Wrocław University
of Science and Technology

Intermediate Project – Report

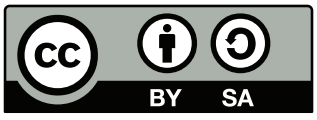
Automated data acquisition and control system of laboratory stand for testing distance sensors

Author:
Piotr Portasiak

Instructor:
Dr. Eng. Witold Paluszyński

Department of Cybernetics and Robotics, Faculty of Electronics
— Wroclaw University of Science and Technology —

February 6, 2019

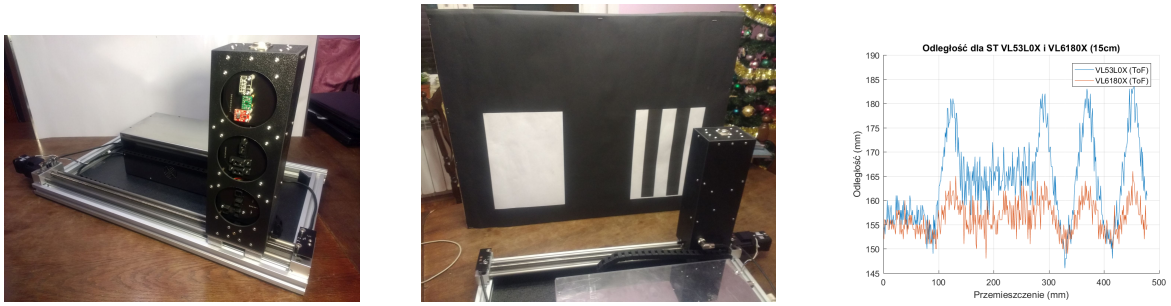


Abstract

The aim of the project was to research and develop a software infrastructure for control-measurement platform. The presented system consists of desktop and embedded application which are intended to work in collaboration in order to perform custom experiment scenarios predefined by user. The project outcomes allows to take advantage of the system for any complex device in which communication, control and measurement data acquisition with real-time constraints are required.

1 Project assumptions

The main purpose of the project was related to development of the desktop and embedded applications for the *Laboratory stand for testing distance sensors* [5]. The mentioned device is presented in Fig. 1. It allows to compare the operation of selected optical sensors depending of relative orientation, texture, color and distance to the observed objects surface. To perform various experiments, which can present the differences between sensors, it was necessary to build a unified measurement data acquisition and control system. It was assumed that the desktop and embedded applications have to work in collaboration to accomplish this.



(a) Device construction (b) Example of experiment (c) Plots of measurements

Figure 1: *Laboratory stand for testing distance sensors*

2 General overview of the software architecture

The general overview of the software architecture is presented in Fig. 2. It is divided into two main parts:

1. Desktop Application with Graphical User Interface (GUI), which communicates with measurement platform in real time in order to parametrize its operation, collect, store and visualize measurement data in a fully automated way. This part of the system works under control of *GNU/Linux* Ubuntu distribution which is fully isolated from user specific operating system settings with usage of *Docker*.

2. Embedded Application which receives and executes higher level tasks delegated from User Desktop Application in real-time. It consists of two independent applications for microcontrollers which are responsible for measurement head positioning and for sensor communication respectively. Both applications works under control of *CMSIS-RTOS* which is *FreeRTOS* wrapper supported by *ST* and *ARM* company.

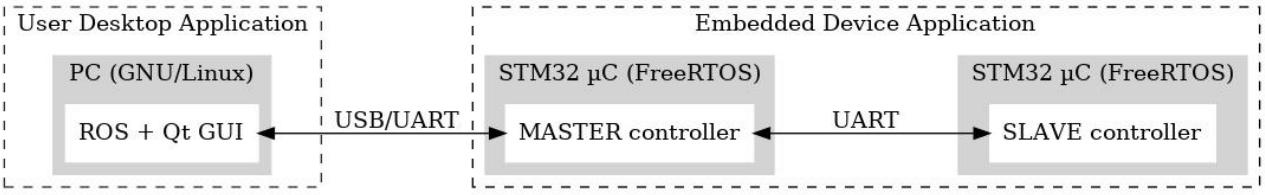


Figure 2: General overview of the software architecture.

3 Embedded Device Application

The embedded software for *Master* and *Slave* controllers were prepared using development tools provided by *ST* company for *STM32* family of microcontrollers. Mainly the following two main program were used:

- STM32CubeMX which is a graphical tool that simplifies configuration of microcontroller peripherals and the generation of the corresponding initialization C code,
- Atollic TrueSTUDIO which is a flexible and extensible development and debugging IDE for STM32 MCU. The program is based on the open standards which are *Eclipse* and *GDB* (GNU Debugger).

Both controllers work under control of *CMSIS-RTOS*. Some functionalities of both controller which requires simultaneous and non-blocking operation were delegated to the separated system tasks. Each node of the system is responsible of executing the corresponding tasks, which are presented in Fig. 3 and listed below:

- *Master* controller:
  - LED – control of LED indicators,
  - MANIP – control of manipulator position.
  - eSTOP – emergency stop of manipulator.
  - eLEFT – left endstop of manipulator,
  - eRIGHT – right endstop of manipulator,
  - RAMP – speed profiler of manipulator,
  - TX1 – message transmitter to PC,
  - RX1 – message receiver from PC,
  - TX2 – message transmitter to SLAVE,
  - RX2 – message receiver from SLAVE.
- *Slave* controller:
  - PTIR1 – reflectance sensor (1),
  - PTIR2 – reflectance sensor (2),
  - PTIR3 – reflectance sensor (3),
  - SHARP1 – PSD sensor (1),
  - SHARP2 – PSD sensor (2),
  - VL53L0X – ToF sensor (1),
  - VL6180X – ALS and ToF sensor (2),
  - ADPS9960 – color sensor (1),
  - TX1 – message transmitter to MASTER,
  - RX1 – message receiver from MASTER.

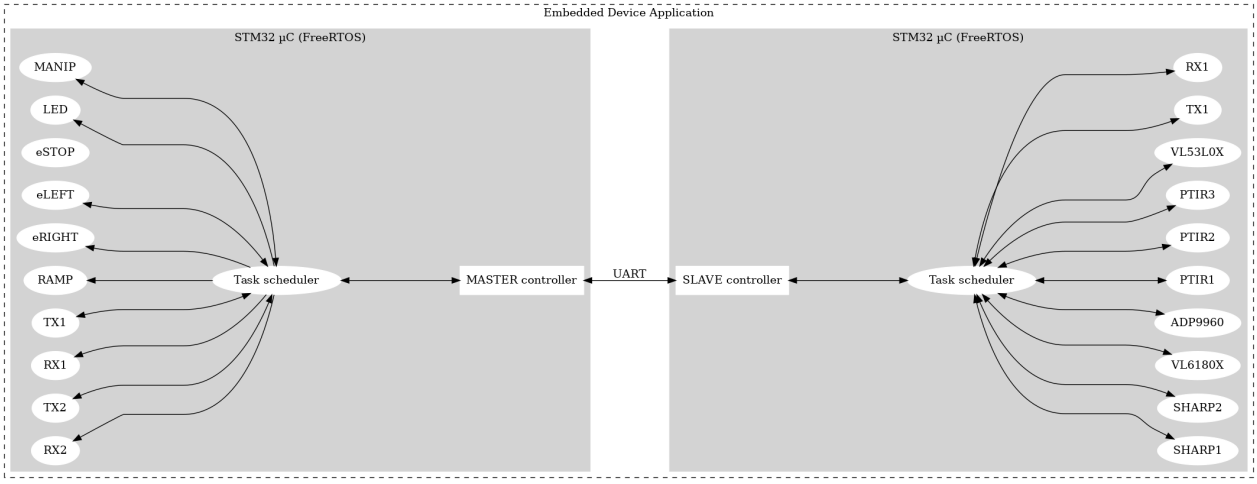


Figure 3: General overview of the embedded system functionalities and task schedulers.

4 Communication stack – implementation

In the Fig. 4 general overview of communication process is shown. Both microcontrollers are connected through common communication channel realized with asynchronous UART module. The communication between *PC* and *SLAVE* controller is realized through *MASTER* module. The data transmission is internally buffered either for transmitter (Tx) and receiver (Rx) on each side of communication end device.

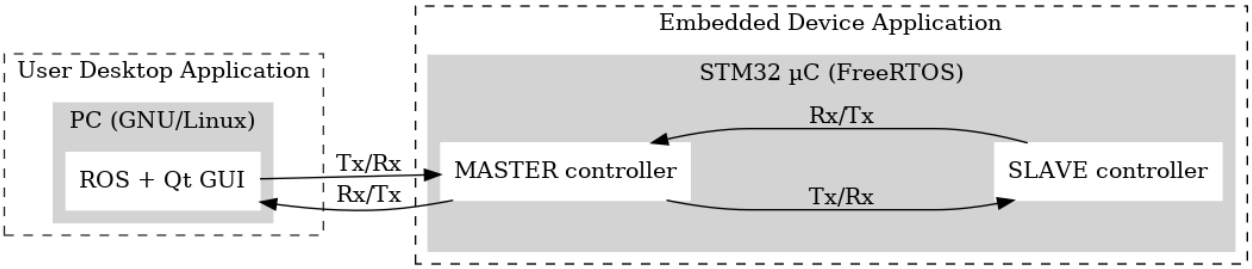
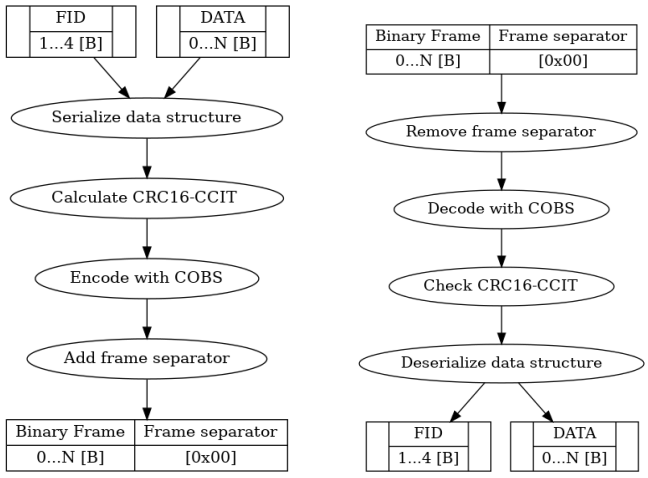


Figure 4: General overview of the communication process.

The general idea of the communication stack takes into account the following issues:

- data is transferred using binary mode, which were investigated to be more efficient than ASCII mode,
- data structures are serialized using *Google Protocol Buffers* for Desktop Application and *nanopb* for Embedded Application,
- packets are framed and stuffed using *Consistent Overhead Byte Stuffing* (COBS) which eliminates zero valued bytes and preserve this byte for frame synchronization,
- *CRC16-CCITT* checksum is used to avoid corrupted frames.

All of the mechanism presented above makes the communication process simply and robust. Simply in sense of frame parsing because the serialization is independent of byte ordering in frames (endianness) and data structures can be directly packed to frame. There is also no need to reserve specific byte for Start Of Frame (SOF) marker because each frame is always separated with zero byte. Communication robustness is achieved using COBS encoding and CRC16 checksum. The message structure and processing steps in order to encode and decode frame for each side of communication channel are presented in the Fig. 5.



(a) Frame encoding (b) Frame decoding

Figure 5: Message structure and processing steps for frame encode/decode.

5 User Desktop Application – *Telemetry*

The User Desktop Application was created using *Qt* which is a cross-platform application development framework for desktop devices. Supported platforms include Linux, OS X, Windows and many others. Thanks to this feature, the created application can be compiled and released for different operating systems with preserving the same behaviour and functionalities. In the Fig. 6 the main parts of the user interface are presented. It consists of the following sections:

1. Connection trigger buttons.
2. Connection settings dialog window.
3. Manual control of the position of manipulator.
4. Automated manipulator control with storing measurements to file.
5. Received messages logger.
6. Visualization of manipulator positioning.
7. Tabbed menu for switching between different plots windows.
8. Credits window.

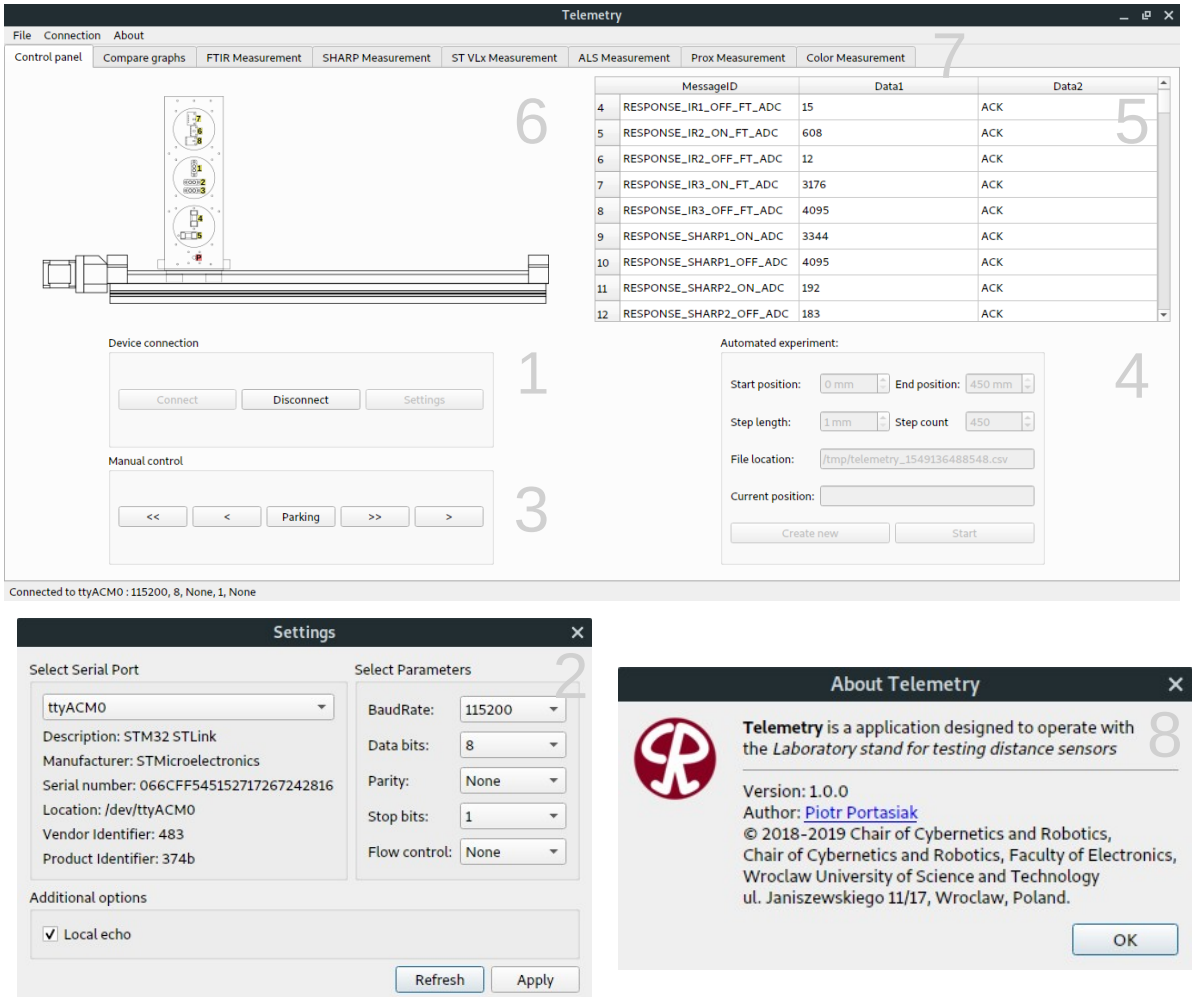
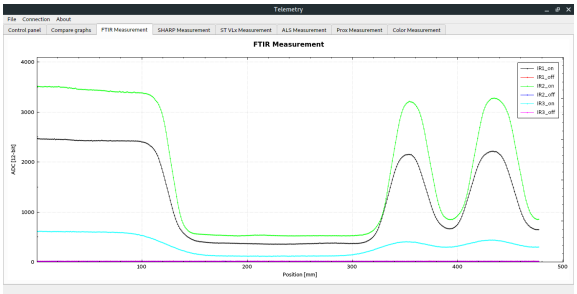
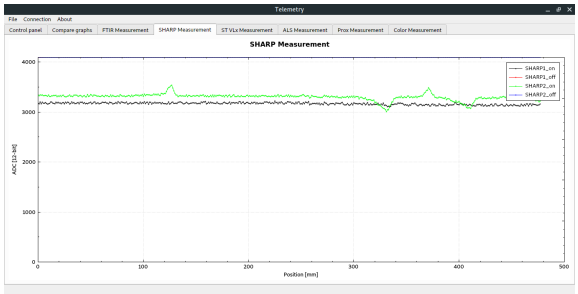


Figure 6: *Telemetry* – main parts of the user interface.

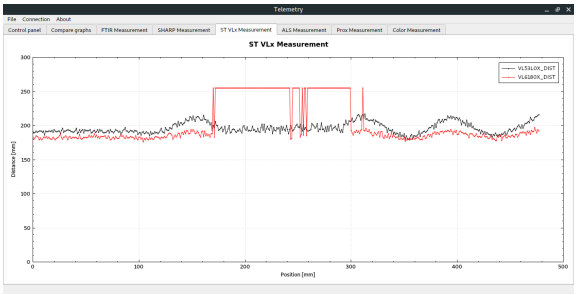
The visualization of measurements data is realized using *QCustomPlots* library which allows to create real-time plots. Received measurement data is stored and appropriate plots are updated automatically in six individual windows (also in one common graph for all plots). Exemplary plots are presented in the Fig. 7.



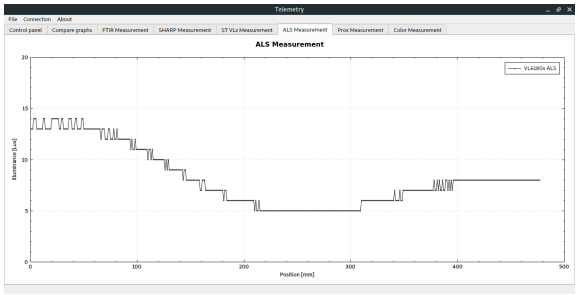
(a) Plots of voltage measurements from reflectance sensors



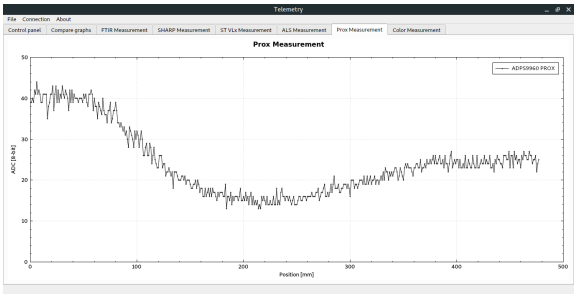
(b) Plots of voltage measurements from PSD sensors



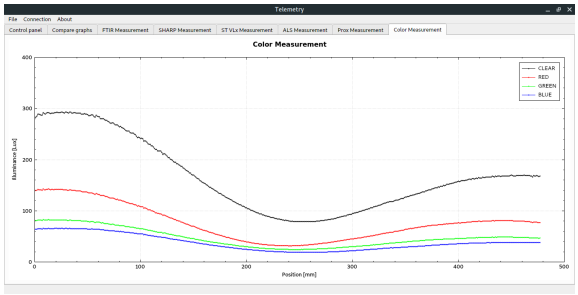
(c) Plots of distance measurements from ToF sensors



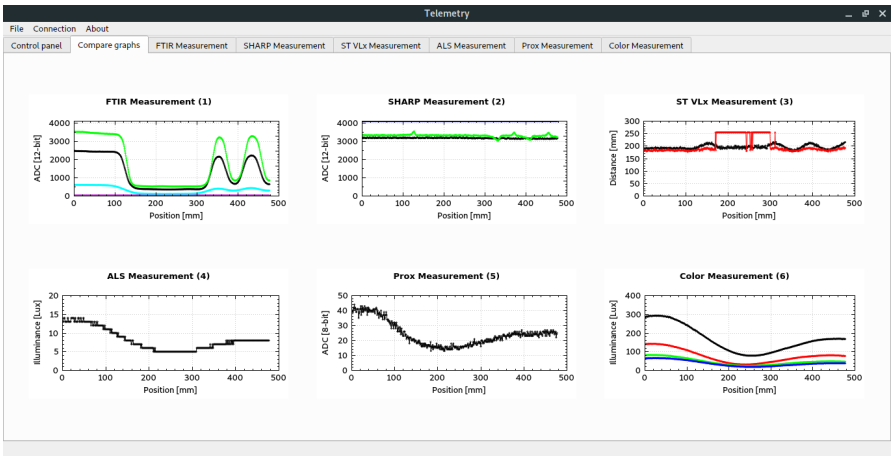
(d) Plot of ambient light intensity from ALS sensor



(e) Plots of proximity measurements from RGB sensor



(f) Plots of intensity of light color from RGB sensor



(g) Common graph for all measurement plots

Figure 7: *Telemetry* – measurement windows

6 Integration with Robot Operation System (ROS)

In the last phase of the project the interface for Robot Operation System (ROS) was prepared. It was possible with usage of ROS packages such as *rqt* which is a Qt-based framework for GUI development. In this way the Qt mechanism of signals-slots along with the ROS publish-subscribe specific communication method can be used. In order to establish connection with the measurement platform the special node was implemented using *Python* to encode and decode system messages and publish measurements data to specific topics which next are plotted in the corresponding windows. Prepared user interface is shown in the Fig. 8.

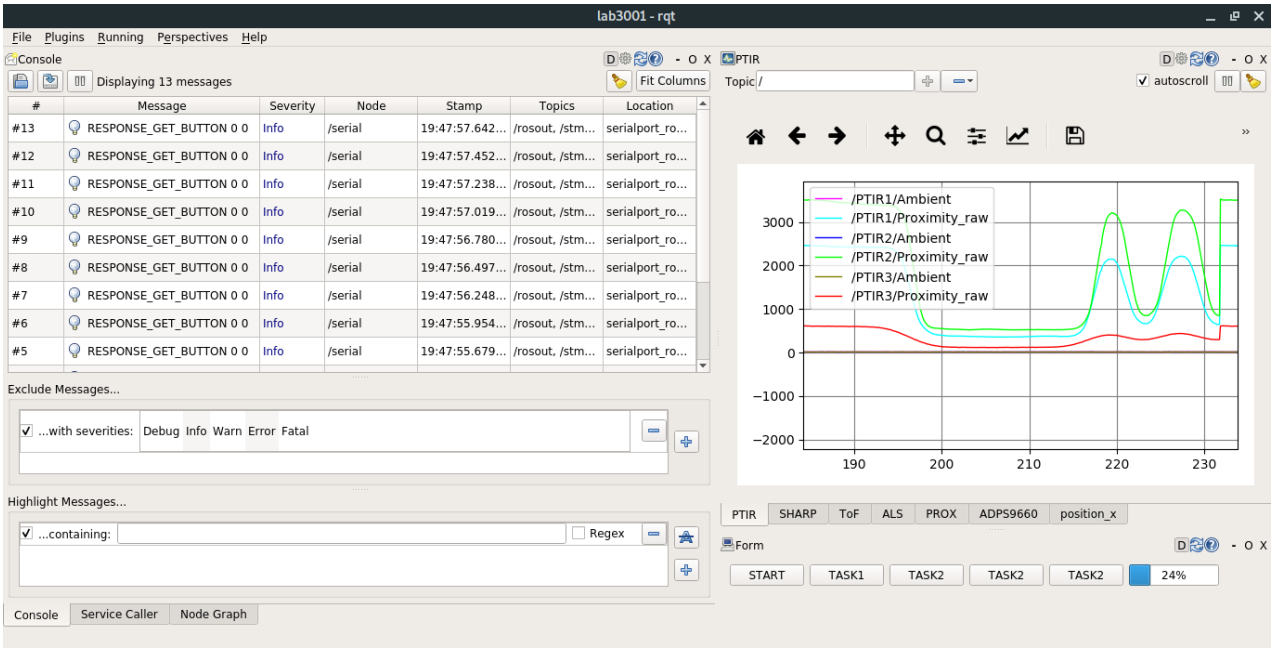


Figure 8: Integration of user interface with ROS.

7 Project summary

It can be stated that project outcomes satisfy all the initial assumptions and even more. The main difficulties related to communication, control and measurements in real-time constraints have been eliminated. The usage of ROS deliver to the system more flexibility and many additional possibilities in the future development. The project results allows to take advantage of the system for any complex application with real-time constraints.

8 Software and licenses

There is no proprietary software involved in this project. Most programs used for development is free:

1. STM32CubeMX – free to commercial use,
2. Atollic TrueSTUDIO. – free to commercial use,
3. CMSIS-RTOS, Docker – Apache 2.0 License,
4. Qt – free to non-commercial use based on LGPL (v. 2.1), GPL (v. 3.0),
5. ROS – free to commercial use, BSD license,
6. Ubuntu 18.04 – free to commercial use, GNU GPL license,
7. Doxygen – free to commercial use, GPL license.

References

[1] Sanford Friedenthal, Alan Moore, Rick Steiner. *A practical guide to SysML*. Elsevier, 2012.

[2] Steve Mackay, John Park. *Practical Data Acquisition for Instrumentation and Control Systems*. Elsevier, 2003.

[3] Guillaume Lazar, Robin Penea. *Mastering Qt5 – Second Edition*. Packt Publishing, 2018.

[4] Richard Barry. *Mastering the FreeRTOS Real Time Kernel*. Real Time Engineers Ltd. 2016.

[5] Piotr Portasiak. Engineer's thesis: *Laboratory stand for testing distance sensors*. Wroclaw University of Science and Technology, 2018. Available on-line at: http://panamint.kcir.pwr.edu.pl/ pportasi/inz/praca_dyplomowa.pdf [Accessed 6 Feb 2018].