //command which exists only for WLAN modules
ReadLineFromPort(GetTechnology_1,10);     //if WLAN-answer: AT*ANDHCP?
ReadLineFromPort(GetTechnology_2,10);    //if WLAN-answer: AT*ANDHCP=0
ReadLineFromPort(GetTechnology_3,10);    //if WLAN-answer: OK

if ((GetTechnology_2.find("ANDHCP")!= -1 ))
//if no Error message is read, module will use WLAN commands
{
      WiFi = true;
      BT = false;
      ret = true;
}
else if ((GetTechnology_2.find("ERROR")!= -1) )      //otherwise BT commands
{
      WiFi = false;
      BT = true;
      ret = true;
}
else   //there is something wrong with communication, no reply for ANDHCP command
      ret=false;
```

Afterwards application sends command to get informed about serial settings of a remote module

```
WriteToPort("AT*AMRS?\r");    //read module's settings
```

and if the parameters match those specified in the variable storing baud rate (during the application this changes 9600 → 115200, eventually different values changed by another part of the program) it will keep them, otherwise the parameters will be changed as required. Here comes the core of the problem, as the commands for changing the parameters differ for BT and WLAN and the correct one need to be sent (see Appendix D- AT*AMRS command).

```
if (WiFi == true)
                  BTSpeedChange="AT*AMRS="+ParamBT+','+"0,1\r";
if (BT == true)
                  BTSpeedChange="AT*AMRS="+ParamBT+','+"1,0\r";
WriteToPort(BTSpeedChange);
Sleep(100);
```

Because BT module does not need to be restarted, but it is enough just to return to DATA mode, application sends following command

```
BT_ATMODE_OFF="AT*ADDM\r";
//Just going back to data mode-settings are already saved by AT*AMRS
command
```

For WLAN modules, different command must be sent in order to make changes take effect.

```
BT_ATMODE_OFF="AT*AMWS=0,0,0,0,1,0\r";
//Resets module going to data mode-WLAN only way how to save the settings
```

Application works in an endless loop, which means that once it gets into this loop it performs given tasks constantly. Function `bool CCommunicationInterfaceRS232::SetBluetoothPortParameters()` containing all the commands mentioned above, is being called every loop cycle and if the parameters don´t match the required ones, they are changed. This happens at the beginning and during the application progress if camera is accidentally disconnected (loss of power supply etc.).

8 Touch panel remote controller

Last part of my thesis deals with development and programming of a remote controller (RC) prototype, which might be in the future used in a new OrthoPilot system. Based on discussions with my supervisor, general requirements were defined as follows.

8.1 General requirements

- RC communicates with PC over wireless WLAN/BT module in form of commands in hexadecimal (HEX) codes
- RC consists of small display (with/without touch panel), external buttons, vibration motor and sound buzzer
- power supply is procured by rechargeable accumulator
- RC is compatible with footswitch functions in the main application, which means that RC sends the same commands
- RC receives commands from PC about markers' visibility; information is displayed on the display
- Display informs surgeon/assistant about instrument to be used in the next step so that he can prepare the instrument in advance
- Vibration and sound alerts are used to inform surgeon about button press or invisibility of marker.

RC output commands	Event
0x40	right button pressed (only)
0x02	left button pressed (only)
0x01	both buttons pressed (= middle button in footswitch)
0x00	no button pressed

RC input commands	Event
0xA1	femur RB green (= tool visible)
0xA2	femur RB yellow (= tool on the border of visibility)
0xA3	femur RB red (= tool not visible)
0xB1	tibia RB green
0xB2	tibia RB yellow
0xB3	tibia RB red
0xAA	pointer RB green
0xBB	pointer RB yellow
0xDD	pointer RB red
0x10	next instrument
0x20	previous instrument

Table 8: RC I/O commands definition

8.2 Hardware

RC prototype is based on development board *Mikromedia for XMEGA* produced by company MikroElektronika (MikroElektronika, Belgrade, Serbia), which provides rich development platform for devices based on ATXMEGA128A1 microcontroller.

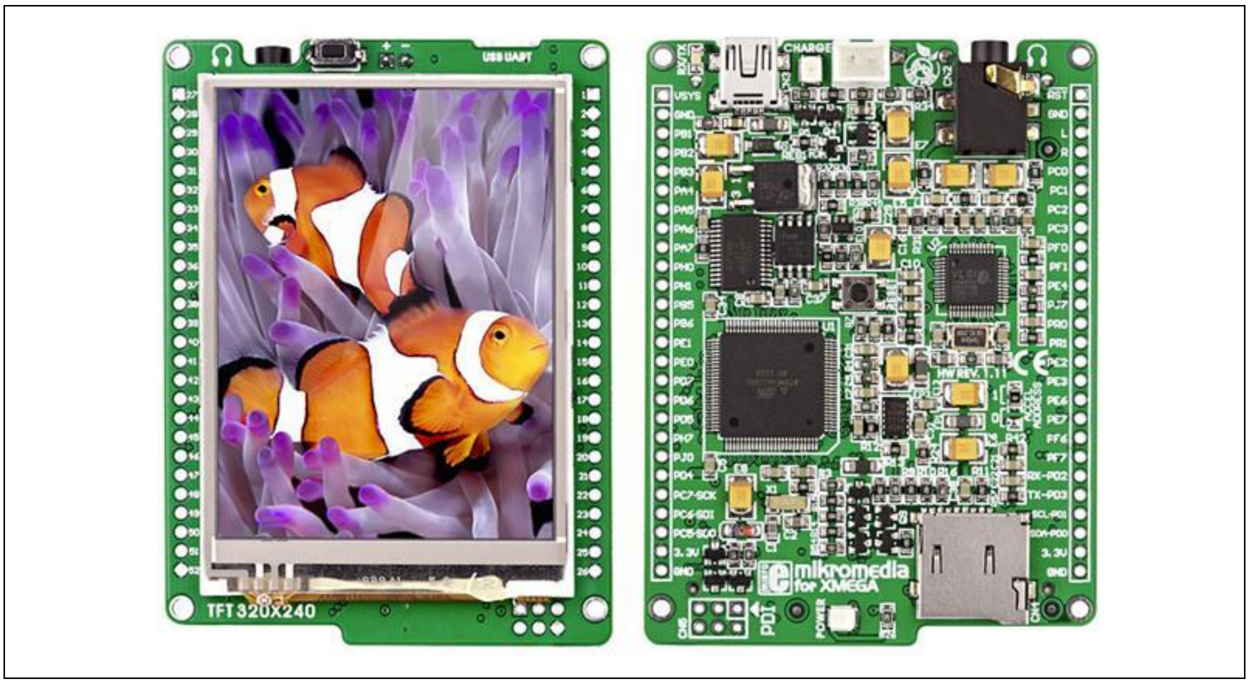


Figure 66: MikroElektronika: Mikromedia for XMEGA [55]

Board itself already contains many electronic components providing many useful features such as:

- ***ATXMEGA128A1, 8-bit microcontroller*** from Atmel® offers good base for wide range of applications; contains 78 multi-purpose I/O[25] pins; features low power consumption
- ***TFT display 320x240 pixels with 262.144 colors***
- ***Touch panel***
- ***On-board battery charger***, which operates over USB connection; Charging current is ~250 mA and charging voltage 4,2V DC.
- ***microSD card slot*** allows user to store large amounts of data externally, thus saving microcontroller memory.
- ***Voltage converter 5V→ 3,3V***
- ***Board manual*** contains all schematics and wiring of on-board components, allowing user to use the same circuits in self-developed final product

[25] Input/output

In my application, in total only 9 pins are used: 1 pin (input) for each button, 1 pin (output) for driving vibration motor, 1 pin (output) carrying signal to sound buzzer, 3x reference GND, 3,3V output for driving the vibration motor and buzzer and 5V pin to supply wireless module. Schematic, as can be seen below, is divided into 4 main circuits:

- ***Buttons (S1, S2) circuit*** (PB1, PB2, 3.3V board pins)
- ***Sound module (G2) circuit*** (PF1, 3.3V board pin)
- ***Vibration motor (M1) circuit*** (PC2, 3.3V board pin)
- ***Wireless module (X3) + blue LED circuit*** (PD2, PD3, 5V board pins)

Rows of pins X1 and X2 are female headers on XMEGA board. Internal board schematic is here not presented, but can be found in [55].

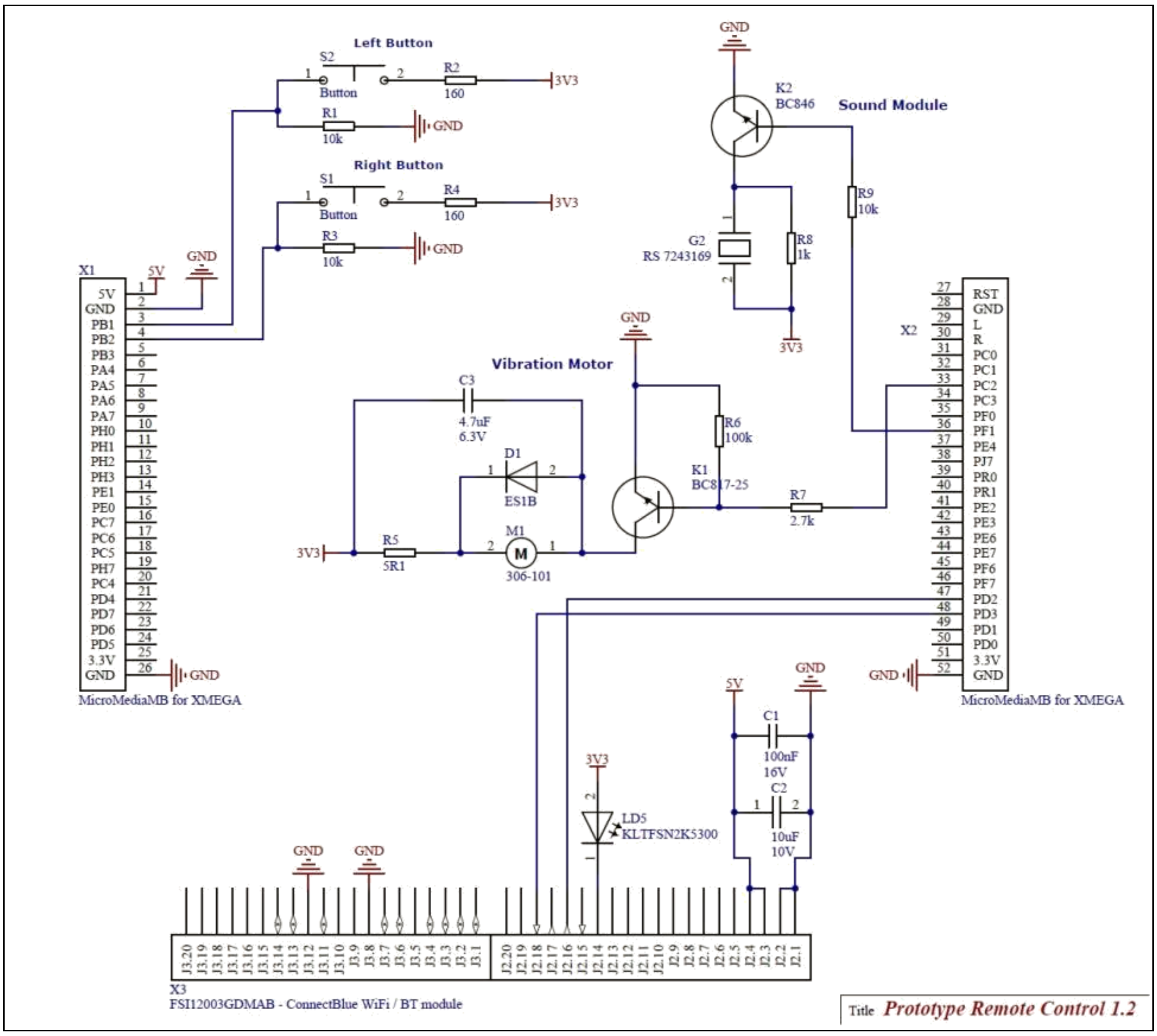


Figure 67: Remote Controller schematic

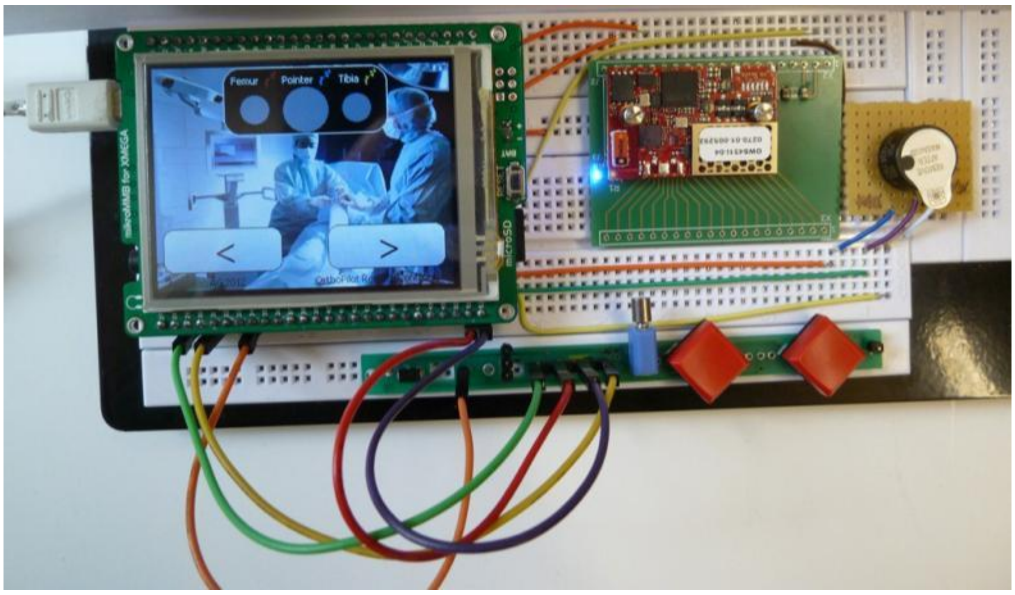
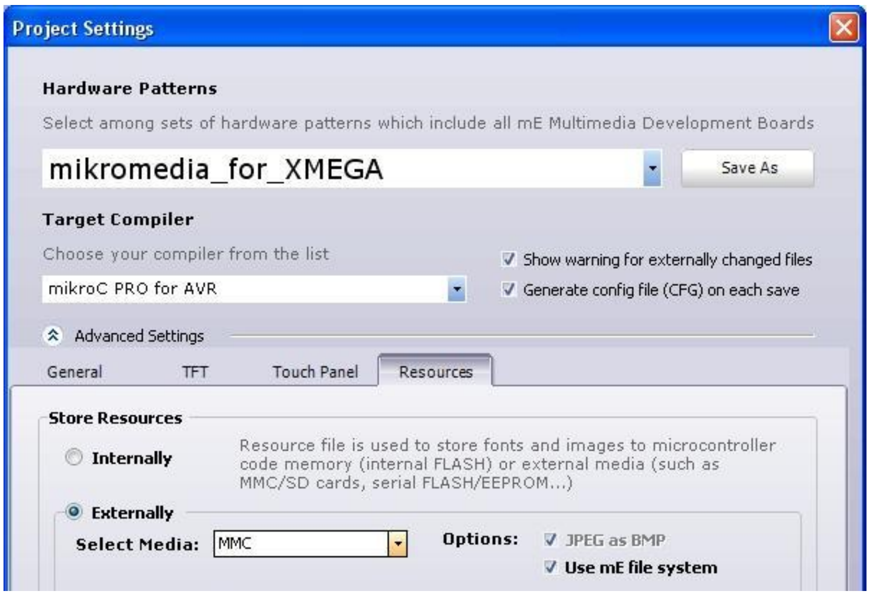


Figure 68: Remote Controller prototype

8.3 Software: microcontroller programming

MikroElektronika provides along their hardware solutions wide range of software for programming the microcontroller, which already contains functions for supported features of the microcontroller. Programming was done using C programming language in *microC PRO for AVR* compiler. Similar to MATLAB GUI (Graphical User Interface) editor, mikroElektronika also offers separated program *Visual TFT* for intuitive graphical development of the screen’s layout. When the desired design is ready, this software automatically generates source code and exports it to *microC* compiler where the rest of coding is done. Thus when *mikroMedia for XMEGA* board is chosen from the product list, V*isual TFT* editor also automatically generates codes which configures CPU, communication with TFT display and touch panel, this means setting correct pins and their direction according to the wiring, which is predefined in case of mikroElektronika development boards. Because the flash memory has only limited capacity and my application uses big amount of pictures, these were stored externally on the microSD card (codes which configure communication with microSD card slot and read the data are generated automatically according to selected product.


Project Settings

Hardware Patterns

Select among sets of hardware patterns which include all mE Multimedia Development Boards

mikromedia_for_XMEGA

Target Compiler

Choose your compiler from the list

mikroC PRO for AVR

[✓] Show warning for externally changed files

[✓] Generate config file (CFG) on each save

Store Resources

Internally — Resource file is used to store fonts and images to microcontroller code memory (internal FLASH) or external media (such as MMC/SD cards, serial FLASH/EEPROM...)

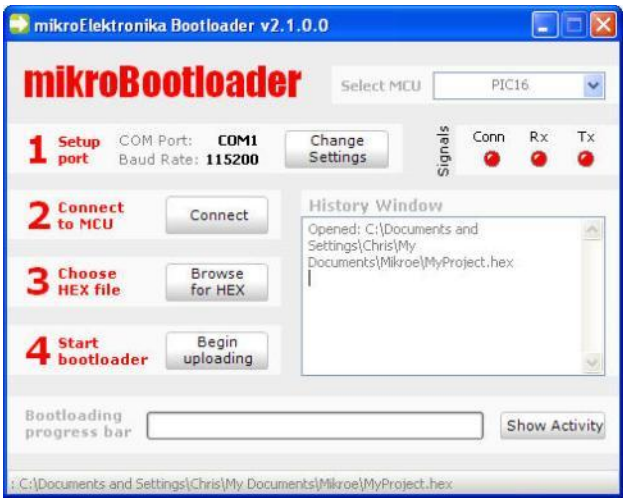
[● Externally]

Select Media: MMC

Options: [✓] JPEG as BMP, [✓] Use mE file system

Figure 69: Visual TFT: External storage option

At the end, the whole project is compiled and converted into HEX file (≈ similar to assembly code) which is afterwards loaded into microcontroller’s flash memory. This is done by mikroElektronika tool *microBootloader*, which communicates with microcontroller over USB cable (again presented as VCP thanks to FTDI chip integrated on the board).


mikroElektronika Bootloader v2.1.0.0

mikroBootloader

Select MCU: PIC16

COM Port: COM1

Baud Rate: 115200

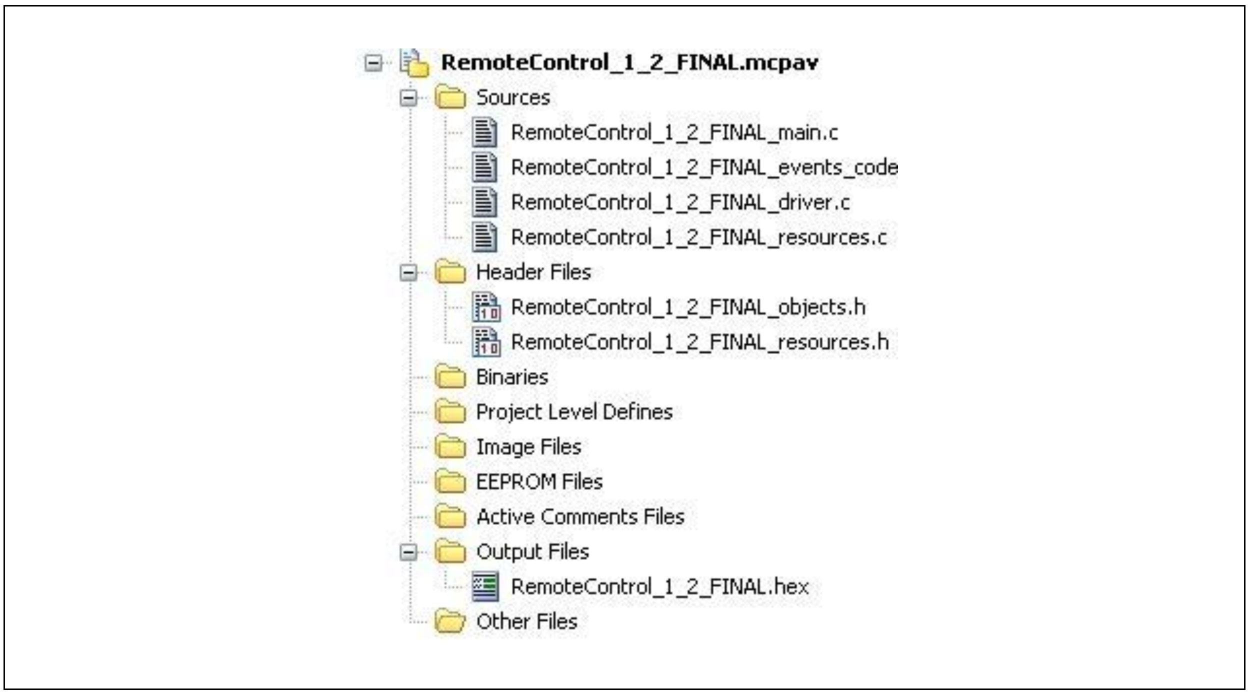
History Window:

Opened: C:\Documents and Settings\Chris\My Documents\Mikroe\MyProject.hex

: C:\Documents and Settings\Chris\My Documents\Mikroe\MyProject.hex

Figure 70: microBootloader software for loading created source code (compliled in HEX code) into microcontroller’s flash memory

Project, which is included on the CD along the thesis, consists in total of 4 (*.c) files and their header (*.h) files, which make it easier to share variables in between individual C-files.


```
RemoteControl_1_2_FINAL.mcpav
  Sources
    RemoteControl_1_2_FINAL_main.c
    RemoteControl_1_2_FINAL_events_code
    RemoteControl_1_2_FINAL_driver.c
    RemoteControl_1_2_FINAL_resources.c
  Header Files
    RemoteControl_1_2_FINAL_objects.h
    RemoteControl_1_2_FINAL_resources.h
  Binaries
  Project Level Defines
  Image Files
  EEPROM Files
  Active Comments Files
  Output Files
    RemoteControl_1_2_FINAL.hex
  Other Files
```

RemoteControl_1_2_FINAL_driver.c and *RemoteControl_1_2_FINAL_resources.c* files are auto-generated by compiler and GUI editor. The other two C-files contain my original source codes. Both can be found in Appendix J and Appendix K

8.3.1 RemoteControl_1_2_FINAL_main.c

At the very beginning, header files are loaded to include variables used in other files followed by defining new variables which will be used within this file. Variables `int InterruptCountLeft`, `int InterruptCountRight`, `char LeftButtonPressed`, `char RightButtonPressed`, `int count` are global variables defined outside of `void main()` function , because are common for all functions in this C-file. In the `void main()` function, all circuits are firstly initialized and afterwards controlled in the following way. Apart from initialization all the other commands are executed within endless loop `while (1) {...}` which maintains program in progress until the microcontroller resets or switches off.

✘ *Connection to UART*

Communication must be firstly initialized by command `UARTD0_Init(921600);` which tells microcontroller to use pins PD2 as RxD input and PD3 as TxD output (description of UARTD0 register in microcontroller data sheet) and baud rate 921600 (wireless module must be set to the same value). To write string to UART port `UARTD0_Write();` is used. Command `UARTD0_Data_Ready();` returns logical 1 if there are any incoming data on the line and those can be read and stored to variable as follows: `UartBuffer = UARTD0_Read();`

✘ *Button control*

Firstly the microcontroller is told to use pins PB1 and PB2 as an input lines by commands specifying port direction `PORTB_DIR.B1 = 0;` and `PORTB_DIR.B2 = 0;`. In order to read the state of line, we use function `LeftButtonState = Button(&PORTB_IN, 1, 1, 1);` `RightButtonState = Button(&PORTB_IN, 2, 1, 1);` where &PORTB_IN is name of port, second input parameter denotes the pin number (0-7), third is the period of time in ms when the line is observed. The last parameter sets, whether pin is active after Low→High (1) or High→Low (0) transition. Function returns value 255 if the pin is active for the given period, 0 in other cases.

✘ *Vibration motor*

Motor is connected to pin PC2 which is set to output direction `PORTC_DIR.B2 = 1;`. To drive the motor line state has to be pulled high. High or Low state of the line is controlled by commands `PORTC_OUT.B2 = 1;` for High and `PORTC_OUT.B2 = 0;` for Low state. Motor is not driven directly from this voltage but signal just opens/closes transistor barrier (see Chapter 8.2 schematic).

✘ *Buzzer (sound module)*

Similarly to vibration motor, at first the direction and pin number is specified. Buzzer is connected to output pin PF1 `PORTF_DIR.B1 = 1;` and initialized by command `Sound_Init(&PORTF_OUT, 1);` while first parameter is port name and second is pin number. In order to play sound of given frequency (e.g. 5000 Hz) and duration (e.g. 200 ms) command `Sound_Play(5000, 200);` is executed.

✘ *TFT display*

As the display is part of development board, the configuration commands are automatically added (pins, their directions). However to change any property of object in GUI during the application progress firstly command changing the property e.g. `Image1.Visible = 0;` is sent, followed by redrawing the object (necessary to display changed value) `DrawImage(&Image1);`. In this way, the circles displaying RB visibility changes their color, if the particular command is received on UART: E.g.

```
//Femur indicator
if (UartBuffer == FemurGreen){
      Circle1.Color = CL_LIME;
      DrawCircle(&Circle1);
}
```

✘ *Touch panel*

Every loop cycle, function `Check_TP();` is called to monitor, whether the panel was touched or not. Command buttons on the screen are set as active and whenever the panel is touched in the region of the button, one of defined events is executed. Events' functions are defined in another file *RemoteControl_1_2_FINAL_events_code.c*. Used events are: `void ButtonRound1Down()`, `void ButtonRound2Down()`, `void ButtonRound1Up()`, `void ButtonRound2Up()`, `void Image1Down()`, `void Image1Up()`. Because on-screen buttons must have the same function as external buttons connected to pins PB1, PB2 and external button does not cause this interrupt (jump to events code file), interconnection is done by common variables `char LeftButtonPressed;` and `char RightButtonPressed;` which informs, whether the button is pressed or not (no matter if it was pressed on external button or on touch screen).

Interrupt Service Routine (ISR)

Vibration motor vibrates for 100 ms if the button is pressed and since it is held microcontroller continuously sends certain command on UART port e.g. `UARTE0_Write(LeftButtonSendPressed);` (`char LeftButtonSendPressed = 0x02;`). Differentiation whether the button is pressed short or long is done in OrthoPilot application on PC (remote controller sends only "pressed", "not pressed"). However remote controller should inform surgeon, that he already holds the button for sufficient time. This is done by longer vibration (500 ms duration) after 1,8 s since the button was pressed (but must be still active!!!) Time measurement in microcontroller programming is generally done using time interrupts, which causes Interrupt Service Routines. As we need to measure time for both buttons separately, two timers were set. `Timer_Init(&TCC0, 3000);` and `Timer_Init(&TCC1, 3000);` commands initialize Timer C0(C1) interrupt at every 3000 us (3 ms). In the ISR functions

```
void Timer1Overflow_ISR() org IVT_ADDR_TCC0_OVF {
      InterruptCountLeft++;
}
void Timer2Overflow_ISR() org IVT_ADDR_TCC1_OVF {
      InterruptCountRight++;
}
```

global variables increment their values and are later evaluated to send motor vibration signal.

```
if (InterruptCountRight == 600)             //600*3000 us = 1,8 s
{
      ButtonRound2.Gradient_End_Color = 0xFF7F24;
      DrawRoundButton(&ButtonRound2);
      PORTC_OUT.B2 = 1;
      Delay_ms(500);
      PORTC_OUT.B2 = 0;
}
```

It is good to avoid entering ISR more times in one loop cycle (small timer ms value), because the condition above would be skipped and motor wouldn't vibrate. 3 ms has shown up to be good value (evaluated by sending `InterruptCountRight` value in every loop to HTerm console).

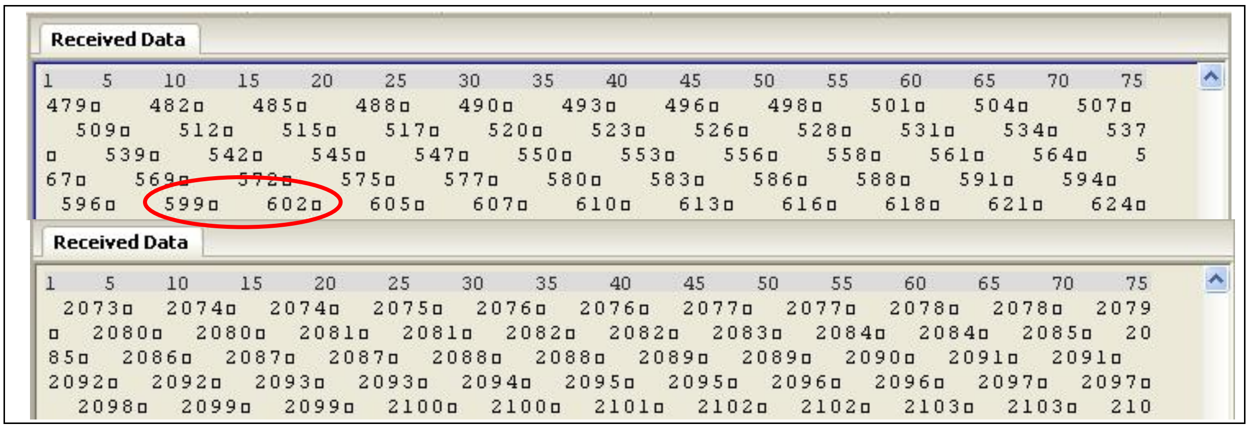

Received Data
```
1    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75
479□    482□    485□    488□    490□    493□    496□    498□    501□    504□    507□
  509□    512□    515□    517□    520□    523□    526□    528□    531□    534□    537
□    539□    542□    545□    547□    550□    553□    556□    558□    561□    564□    5
67□    569□    572□    575□    577□    580□    583□    586□    588□    591□    594□
 596□    599□    602□    605□    607□    610□    613□    616□    618□    621□    624□
```
Received Data
```
1    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75
 2073□  2074□  2074□  2075□  2076□  2076□  2077□  2077□  2078□  2078□  2079
□  2080□  2080□  2081□  2081□  2082□  2082□  2083□  2084□  2084□  2085□  20
85□  2086□  2087□  2087□  2088□  2088□  2089□  2089□  2090□  2091□  2091□
2092□  2092□  2093□  2093□  2094□  2095□  2095□  2096□  2096□  2097□  2097□
  2098□  2099□  2099□  2100□  2100□  2101□  2102□  2102□  2103□  2103□  210
```

Figure 71: HTerm: top: wrong setting (timer = 1ms), number 600 would be skipped. Bottom: correct setting (timer = 3ms), in every loop cycle the value is incremented at max once (or not at all)

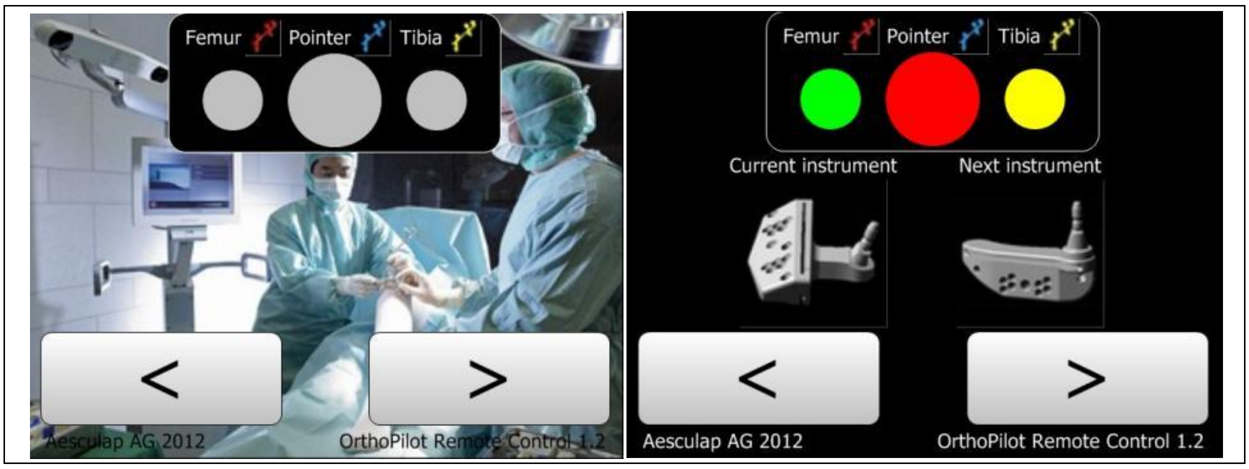


Figure 72: Application screenshots

9 Discussion & Conclusion

Targets of the thesis specified at the beginning have slightly changed throughout the thesis progress mostly due to company strategy, priorities and modifying overall concept. For example point 3) *“Perform configuration and tests with access point.”* was not done at all, because it is not intended to use AP in the OR anymore. Reason for this is the fact, that it is not necessary to use it as the same OR internal network can be done using ad-hoc setting between wireless modules and PC unit. Because one of the primary efforts is to minimize production cost, network with/without AP is a big difference (Cisco AP approx. 800 EUR vs. 3 pieces connectBlue OWS451modules: less than 300 EUR). Based on discussions with my supervisor Mr. Wehrle we decided to abandon field of extensive theoretical study of wireless networks (although it is also definitely needed for such a complex system) and devices testing, and focus more on practical, creative and more amusing work. This dealt with two particular interfaces between camera, RC and PC unit using newly tested wireless modules connectBlue OWS451.

Unfortunately, Camera-Wireless PCB couldn’t be finished and tested by the date of submitting the thesis, due to PCB manufacturing delay. Soldering components and evaluation will be done as soon as the PCBs are received during next weeks. However the circuit was tested on the Bread board as it was built before developing the PCB layout. It must be honestly admitted that expectation of higher frame rate compared to first prototype ([13]) was not fulfilled at all, only small increase (2 FPS) which was caused by using new technology (802.11 instead of BT). Additionally it was found out that the particular setting of COM port has much bigger influence on the transfer speed. The setting is related to VCP settings in Windows systems *(System → Device Manager → Ports (COM &LPT) → USB Serial Port (COM X) → Port Settings → Advanced → BM Options → Latency Timer value).*

These advanced settings are accessible only for virtual ports and are installed together with FTDI Serial to USB drivers. It allows user to configure properties of USB data packets, Timeouts (in case of no response). By setting Latency timer to second lowest value 2 ms (1 ms is not recommended as it is exactly duration of one USB packet frame) the frame rate was increased by 8 almost to 20 FPS (see table below).

Technology	Latency value = 16 (default)	Latency value = 2	Baud rate (PC side)
Cable	≈ 30 FP		---
Bluetooth	9,7 FPS	12,2 FPS	115200
802.11 /WLAN	12,8 FPS	15,0 FPS	
Bluetooth	11,3 FPS	14,3 FPS	921600
802.11 /WLAN	15,2 FPS	19,1 FPS	

Table 9: Camera-PC Frame rate overview

Probable reason for smaller frame rate compared to wired solution may be much higher number of interfaces, where the signal changes its character (USB → UART → TCP/IP → UART → RS-422), and small delay at each of them. Anyway, 20 FPS means, that camera obtains 20 transformation matrices per second, which is more than sufficient. To conclude, new prototype does not increase the speed significantly but its main advantages are small dimensions which allow device to be mounted directly on camera body.

Second practical task, remote controller development was meant just as demonstration of features offered by microcontroller in combination with display. Because of lack of time and insufficient development the circuits were built just on Bread board without creating any costly PCB. Another reason is that department team is not really convinced whether the remote control with embedded display will be used, due to its dimensions which make difficult to mount controller on an ergonomic handle. Another fact is, that use of 320x240 display and programs in C language may look as a step back as current devices (smartphones, tablets) offer much better performance, display resolution, programming options (Java, C++) and overall richer possibilities. On the other hand it is necessary to mention that it is more difficult (almost impossible) to obtain medical accreditation for such a device unless it is done (agreed with) by manufacturer.

Regarding the overall concept, there is an expected limitation in use of 2 (3) wireless modules (RC, Camera, PC, footswitch) at one time, which haven’t been evaluated yet. The limitation is related to addressing particular device, because the WLAN wireless modules don’t support extended data mode (described in Chapter 6.3.3). Whenever Listener (module at PC side) sends data intended for one peer, these data are obtained by all the others as well. Currently OrthoPilot application is coded to run each communication (camera, footswitch) in separated thread. It is possible because devices use different interface or at least port. Once the both devices will be connected to one module on PC, there is no chance to divide application in two threads as it is not possible to connect twice to one COM port. Thus solution would be to recode application into one thread, but this will slow down the whole application. Another problem would occur if the PC sends data to camera and but as a reply, data from remote controller would be received. Because camera replies with error message for every unknown command, much unrequired data would need to be transferred.

Things would go much easier if the PC module (wireless modules generally) allow to create two independent listeners on two different ports (e.g. 7777 and 8888). This is unfortunately not possible, but nice solution is (theoretically) hidden in wireless network card which is embedded in every panel PC (at least those which are currently being tested at ENT department). Because the devices (connectBlue modules and wireless network card) use the same technology and data are sent in TCP/IP packets, the end device can be any of the devices supporting this technology. Difference is only in TCP/IP ↔ application interface.

When current schema is used (PC wireless module OWS451), data in application are sent to serial COM port. If the schema described above (wireless network card in PC) would be used, another physical layer would be used and the serial port would be bypassed. Network programming using Internet sockets would be applied in application instead of commands to write/read to/from serial port. In this case, more ports one on IP address could be opened at the same time (address of wireless network card - **server**) and application could run single thread for each connected module (every peer -**client**- connected to another port). Besides this advantage, it is also possible, that higher frame rates might be reached (camera ↔ PC connection), because two of the converting interfaces (TCP/IP → UART → USB) would be omitted. Here comes one very last figure for easier understanding of my idea.

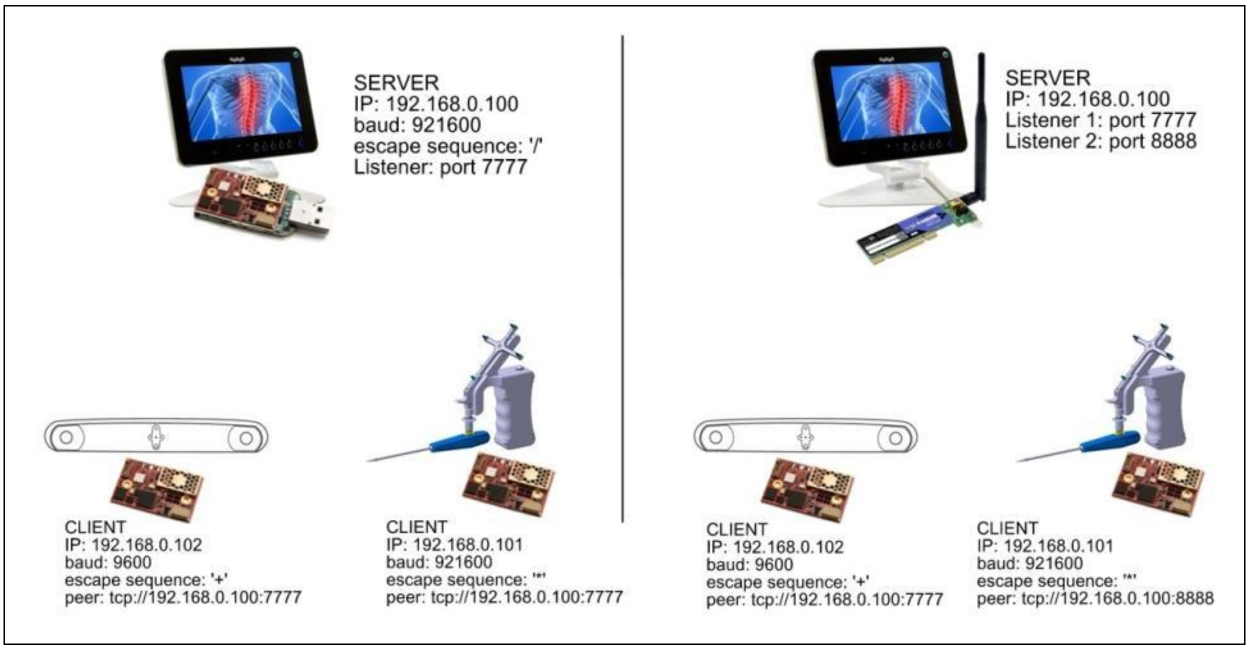


Figure 73: Current state (left) and future idea (right) of the network concept

I conclude my thesis with a hope that the whole thesis together with this last idea will help my followers understand and continue my work as well as my part couldn't have been done without plenty of great ideas and admirable knowledge of my predecessors and current colleagues.

References

[1] *Aesculap*: About Company
http://www.thinkaesculap.be/aesculap_gb 20.6.2012

[2] *OrthoPilot*® Navigation System
http://www.orthopilot.com/ 22.6.2012

[3] http://www.libones.com/orthopilot.htm 22.6.2012

[4] *NDI*: Polaris Family of Optical Tracking Systems
http://www.ndigital.com/medical/polarisfamily.php 25.6.2012

[5] *BBraun*: OrthoPilot® Next Generation
http://braunoviny.bbraun.cz/clanky/ 1.7.2012

[6] *Wedekind*, D.: Fachpraktikumsbericht WS11, Aesculap AG, Tuttlingen, Germany, 2011

[7] *Wehrle*, Ch.: Ultraschall-Navigationssystem in der Orthopädie, Diploma Thesis, Fachhochschule Konstanz, Konstanz, Germany, 2005

[8] *National Instruments*: Serial Communication Overview:
http://www.ni.com/white-paper/2895/en 7.7.2012

[9] *TalTech*: Introduction to Serial Communications:
http://www.taltech.com/support/entry/serial_intro 7.7.2012

[10] *Wikipedia*: Universal asynchronous receiver/transmitter
http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter 7.7.2012

[11] http://www.coolgear.com/images/DB9PMF3FT11.jpg 7.7.2012

[12] http://techpubs.sgi.com/library/dynaweb_docs/0650/SGI_EndUser/books/MSB3xx_UG/sgi_html/figures/db9.pinouts.gif 7.7.2012

[13] *Nickel*, V.: Evaluirung geeigneter kabelloser Übertragungssysteme zur Anbindung unterschiedlicher Peripheriegeräte an das Ortho-Pilot-Navigationssystem. Bachelor Thesis, Hochschule Furtwangen University, Furtwangen, 2011.

[14] *Wikipedia*: RS-422
http://en.wikipedia.org/wiki/RS-422 7.7. 2012

[15] *Wikipedia*: Differential signaling
http://en.wikipedia.org/wiki/Differential_signaling 7.7.2012

[16] Characteristics of RS232, RS422, RS423 and RS485
http://www.lammertbies.nl/comm/info/RS-485.html 8.7.2012

[17] Logic Levels: TTL, 5V-CMOS, 3,3V-CMOS
http://www.ne555.at/elektronik-grundlagen/digitaltechnik/310.html 8.7.2012

[18] *Wikipedia*: Universal Serial Bus
http://en.wikipedia.org/wiki/Universal_Serial_Bus#Communication 8.7.2012
http://cs.wikipedia.org/wiki/Universal_Serial_Bus 8.7.2012

[19] The development of wireless technologies

[20] http://www.radio-electronics.com/info/wireless/overview/wireless_dev.php 8.7.2012

[21] *Wikipedia*: ISM-Bands
http://en.wikipedia.org/wiki/ISM_band 8.7.2012

[22] *Elektronik Kompendium*: IEEE 802.11 / WLAN-Grundlagen
http://www.elektronik-kompendium.de/sites/net/0610051.htm 8.7.2012

[23] 2.4 GHz 802.11 Channel-to-Frequency Mappings
http://www.l-com.com/content/Bandpass_Filters_FAQ.html 9.7.2012

[24] *connectBlue*: Wireless LAN
http://www.connectblue.com/technologies/wireless-lan-wlan/ 9.7.2012

[25] *Elektronik Kompendium*: Bluetooth 1.0 / 1.1 / 1.2
http://www.elektronik-kompendium.de/sites/kom/0803301.htm

[26] *connectBlue*: Classic Bluetooth Technology
http://www.connectblue.com/technologies/classic-bluetooth-technology/ 10.7.2012

[27] *connectBlue*: IEEE 802.15.4 / ZigBee
http://www.connectblue.com/technologies/ieee-802154-zigbee/ 10.7.2012

[28] *ZigBee* Alliance: Standards
http://www.zigbee.org/About/AboutTechnology/Standards.aspx 10.7.2012

[29] *NDSsi*, Ehsan Ayar, 2010, UWB Wireless Video Transmission Technology in Medical Applications
http://ndssi.com/new/data/uploads/pdf/Surgical/ZeroWire/Wireless-Video-Transmission-in-Medical-Applications.pdf 10.7.2012

[30] *Electronic Design*: Discover WiMedia UWB
http://electronicdesign.com/article/communications/discover-wimedia-uwb19836 10.7.2012

[31] *Wikipedia:* Wireless USB
http://en.wikipedia.org/wiki/Wireless_USB 10.7.2012

[32] *EnOcean*: Radio Technology:
http://www.enocean.com/en/radio-technology/ 11.7.2012

[33] *Wikipedia:* Electromagnetic compatibility
http://en.wikipedia.org/wiki/Electromagnetic_compatibility 11.7.2012

[34] *Gärtner*, A.: Medizintechnik und Informationstechnologie, Köln: TÜV Media GmbH 2011, ISBN 978-3-8249-1415-9

[35] ISO 14971:2007, International Standard: Medical devices - Application of risk management to medical devices

[36] IEC 60601-1:2005, International Standard: Medical electrical equipment 3rd Edition

[37] *Intertek*: IEC 60601-1: Changes from 2nd to 3rd Edition
http://www.ownersguidepdf.com/download-manual-ebook/iec-60601-pdf.pdf

[38] IEC 62304: 2006, International Standard: Medical device software-Software life cycle processes

[39] IEC 80001-1:2010, International Standard: Application of risk management for IT-networks incorporating medical devices

[40] FCC: Code of Federal Regulations, Title 47, Part 15: Radio frequency devices, Edition 10-1-11

[41] *ETSI*: The R&TTE Directive http://www.etsi.org/website/Technologies/RTTE.aspx 11.7.2012

[42] *European Commission*: Radio and telecommunications terminal equipment http://ec.europa.eu/enterprise/sectors/rtte/documents/ 11.7.2012

[43] *FDA*: Draft Guidance for Industry and FAD Staff: RF Wireless Technology in Medical Devices, 2007 http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm077272.pdf 11.7.2012

[44] *FDA*: Draft Guidance for Industry and FAD Staff: Mobile Medical applications, 2011 http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM263366.pdf 11.7.2012

[45] *Tikkanen*, J.: Wireless Electromagnetic Interference (EMI) in Healthcare Facilities, Ontario: BlackBerry http://www.blackberry.com/solutions/pdfs/Healthcare/Wireless_EMI_in_Healthcare_Facilities_White_Paper.pdf 12.7.2012

[46] *Bosch*: White Paper: Video Compression for CCTV http://www.pinders.com/Video%20Compression%20for%20CCTV.pdf 28.7.2012

[47] *connectBlue*: cB-OBS433 Electrical Mechanical Data Sheet http://support.connectblue.com/display/PRODBTSPA/cB-OBS433+Electrical+Mechanical+Data+Sheet 15.7.2012

[48] *connectBlue*: cB-OWS451 and OWL253 Electrical Mechanical Data Sheet http://support.connectblue.com/display/PRODWSPA/cB-OWS451+and+OWL253+Electrical+Mechanical+Data+Sheet 15.7.2012

[49] *connectBlue*: Wireless LAN serial port adapter AT commands http://support.connectblue.com/display/PRODWSPA/Wireless+LAN+serial+port+adapter+AT+commands 16.7.2012

[50] *Maxim*: RS-485/RS-422 Transceivers Data Sheet http://www.maxim-ic.com/datasheet/index.mvp/id/1079 20.7.2012

[51] *Pulse*: Understanding Common Mode Noise http://www.pulseelectronics.com/download/3100 23.7.2012

[52] Decoupling, bypassing capacitors http://www.vagrearg.org/?p=decoupling 23.7.2012

[53] *NDI*: Hybrid Polaris Spectra User Guide, Rev. 2, March 2008

[54] *Linear Technology*: LT3663 - 1.2A Step-Down Switching converter http://cds.linear.com/docs/Datasheet/3663fb.pdf 23.7.2012

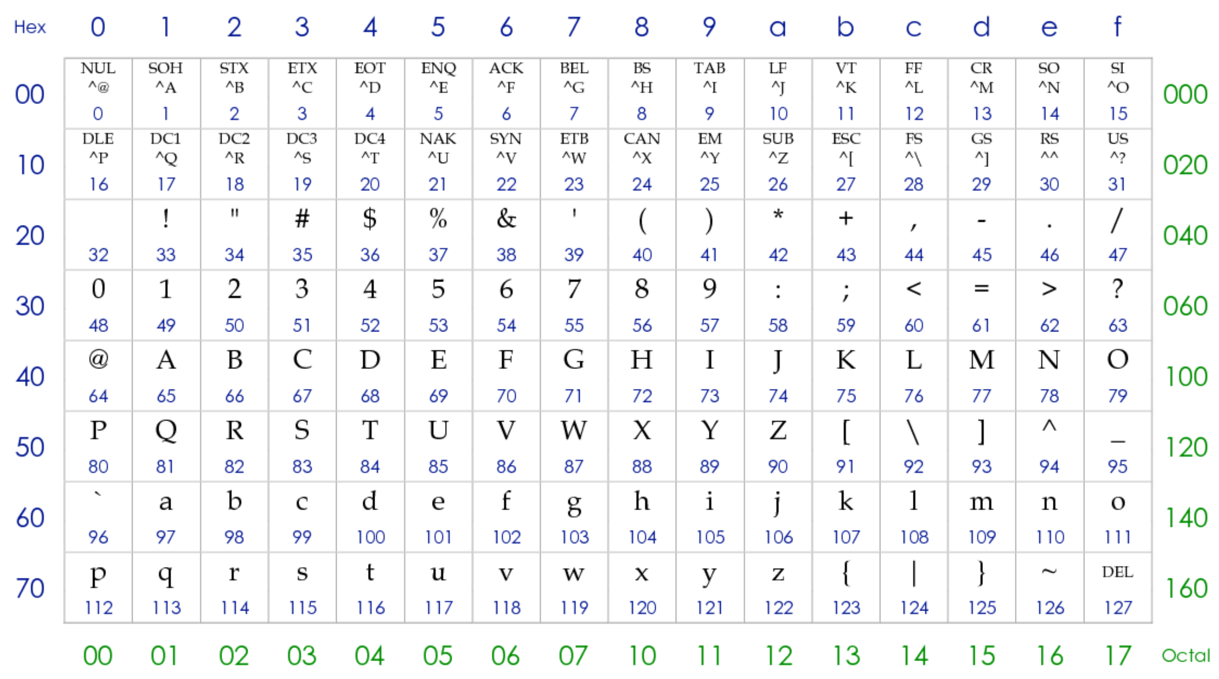
[55] *MikroElektronika*: mikroMedia for XMEGA Schematic http://www.mikroe.com/eng/downloads/get/1668/mikromedia_xmega_sch_v111.pdf 30.7.2012

[56] *Mackay, S.;Wright, E.;Reynders, D.;Park, J.:* Practical Industrial Data Networks: Design, Instalation and troubleshooting, Newnes, Oxford, 2004. ISBN 0-7506-5807-X

[57] *O'Rourke, B.:* Wireless HD Video Technologies: WHDI, WirelessHD, and WiGig Try to Create a Market, In-Stat-an NPD Group Company, 2012. Product No: IN1205134MI

Appendices

Appendix A ASCII Code Table [a]



Hex	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00	NUL ^@ 0	SOH ^A 1	STX ^B 2	ETX ^C 3	EOT ^D 4	ENQ ^E 5	ACK ^F 6	BEL ^G 7	BS ^H 8	TAB ^I 9	LF ^J 10	VT ^K 11	FF ^L 12	CR ^M 13	SO ^N 14	SI ^O 15	000
10	DLE ^P 16	DC1 ^Q 17	DC2 ^R 18	DC3 ^S 19	DC4 ^T 20	NAK ^U 21	SYN ^V 22	ETB ^W 23	CAN ^X 24	EM ^Y 25	SUB ^Z 26	ESC ^[27	FS ^\ 28	GS ^] 29	RS ^^ 30	US ^? 31	020
20	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	040
30	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	060
40	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	100
50	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95	120
60	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	140
70	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127	160
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	Octal

Appendix B 2,4 GHz band channels [b]

channel	frequency(MHz)	United States	Most of world	Japan
1*	2412	Yes	Yes	Yes
2	2417	Yes	Yes	Yes
3	2422	Yes	Yes	Yes
4	2427	Yes	Yes	Yes
5*	2432	Yes	Yes	Yes
6	2437	Yes	Yes	Yes
7	2442	Yes	Yes	Yes
8	2447	Yes	Yes	Yes
9*	2452	Yes	Yes	Yes
10	2457	Yes	Yes	Yes
11	2462	Yes	Yes	Yes
12	2467	No	Yes	Yes
13*	2472	No	Yes	Yes
14	2484	No	No	11b only

*With 802.11g and newer only the channels 1, 5, 9, and 13 shall be used in order to obey the non-overlapping 20 MHz OFDM channel scheme borrowed from 802.11a.

[a] http://www.docstoc.com/docs/9862447/ASCII-Chart 12.7.2012

[b] Wikipedia: List of WLAN channels
http://en.wikipedia.org/wiki/List_of_WLAN_channels 12.7.2012

Appendix C 5 GHz band channels [c]

channel	frequency(MHz)	United States	Europe	Japan
183	4915	No	No	No
184	4920	No	No	Yes
185	4925	No	No	No
187	4935	No	No	No
188	4940	No	No	Yes
189	4945	No	No	No
192	4960	No	No	Yes
196	4980	No	No	Yes
7	5035	No	No	No
8	5040	No	No	No
9	5045	No	No	No
11	5055	No	No	No
12	5060	No	No	No
16	5080	No	No	No
34	5170	No	No	Yes
36	5180	Yes	Yes	Yes
38	5190	No	No	Yes
40	5200	Yes	Yes	Yes
42	5210	No	No	Yes
44	5220	Yes	Yes	Yes
46	5230	No	No	Yes
48	5240	Yes	Yes	Yes
52	5260	Yes	Yes-DFS/TPC	Yes-DFS/TPC
56	5280	Yes	Yes-DFS/TPC	Yes-DFS/TPC
60	5300	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
64	5320	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
100	5500	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
104	5520	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
108	5540	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
112	5560	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
116	5580	Yes/DFS	Yes-DFS/TPC	Yes-DFS/TPC
120	5600	No	Yes-DFS/TPC	Yes-DFS/TPC
124	5620	No	Yes-DFS/TPC	Yes-DFS/TPC
128	5640	No	Yes-DFS/TPC	Yes-DFS/TPC
132	5660	Yes /DFS	Yes-DFS/TPC	Yes-DFS/TPC
136	5680	Yes /DFS	Yes-DFS/TPC	Yes-DFS/TPC
140	5700	Yes /DFS	Yes-DFS/TPC	Yes-DFS/TPC
149	5745	Yes	No	No
153	5765	Yes	No	No
157	5785	Yes	No	No
161	5805	Yes	No	No
165	5825	Yes	No	No

[c] Wikipedia: List of WLAN channels
http://en.wikipedia.org/wiki/List_of_WLAN_channels 12.7.2012

Appendix D AT commands – selection [48]

AT*AMRS RS-232 Settings

Syntax	Description
`AT*AMRS=<baud_rate>,<data_bits>,<stop_bits>,<parity>,<flow_control>,<reserved>,<store><CR>`	Write the RS-232 settings. Automatically stores the settings.
`AT*AMRS?`	Read the RS-232 settings.

Parameters	Type	Description
baud_rate	integer	Sets the baud rate. 1 = 300 2 = 1200 3 = 2400 4 = 4800 5 = 9600 6 = 19200 7 = 38400 8 = 57600 (default) 9 = 115200 10 = 230400 11 = 460800 12 = 921600 13 = 1382400 14 = 2764800 > 300 = set to this baud rate
data_bits	integer	Sets the data bits. 1 = 8 bits (default) 2 = 7 bits 3 = 6 bits 4 = 5 bits
stop_bits	integer	Sets stop bit. 1 = 1 bit (default) 2 = 2 bits
parity	integer	Sets parity. 1 = None (default) 2 = Odd 3 = Even
flow_control	integer	Flow control settings 1 = cts/rts (default) 2 = None
reserved	integer	Reserved for future use. Use 0.
store	integer	0 = Do not store 1 = Store (will store between reboots)

Responses	Description
`<CR><LF>*AMRS<baud_rate>,<data_bits>,<stop_bits>,<parity>,<flow_control><CR><LF>OK<CR><LF>`	Successful read response
`<CR><LF>OK<CR><LF>`	Successful response
`<CR><LF>ERROR<CR><LF>`	Error response

AT*ADDM Enter Data Mode

Syntax	Description
`AT*ADDM<CR>`	Enter data mode.

Description	Responses
`<CR><LF>OK<CR><LF>`	Successful response
`<CR><LF>ERROR<CR><LF>`	Error response

AT*AMWS Watchdog Settings

Syntax	Description
`AT*AMWS=<reserved1>,<inactivity_timeout>,<reserved3>,<reserved4>,<reset>,<store><CR>`	Write the watchdog settings. The watchdog functionality will disconnect from a remote peer if one of the given conditions are met.
`AT*AMWS?`	Read the watchdog settings

Parameters	Type	Description
reserved1	integer	Reserved for future use. Use 0.
inactivity_timeout	integer	Disconnect WLAN after this long idle time in seconds (default 0, i.e. inactivated)
reserved3	integer	Reserved for future use. Use 0.
disconnect_reset	integer	Will reset the module if all peers are disconnected. 1 = On, 0 = Off, Default = 0
reset	integer	1 Will reset the unit immediately. (Will not store nor return any response)
store	integer	0 = Do not store 1 = Store (will store between reboots)

Responses	Description
`<CR><LF>*AMWS:<reserved1>,<inactivity_timeout>,<reserved3>,<disconnect_reset>,<reset><CR><LF>OK<CR><LF>`	Successful read response
`<CR><LF>OK<CR><LF>`	Successful response
`<CR><LF>ERROR<CR><LF>`	Error response

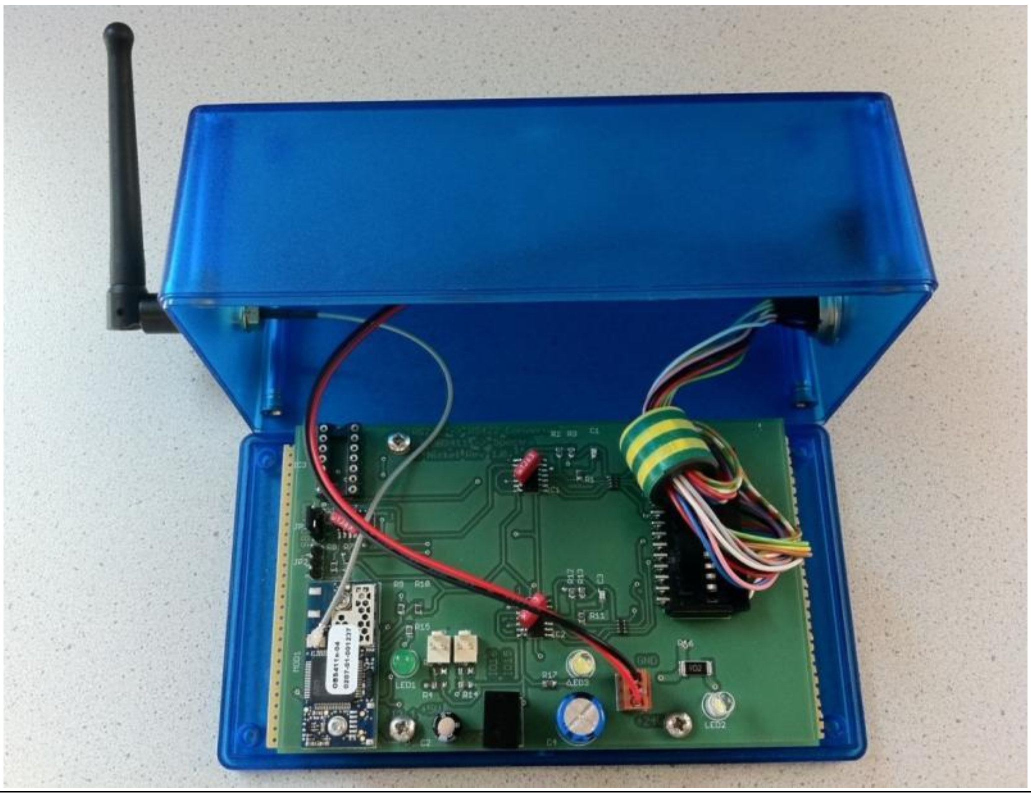
AT*ANDHCP DCHP Activation

Syntax	Description
`AT*ANDHCP=<on>,<store><CR>`	Activate/deactivate DHCP. If activated, this will take precedence over settings made with AT*ANIP.
`AT*ANDHCP?`	Read the current DHCP setting

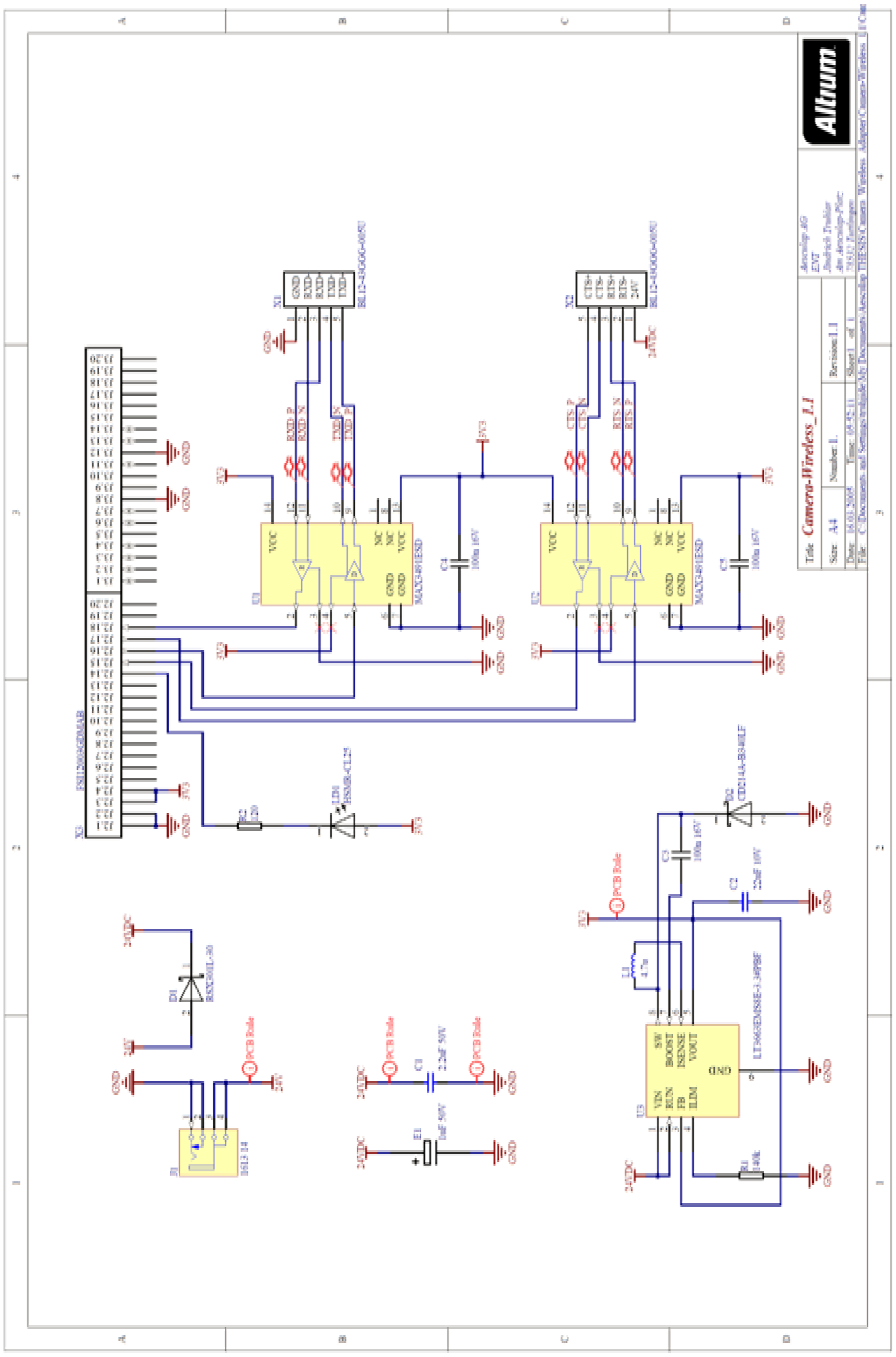
Parameters	Type	Description
On	integer	0 = Use static IP address (default) 1 = Acquire an IP address using DHCP 2 = DHCP Server. Use static IP address + act as DHCP server.
store	integer	0 = Do not store 1 = Store (will store between reboots)

Responses	Description
`<CR><LF>*ANDHCP:<on><CR><LF>OK<CR><LF>`	Successful read response
`<CR><LF>OK<CR><LF>`	Successful response
`<CR><LF>ERROR<CR><LF>`	Error response

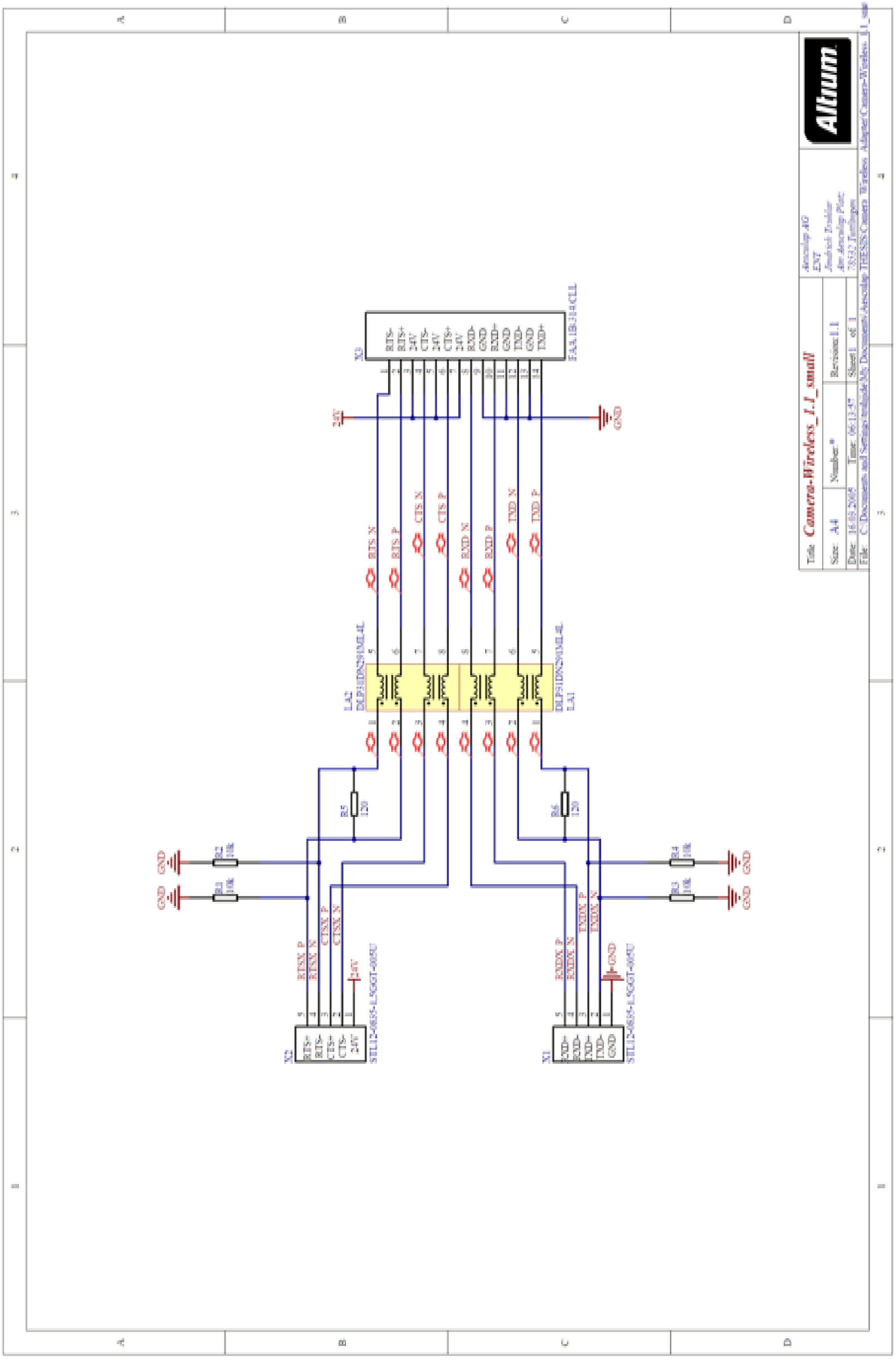
Appendix E Camera - module interface solution according to [13]



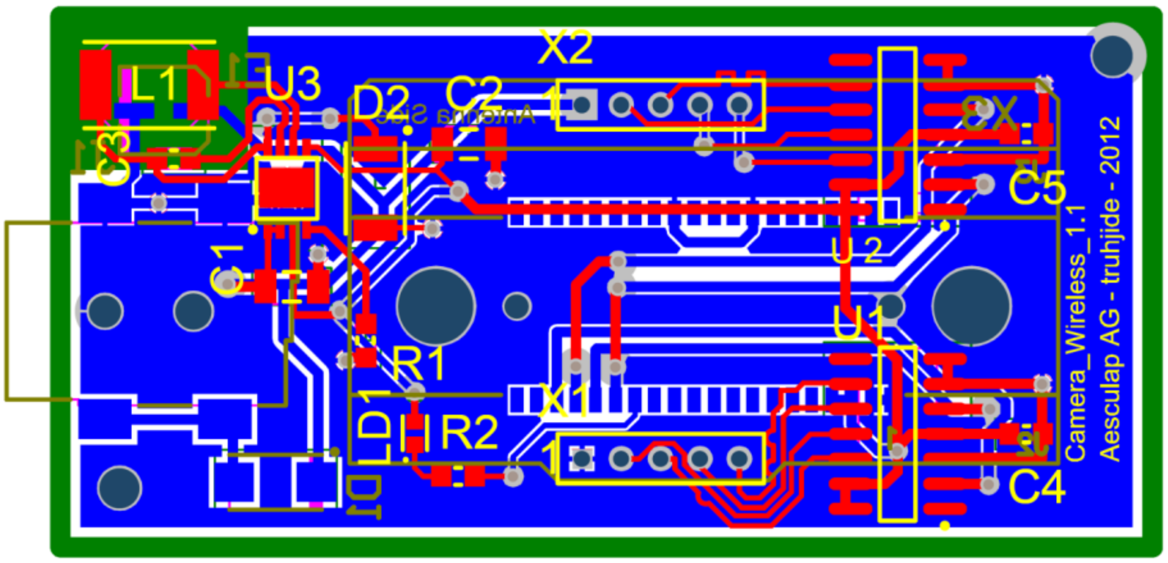
Appendix F Camera_Wireless_1.1 Schematic (A)

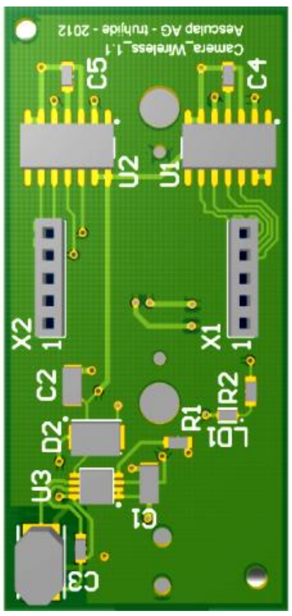


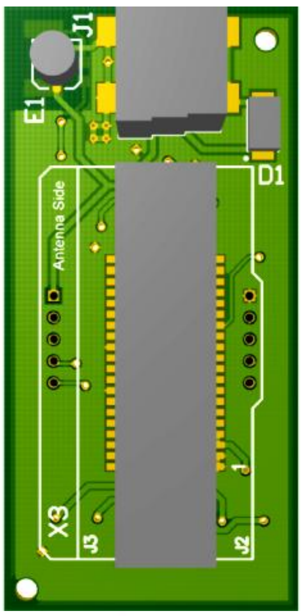
Appendix G Camera_Wireless_1.1_small Schematic (B)

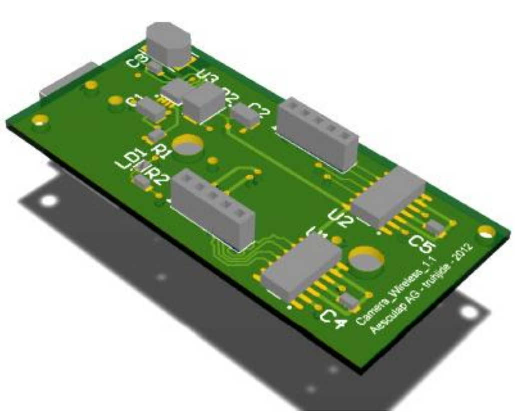


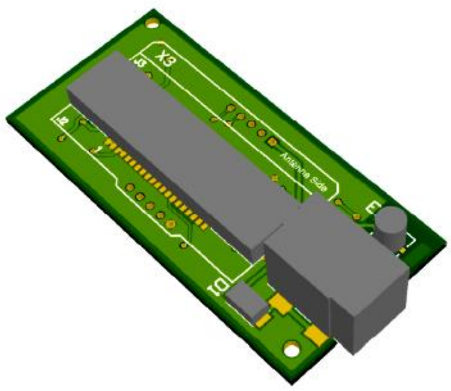
Appendix H Camera_Wireless_1.1 PCB + 3D model



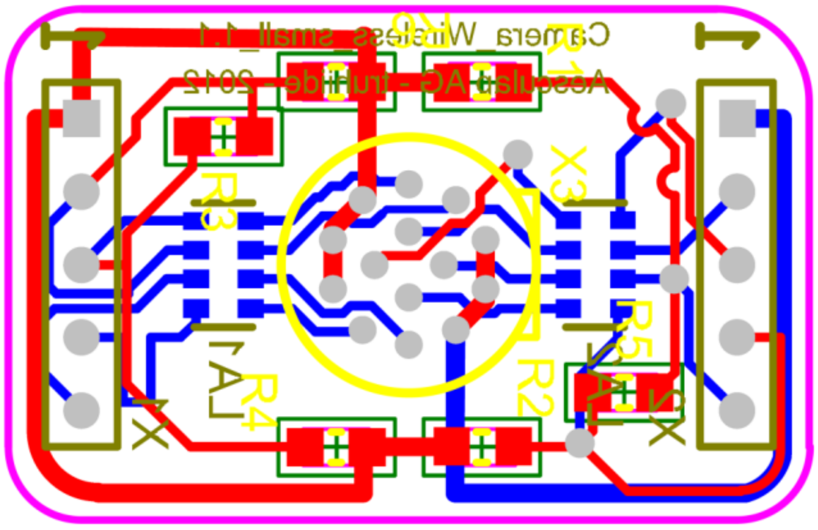


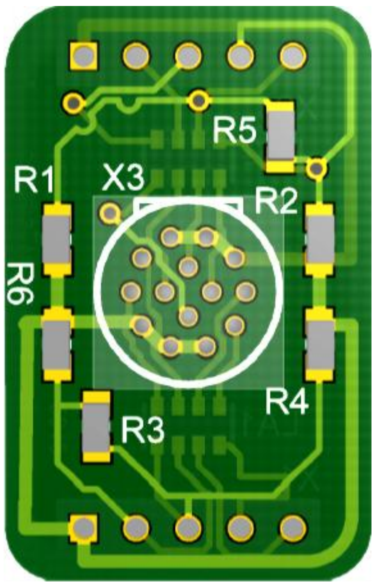


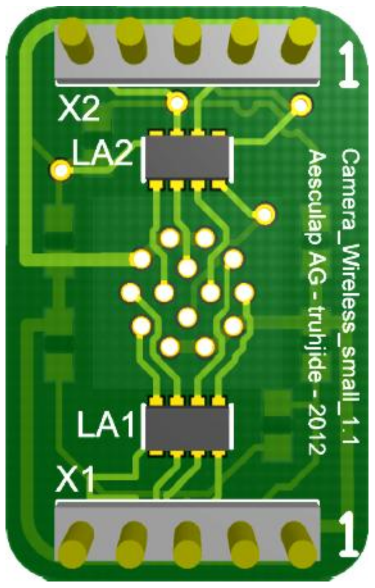


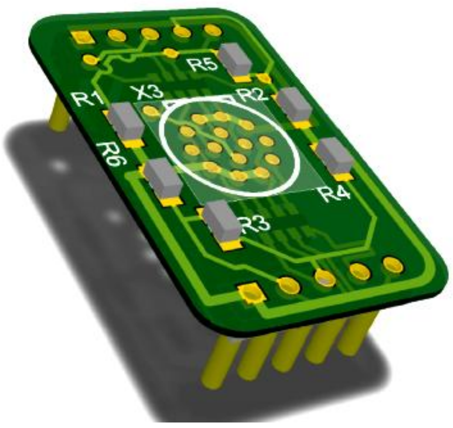


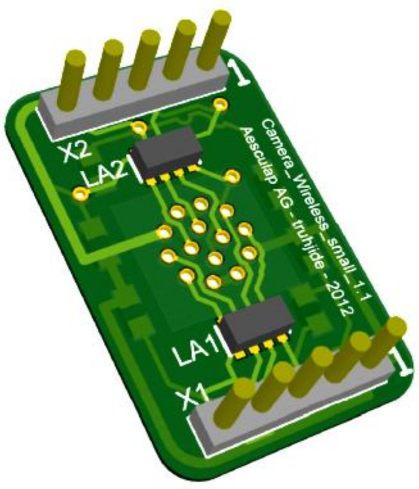
Appendix I Camera_Wireless_1.1_small PCB + 3 D model











Appendix J RemoteControl_1_2_FINAL_main.c

```c
/*
 * Project name:
     RemoteControl_1_2_FINAL.vtft
 * Created by:
     Truhlar Jindrich (truhjide), Aesculap AG
 * Date of creation
     1.07.2012
 * Time of creation
     07:23:15
 * Test configuration:
     MCU:              ATxmega128A1
     Dev.Board:        mikromedia_for_XMEGA
                       http://www.mikroe.com/eng/products/view/688/mikromedia-for-
xmega/
     Oscillator:       32000000 Hz
     SW:               mikroC PRO for AVR
                       http://www.mikroe.com/eng/products/view/228/mikroc-pro-for-
avr/
 * Description:
     MCU management for Remote Control 1.2 prototype.
     - sends and recieves commands via UART interface
     - vibration allert
     - sound beep allert
     - rigid body visibility - traffic light
     - current and following instrument
     - forward and backward button (touch panel(does not support pressing both
buttons simultineously) + external buttons)
     - images and fonts are stored externally on SD card:  'RemoteCo.RES' file
 */

#include "RemoteControl_1_2_FINAL_objects.h"
#include "RemoteControl_1_2_FINAL_resources.h"

//------------------------------------------------------------------------------
//Global variables
//------------------------------------------------------------------------------
int InterruptCountLeft = 0;
int InterruptCountRight = 0;
char LeftButtonPressed = 0;
char RightButtonPressed = 0;
int count = 0;                                  //to send 0x00 only once, if not == 0,
dont send 0x00 command

void main() {
 //-----------------------------------------------------------------------------
 //Initialization, variable defining
 //-----------------------------------------------------------------------------
     unsigned short LeftButtonState = 0;      //Logic level on pin PB1 (dafault LOW)
     unsigned short RightButtonState = 0;     //Logic level on pin PB2 (dafault LOW)
     unsigned short LeftButtonStateOld = 0;       //old state
     unsigned short RightButtonStateOld = 0;      //old state
     int i = 0;                                   //increment
     int j = 0;                                   //increment
     char UartBuffer;                             //stores received data from UART


     const unsigned long Images[] =   { a1_jpg,  a2_jpg,  a3_jpg,
                                        a4_jpg,  a5_jpg,  a6_jpg,
                                        a7_jpg,  a8_jpg,  a9_jpg,
                                        a10_jpg, a11_jpg, a12_jpg,
                                        a13_jpg, a14_jpg, a15_jpg,
                                        a16_jpg, a17_jpg, a18_jpg };
     //Output commands
     char LeftButtonSendPressed    = 0x02;        // (all compatible with
footswitch class of OrthoPilot software)
     char RightButtonSendPressed   = 0x40;
     char BothButtonsSendPressed   = 0x01;
     char NoButtonSendPressed      = 0x00;
     //Input commands
     char FemurGreen               = 0xA1;
     char FemurYellow              = 0xA2;
     char FemurRed                 = 0xA3;
     char TibiaGreen               = 0xB1;
     char TibiaYellow              = 0xB2;
     char TibiaRed                 = 0xB3;
     char PointerGreen             = 0xAA;
     char PointerYellow            = 0xBB;        // to keep compatibility with
previous RemoteControl 1.0
     char PointerRed               = 0xDD;        //
     char NextInstrument           = 0x10;
     char PreviousInstrument       = 0x20;

     Start_TP();

    //---------------------------------------------------------------------------
    //testing connection-Blinking circles
    //---------------------------------------------------------------------------

    Circle1.Color = CL_LIME;
    DrawCircle(&Circle1);
    Delay_ms(1000);
    Circle1.Color = 0x8410;
    DrawCircle(&Circle1);

    Circle2.Color = CL_YELLOW;
    DrawCircle(&Circle2);
    Delay_ms(1000);
    Circle2.Color = 0x8410;
    DrawCircle(&Circle2);

    Circle3.Color = CL_RED;
    DrawCircle(&Circle3);
    Delay_ms(1000);
    Circle3.Color = 0x8410;
    DrawCircle(&Circle3);

    //---------------------------------------------------------------------------
    //vibration motor test
    //---------------------------------------------------------------------------
    PORTC_DIR.B2 = 1;        // Set PC2 pin as output

    while  (i <= 2){
    PORTC_OUT.B2 = 1;
    Delay_ms(100);
    PORTC_OUT.B2 = 0;
    Delay_ms(100);
    i++;
    }

    //---------------------------------------------------------------------------
    //buzzer test
    //---------------------------------------------------------------------------
    PORTF_DIR.B1 = 1;        // Set PF1 pin as output
    Sound_Init(&PORTF_OUT, 1);
    Sound_Play(5000, 200);
    Sound_Play(3400, 200);


    //---------------------------------------------------------------------------
    //Enabling time interrupts
    //---------------------------------------------------------------------------

    PMIC_CTRL = 0b100;                // Enable medium level interrupts
    CPU_SREG.B7 = 1;
    Timer_Init(&TCC0, 3000);      // Initialize TimerC0 interrupt at every 3 ms
(avoid more interrupts in one loop-unstability!!!)
    Timer_Init(&TCC1, 3000);      // Initialize TimerC1 interrupt at every 3 ms
(avoid more interrupts in one loop-unstability!!!)

    //----------------------------------------------------------------------------
    //Buttons pin initialization
    //----------------------------------------------------------------------------
    PORTB_DIR.B1 = 0;         // Set PB1 pin as input (LEFT BUTTON)
    PORTB_DIR.B2 = 0;         // Set PB2 pin as input (RIGHT BUTTON)
    //----------------------------------------------------------------------------
    //Initializing UART communication
    //----------------------------------------------------------------------------

    UARTE0_Init(921600);               //Baud 921 600, PD2=Rx, PD3=Tx  (see manual)
    Delay_ms(1000);


  while (1) {
    Check_TP();

    LeftButtonState = Button(&PORTB_IN, 1, 1, 1);
    RightButtonState = Button(&PORTB_IN, 2, 1, 1);

    /*
    IntToStr(InterruptCountLeft, txtleft);         //used to see  real-time value
of InterruptCount on serial port in order to set good value for timer
    UARTD0_Write_Text(txtleft);
    */

    //----------------------------------------------------------------------------
    //LEFT EXTERNAL BUTTON
    //----------------------------------------------------------------------------
       if (LeftButtonState != 0 && LeftButtonPressed == 0){
          ButtonRound1Down();
          LeftButtonStateOld = LeftButtonState;
       }
       if (LeftButtonState == 0 && LeftButtonStateOld != LeftButtonState){
            Image1Up();
            LeftButtonStateOld = LeftButtonState;
       }
       if   (LeftButtonPressed == 1 && RightButtonPressed == 0)
       {
            //if (InterruptCountLeft > 5)                 //send command after
0,015s , in this time, it waits for pressing also right button.gives time if the
user wants to press both buttons
            UARTE0_Write(LeftButtonSendPressed);
            if (InterruptCountLeft == 600)           //600*3000 us = 1,8 s
               {
               ButtonRound1.Gradient_End_Color = 0xFF7F24;
               DrawRoundButton(&ButtonRound1);
               PORTC_OUT.B2 = 1;
               Delay_ms(500);
               PORTC_OUT.B2 = 0;
               }
       }
    //----------------------------------------------------------------------------
    //RIGHT EXTERNAL BUTTON
    //----------------------------------------------------------------------------
       if (RightButtonState != 0 && RightButtonPressed == 0){
          ButtonRound2Down();
          RightButtonStateOld = RightButtonState;
       }
       if (RightButtonState == 0 && RightButtonStateOld != RightButtonState){
            Image1Up();
            RightButtonStateOld = RightButtonState;
       }
       if   (RightButtonPressed == 1 && LeftButtonPressed == 0)
       {
            //if (InterruptCountRight > 5)                //send command after
0,015s , in this time, it waits for pressing also left button.gives time if the
user wants to press both buttons
            UARTE0_Write(RightButtonSendPressed);
            if (InterruptCountRight == 600)             //600*3000 us = 1,8 s
               {
               ButtonRound2.Gradient_End_Color = 0xFF7F24;
               DrawRoundButton(&ButtonRound2);
               PORTC_OUT.B2 = 1;
               Delay_ms(500);
               PORTC_OUT.B2 = 0;
               }
       }
    //----------------------------------------------------------------------------
    //BOTH BUTTONS PRESSED
    //----------------------------------------------------------------------------
         if   (RightButtonPressed == 1 && LeftButtonPressed ==1)
       {
            UARTE0_Write(BothButtonsSendPressed);
            if (InterruptCountRight == 600 )             //600*1000 us = 1,8 s
               {
               ButtonRound1.Gradient_End_Color = 0xFF7F24;
               DrawRoundButton(&ButtonRound1);
               ButtonRound2.Gradient_End_Color = 0xFF7F24;
               DrawRoundButton(&ButtonRound2);
               PORTC_OUT.B2 = 1;
               Delay_ms(500);
               PORTC_OUT.B2 = 0;
               }
       }
    //----------------------------------------------------------------------------
    //NO BUTTON PRESSED
    //----------------------------------------------------------------------------
         if ((RightButtonPressed == 0 && LeftButtonPressed ==0))
         {
            if  (count==0)
            {
             UARTE0_Write(NoButtonSendPressed);
             count=1;
             }
         }

    if(UARTE0_Data_Ready()== 1) {
    //----------------------------------------------------------------------------
    //Traffic light control
    //----------------------------------------------------------------------------
         UartBuffer = UARTE0_Read();

         //Femur indicator
         if (UartBuffer == FemurGreen){
             Circle1.Color = CL_LIME;
             DrawCircle(&Circle1);
             //UARTE0_Write_Text("Femur GREEN\r");
         }
         if (UartBuffer == FemurYellow){
             Circle1.Color = CL_YELLOW;
             DrawCircle(&Circle1);
             //UARTE0_Write_Text("Femur YELLOW\r");
         }
         if (UartBuffer == FemurRed){
             Circle1.Color = CL_RED;
             DrawCircle(&Circle1);
             //UARTE0_Write_Text("Femur RED\r");
             Sound_Play(3400, 400);           //sound allert
         }

         //Tibia indicator
         if (UartBuffer == TibiaGreen){
             Circle2.Color = CL_LIME;
             DrawCircle(&Circle2);
             //UARTE0_Write_Text("Tibia GREEN\r");
         }
         if (UartBuffer == TibiaYellow){
             Circle2.Color = CL_YELLOW;
             DrawCircle(&Circle2);
             //UARTE0_Write_Text("Tibia YELLOW\r");
         }
         if (UartBuffer == TibiaRed){
             Circle2.Color = CL_RED;
             DrawCircle(&Circle2);
             //UARTE0_Write_Text("Femur RED\r");
             Sound_Play(3400, 400);           //sound allert
         }

         //Pointer indicator

         if (UartBuffer == PointerGreen){
             Circle3.Color = CL_LIME;
             DrawCircle(&Circle3);
             //UARTE0_Write_Text("Pointer GREEN\r");
         }
         if (UartBuffer == PointerYellow){
             Circle3.Color = CL_YELLOW;
             DrawCircle(&Circle3);
             //UARTE0_Write_Text("Pointer YELLOW\r");
         }
         if (UartBuffer == PointerRed){
             Circle3.Color = CL_RED;
             DrawCircle(&Circle3);
             //UARTE0_Write_Text("Pointer RED\r");
             Sound_Play(3400, 400);             //sound allert
         }
      //--------------------------------------------------------------------------
      //Updating Screen, instrument display
      //--------------------------------------------------------------------------

               if (UartBuffer == NextInstrument)
               {
                   if (j <((sizeof(Images)/sizeof(Images[0])-1)))         // With
18 images, last Image has index j=17.
                     j++;
//sizeof(Images)/sizeof(Images[0]) returns length of array Image
                   if (j == 1)
                   {
                    Image1.Visible = 0;              //set invisible backgroung
Image so that the black background will be visible
                    DrawImage(&Image1);
                    DrawScreen(&Screen1);

                    Label4.Font_Color = CL_WHITE;     //turns white bottom labels
                    DrawLabel(&Label4);
                    Label5.Font_Color = CL_WHITE;
                    DrawLabel(&Label5);
                    Label6.Visible = 1;              //turns visible labels
Current and Next instrument
                    DrawLabel(&Label6);
                    Label7.Visible = 1;
                    DrawLabel(&Label7);

                    DrawRoundBox(&BoxRound1);        //redraws all elements
                    DrawCircle(&Circle1);
                    DrawCircle(&Circle2);
                    DrawCircle(&Circle3);
                    DrawLabel(&Label1);
```

```
                       DrawLabel(&Label2);
                       DrawLabel(&Label3);
                       DrawImage(&Image20);
                       DrawImage(&Image21);
                       DrawImage(&Image22);
                       DrawRoundButton(&ButtonRound1);
                       DrawRoundButton(&ButtonRound2);
                     }
                     TFT_Ext_Image(61, 90, Images[j-1], 1);        //displays images
on defined position
                     TFT_Ext_Image(177, 90, Images[j], 1);

                  }

                  if (UartBuffer == PreviousInstrument)
                  {
                     if (j>0)
                        j--;
                     if (j == 0)
                     {
                       Image1.Visible = 1;                //set visible backgroung
Image. First screen
                       DrawImage(&Image1);
                       Label4.Font_Color = CL_BLACK;      //turns black bottom labels
                       DrawLabel(&Label4);
                       Label5.Font_Color = CL_BLACK;
                       DrawLabel(&Label5);

                       Label6.Visible = 0;                //turns invisible labels
Current and Next instrument
                       DrawLabel(&Label6);
                       Label7.Visible = 0;
                       DrawLabel(&Label7);

                       DrawRoundBox(&BoxRound1);          //redraws all elements
                       DrawCircle(&Circle1);
                       DrawCircle(&Circle2);
                       DrawCircle(&Circle3);
                       DrawLabel(&Label1);
                       DrawLabel(&Label2);
                       DrawLabel(&Label3);
                       DrawImage(&Image20);
                       DrawImage(&Image21);
                       DrawImage(&Image22);
                       DrawRoundButton(&ButtonRound1);
                       DrawRoundButton(&ButtonRound2);
                     }
                     else
                     {
                       TFT_Ext_Image(61, 90, Images[j-1], 1);        //displays images
on defined position
                       TFT_Ext_Image(177, 90, Images[j], 1);
                     }
                  }
          }
      }
}

//---------------------------------------------------------------------------------
//ISR- interrupt service routines, timer interrupts
//---------------------------------------------------------------------------------
void Timer1Overflow_ISR() org IVT_ADDR_TCC0_OVF {
      InterruptCountLeft++;
}
void Timer2Overflow_ISR() org IVT_ADDR_TCC1_OVF {
      InterruptCountRight++;
}
```

Appendix K RemoteControl_1_2_FINAL_events_code.c

```
#include "RemoteControl_1_2_FINAL_objects.h"
#include "RemoteControl_1_2_FINAL_resources.h"

//--------------------- User code ---------------------//

 extern int InterruptCountLeft;
 extern int InterruptCountRight;
 extern char LeftButtonPressed;
 extern char RightButtonPressed;
 extern int count;
//----------------- End of User code -----------------//

// Event Handlers
void ButtonRound1Down() {
    ButtonRound1.Gradient_End_Color = 0x80FF;
    DrawRoundButton(&ButtonRound1);

    LeftButtonPressed = 1;
    PORTC_OUT.B2 = 1;
    Delay_ms(100);
    PORTC_OUT.B2 = 0;

    Timer_Interrupt_Enable(&TCC0);   // Enable TimerC0 interrupt
    InterruptCountLeft = 0;
}

void ButtonRound2Down() {
    ButtonRound2.Gradient_End_Color = 0x80FF;
    DrawRoundButton(&ButtonRound2);

    RightButtonPressed = 1;
    PORTC_OUT.B2 = 1;
    Delay_ms(100);
    PORTC_OUT.B2 = 0;

    Timer_Interrupt_Enable(&TCC1);   // Enable TimerC0 interrupt
    InterruptCountRight = 0;
}

void ButtonRound1Up() {
    ButtonRound1.Gradient_End_Color = CL_SILVER;
    DrawRoundButton(&ButtonRound1);
    Timer_Interrupt_Disable(&TCC0);   // Disable TimerC0 interrupt

    LeftButtonPressed = 0;
    count=0;
}

void ButtonRound2Up() {
    ButtonRound2.Gradient_End_Color = CL_SILVER;
    DrawRoundButton(&ButtonRound2);
    Timer_Interrupt_Disable(&TCC1);   // Disable TimerC0 interrupt

    RightButtonPressed = 0;
    count=0;
}

void Image1Down() {

}

void Image1Up() {   //in case that the button is not released in the button region,
the same ButtonUp event must be performed in the whole screen
```

```
    if(LeftButtonPressed ==1)
    {
       ButtonRound1.Gradient_End_Color = CL_SILVER;
       DrawRoundButton(&ButtonRound1);
       Timer_Interrupt_Disable(&TCC0);   // Disable TimerC0 interrupt
       //InterruptCountLeft = 0;
       LeftButtonPressed = 0;
       count=0;
    }
    if(RightButtonPressed ==1)
    {
       ButtonRound2.Gradient_End_Color = CL_SILVER;
       DrawRoundButton(&ButtonRound2);
       Timer_Interrupt_Disable(&TCC1);   // Disable TimerC0 interrupt
       //InterruptCountRight = 0;
       RightButtonPressed = 0;
       count=0;
    }
}
```

Appendix L Content of CD

CD contains following items:

- ***Thesis (folder)***
  Contains electronic version of this thesis

- ***Camera-Wireless PCB (folder)***
  Contains schematics and PCB Altium Designer files; code for Camera-PC interface
- ***Remote Controller (folder)***
  Contains C-files and H-files of created microcontroller application