This is a preprint of an article submitted to 6th International Conference on Modelling and Simulation for Autonomous Systems held in Palermo, Italy, on 29–30 October 2019. The final version is available under DOI 10.1007/978-3-030-43890-6_21 and online: https://link.springer.com/chapter/10.1007/978-3-030-43890-6_21.

Practical comparison of communication technologies suitable for field robotics

Matous Hybl, Tomas Jilek, and Ludek Zalud

DCI FEEC, Brno University of Technology, Brno, the Czech Republic
CEITEC, Brno University of Technology, Brno, the Czech Republic
xhyblm00@stud.feec.vutbr.cz,{tomas.jilek,ludek.zalud}@ceitec.vutbr.cz
http://www.feec.vutbr.cz, http://www.ceitec.vutbr.cz

Abstract. The aim of this paper is to describe development of an evaluation system for practical comparison of wireless communication technologies used in field robotics. The evaluation system is capable of measuring data rate, drop rate and latency in both communication directions. The evaluation system closely simulates communication of a teleoperated field robot with its base station - which is a highly asymmetric communication link, because there is a high data rate video feed transmission from the robot to the base station.
This paper describes specifics of wireless communication links used in field robotics and considerations that need to be taken into account when developing such system. The wireless link evaluation system is described from hardware and software point of view.

Keywords: Field robotics, Wireless communication technology, Evaluation system, 802.11, Mobile robot.

1 Introduction & Motivation

Communication is one of the key areas in mobile and especially field robotics where intensive research needs to be conducted because success of teleoperated field robotics missions depends on stable, long range and low latency communication. The nature of field robotics missions determines that the communication link must be wireless which brings significant challenges concerned with latency and most importantly range.

This article describes an approach to building an evaluation tool for communication links used in field robotics. It is capable of measuring latency, loss rate, reordering and with suitable communication hardware also SNR and RSSI. The evaluation system works by simulating traffic that is usually present in teleoperated field robotics missions - there are control data transmitted from the operator to the robot and video feed with telemetry data transmitted in the opposite direction - from the robot to the operator. This way the results are better than with just measuring data rates with random data as the measurement process is an approximation of a real life mission. First version of this evaluation

system was described in [1] where it was successfully tested while evaluating 2.4 GHz and 5 GHz 802.11 communication links.

The motivation behind development of such evaluation system was the ability of rapid testing of communication interfaces and therefore being able to deliver the best wireless communication performance in field robotics missions which are primary research goal of our research group. The main goal is to greatly improve communication link for our research group's Orpheus series robots [2], [3], [4].

2 Measuring communication link properties

The goal of this section is to provide insight into requirements needed for implementation of the communication link evaluation system. First, specifics of communication in mobile robotics are described. Analysis of video feed communication model is done in order to closely simulate video feed traffic. For latency measurement time synchronization is required which is also described here.

2.1 Specifics of communication in field robotics

Data traffic in teleoperated field robots consists of two types of UDP datagrams - video feed coming from the robot to the base station and control and telemetry data that are transmitted in both directions. A schematic diagram showing the traffic can be seen in the figure 1. As can be seen in the figure, the base station sends control data at frequency of about 30 Hz and the data has a length of about 80 B, while the robot sends video feed with data rate of approximately 800 kbps and telemetry data which are responses to the control data therefore have about the same frequency of 30 Hz but their length is about 120 B.

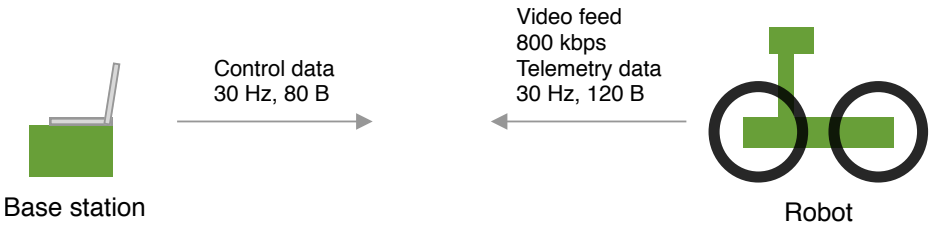

Fig. 1. A graphical representation of communication between the base station and the robot

2.2 Analysis of video feed data

In order to closely simulate the video feed traffic the video feed must be captured and analyzed. The video feed consists of UDP datagrams that contain h.264 or

h.265 encoded video. This codec effectively compresses small changes in the picture resulting in low data rate, while full frame changes such as moving with the camera result in high data rate. Therefore to accurately analyze the video feed, the camera needs to be moving during the capture to generate high data rate video feed. The data required to simulate the feed are timestamps and lengths of the datagrams.

The video feed was captured using Wireshark by applying filter on the interface that is used to connect to the camera. The applied filter was `udp and ip.src == cameraIP and ip.dst == computerIP` where `cameraIP` is the IP address of the camera and `computerIP` is the IP address of the interface that is used to connect to the camera. The measured data can be then exported in the Wireshark using `File -> Export Packet Dissections -> As CSV...`.

Captured datagram information were then analyzed using MATLAB. First the CSV file was loaded to a matrix and timestamps alongside with datagram lengths were extracted. The data can be seen in the figure 2. As can be seen the majority of captured datagrams have 1400 B length. A window with length of one second containing a lot of datagrams with 1400 B length (indicating high data rate) was selected. Datagrams contained in the selected second are shown in the figure 3.

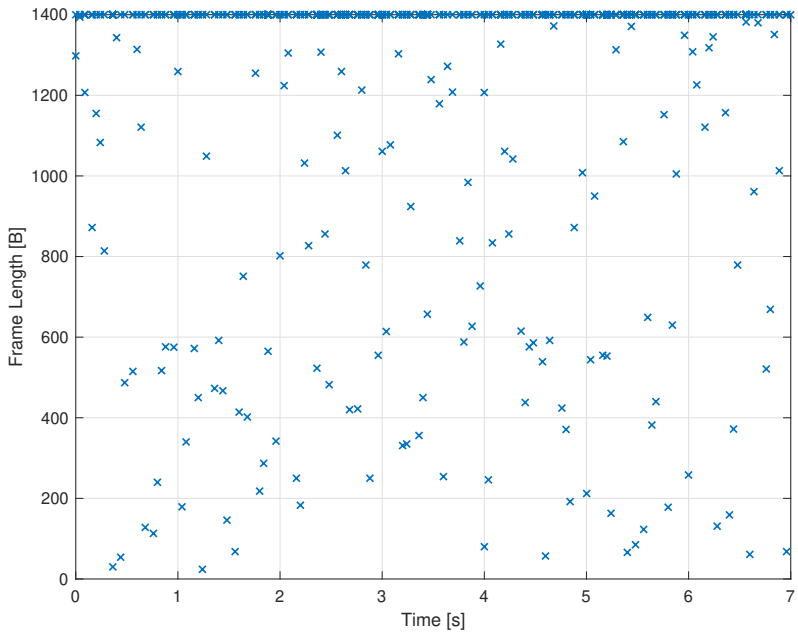

Fig. 2. Captured video feed datagram length in time

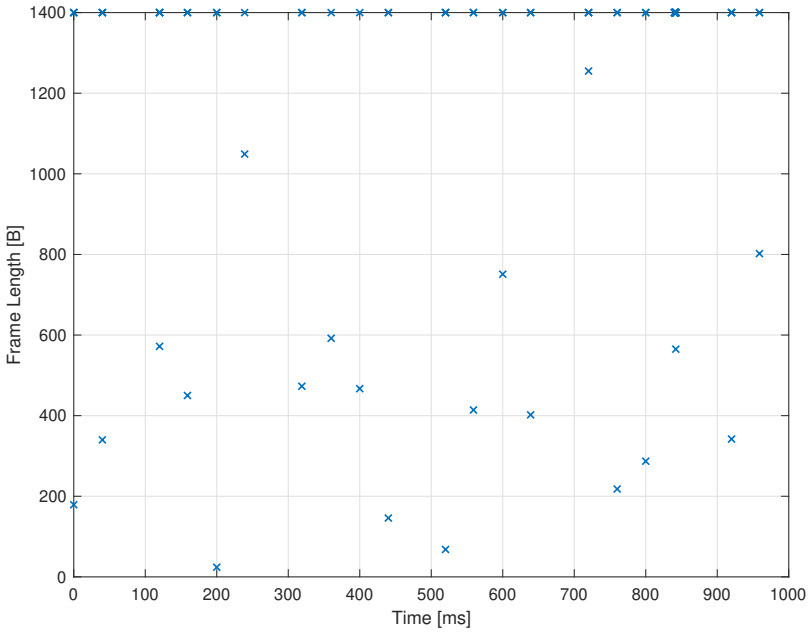

Fig. 3. Selected window of the the video feed datagram capture

The selected datagrams then needed to be transformed into data that could be used by the evaluation system. That means that timestamps in seconds needed to be transformed to milliseconds. Observing the timestamps showed that the data is grouped into groups of four to eight datagrams. Furthermore by observing the data it can be seen that the timestamps can be rounded to 10 ms resolution, which is useful for optimizing the evaluation system sending algorithm. Timestamps and datagram lengths were then exported so that they could be used in the evaluation system.

2.3 Time synchronization

Latency measurement requires time synchronization between the two communication simulators, which can be achieved using PTP (precision time protocol - IEEE 1588) [5]. This protocol is implemented on Linux using the `ptpd` daemon. In case of this measurement system, `ptdp` was installed on both of the Raspberry Pis. The robot simulating Raspberry Pi then serves as a master in the PTP and is configured using command `sudo ptpd -M -i eth0` while the base simulating Raspberry Pi serves as a PTP slave and is configured using `sudo ptpd -s -i eth0`. The slave automatically finds the master and adjusts time accordingly.

The previously described setup works only when the two parts of the evaluation system are present on the same network and the connection is done using a transparent radio bridge. In the case that there is not a transparent link and the two parts are on separate networks another parameter needs to be passed o the `ptpd` daemon command. The robot simulating part needs to be started with command `sudo ptpd -M -i eth0 -u BASE_STATION_SIMULATOR_ADDRESS` and the base station simulator with command `sudo ptpd -s -i eth0 -u ROBOT_SIMULATOR_ADDRESS`.

3 System for evaluation of communication links

Our proposed and developed evaluation system for communication links works by closely simulating traffic generated by the robot and the base station, therefore provides accurate results and shows the behavior of the link.

There are two parts of the evaluation system - robot and base station simulator - which simulate traffic generated by the robot and the base station. These parts consist of a Raspberry Pi running custom software. To control the evaluation system another program is used. The robot simulator is supposed to be mobile, therefore it was equipped with a 3 cell Li-Poly battery and a step-down converter.

In order to achieve best results, the robot simulating part of the evaluation system has to be placed in certain height above the ground in order to avoid radio waves being absorbed by the ground. For some experiments it was therefore placed on a cardboard box with height of 490 mm. The base simulating part of the measurement system was placed on a table in a designated spot. Both parts of the evaluation system can be seen during an experimental evaluation with MikroTik router based communication link in the figures below.

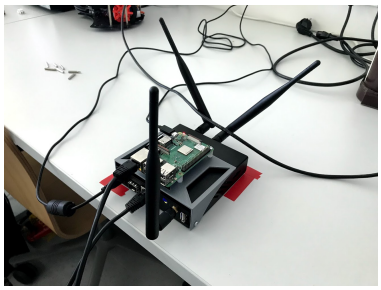

Fig. 4. Base simulator hardware placed on a designated spot on a table

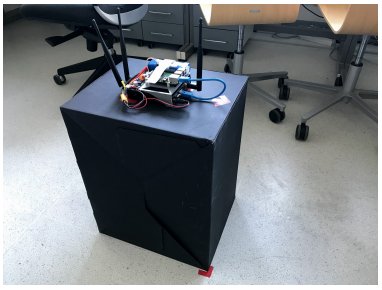

Fig. 5. Robot simulator hardware placed on a box used to maintain altitude

3.1 Communication simulating software

Communication simulating program was designed to simulate the communication model of the robot. The core of the program is based on an epoll based scheduler that schedules sending and receiving of UDP datagrams to simulate traffic.

The program is meant to be run all the time. It always waits for an incoming TCP connection that begins the evaluation. Once the evaluation begins, datagrams are sent periodically and received datagrams are parsed and analyzed. The evaluation runs for predetermined number of seconds and saves result of the evaluation for each second. After the evaluation is complete the resulting data are sent to the TCP connection that started the evaluation. A simple diagram showing communication between the programs is shown in the figure 6.

Implementation-wise there is only one program that can be switched to act as a robot simulator a base station simulator using command line arguments. The program is also capable of communication with MikroTik routers utlizing MikroTik API [6] in order to read RSSI and SNR values, address of the router is also specified using a command line argument.

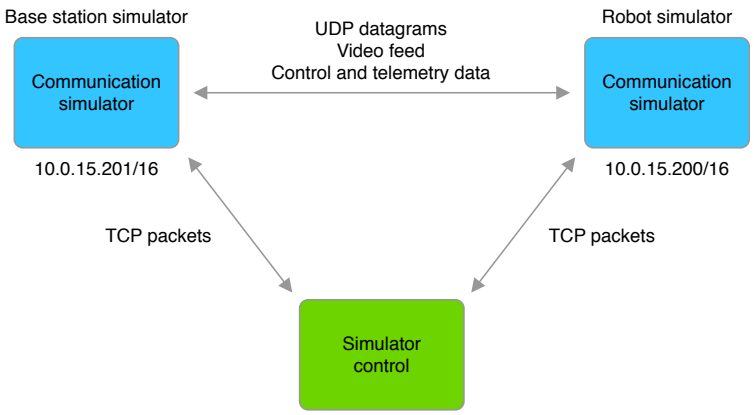

Fig. 6. Diagram showing communication between programs in the measurement system

Program binaries are installed on the Raspberry Pis using a Makefile generated by CMake. After installation the program can be launched by calling `commsim -a BASE_STATION_SIMULATOR_ADDRESS -m MIKROTIK_ADDRESS` or `commsim -b -a ROBOT_SIMULATOR_ADDRESS -m MIKROTIK_ADDRESS` depending on whether the program should simulate traffic from the robot or from the base station. The `MIKROTIK_ADDRESS` is an optional parameter and the -b parameter specifies that the simulation program should act as a base station simulator.

Simulating control and telemetry traffic Control and telemetry traffic consists of datagrams with average length of 80 bytes in the direction to robot and of

120 bytes in the direction from robot. Datagrams in both directions are sent with frequency of 30 Hz therefore need to be sent every 33 ms. Structure of datagrams sent by the evaluation system is shown in the figure 7. As can be seen in the figure, the first byte is a header with a predefined single byte constant `0x10` that can be used to ignore packets that are not sent by the evaluation system. Then number of the second of the evaluation is sent as a single byte to identify datagrams that are received after the second concludes. Single byte sequence number follows the second that is used to identify reordered datagrams and finally there are eight bytes containing timestamp in milliseconds since 1.1.1970 that is used to calculate the latency. The rest of the datagram is filled with pseudo random data.

Fig. 7. Structure of a packet used for simulating control and telemetry traffic

Simulating video feed traffic Video feed capture and analysis in the section 2.2 showed that video feed datagrams are sent in batches irregularly and the sending loop can be optimized by running every 10 ms. First on the program launch datagram lengths obtained by the analysis are grouped to batches according to the timestamp. Then in every sending loop iteration the program checks if any datagrams should be sent and sends them if needed. Datagrams are then received and analyzed similarly to the control and telemetry datagrams.

The structure of the datagrams is similar to the one described in the section 3.1. However, there are some minor changes to it. Firstly the header byte contains `0x20` constant. Secondly apart from sending second, millisecond is also sent in two bytes. In the evaluation of telemetry and control traffic, the millisecond is currently not sent but could be used to further improve drop rate and datagram reorder analysis. The datagram structure is shown in the figure 8.

Fig. 8. Structure of a datagram used for simulating video feed traffic

Measuring latency Synchronization of time described in the section 2.3 alongside with timestamps being send in the datagram is used to measure latency

between sending and receiving datagrams. The calculation then requires only to subtract time received in the datagram from current time in milliseconds. To avoid for integer overflow errors caused by imprecise time synchronization it is vital that the smaller time is always subtracted from the greater time. After calculation of latency, it is saved and maximal and minimal latency are calculated. When evaluation is completed in the second, average latency is computed according to the equation 1 as well as standard deviation of the latency according to the equation 2. Code used for latency computation is shown in the listing 1.1.

$$\bar{t} = \frac{1}{N} \sum_{i=1}^{N} t_i \tag{1}$$

where $\bar{t}$ is average latency, t is vector of measured latencies and N is number of measured latencies

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (t_i - \bar{t})^2} \tag{2}$$

where $\bar{t}$ is average latency, t is vector of measured latencies and N is number of measured latencies

```
1  uint64_t time = millis();
2  uint64_t receivedTime = 0;
3  memcpy(&receivedTime, datagram.data + 5, 8)
4
5  uint64_t latency = 0;
6  if (time > receivedTime) {
7    latency = time - receivedTime;
8  } else {
9    latency = receivedTime - time;
10 }
11
12 if (latency > maxLatency) {
13   maxLatency = (uint16_t) latency;
14 }
15 if (latency < minLatency) {
16   minLatency = (uint16_t) latency;
17 }
18
19 latencies[latencyIndex] = (uint16_t) latency;
20 latencyIndex++;
```

Listing 1.1. Calculating latency

Measuring datagram reordering and loss There are two mechanisms used to detect delayed datagrams. Firstly, the second in which the datagrams were

sent must be the same as the second in which they were received. If that is not the case, then the datagrams are counted. Also if sequence number of the received frame is lower than sequence number of previously received datagram, the datagram is counted. Algorithm used for delayed datagram detection is shown in the listing 1.2.

```
1  uint8_t frameSecond = datagram.frame[1];
2
3  if (frameSecond != state.second) {
4    delayedPacketsFromLastSeconds++;
5    return;
6  }
7
8  if (newFrameNumber < lastReceivedFrameNumber) {
9    delayedPackets++;
10 } else {
11   lastReceivedFrameNumber = newFrameNumber;
12 }
```

Listing 1.2. Detecting delayed datagram

Lost datagrams are calculated from the number of sent datagrams that is hardcoded to the program by subtracting the number of received datagrams from the current second.

3.2 Evaluation system control and data processing

Evaluation is started using the evaluation control program. The program first starts the measurement by sending a predefined TCP packet to the simulator programs and then waits until the simulator programs send data back over the TCP socket. Once the measurement results are sent from both of the simulators the results are saved in CSV format. An example of calling the program is `csc -r ROBOT_SIMULATOR_ADDRESS -b BASE_STATION_SIMULATOR_ADDRESS -f OUTPUT_FILE`. An example of the output file contents is shown in the listing 1.3. Similarly to the communication simulation program the control program binary is installed using a Makefile generated by CMake.

```
1  robot
2  sentPackets,receivedPackets, ...
3  30,27,0,0,29.12,41.92,3,155,80,0,0,0,0,0,1000,0,-89,14
4  30,30,0,0,8.29,7.32,2,26,80,0,0,0,0,0,1000,0,-89,14
5  ...
6  base
7  sentPackets,receivedPackets, ...
8  30,25,0,0,24.5,22.95,0,77,0,70,0,0,41.76,37.45,
       3,152,-88,17
9  30,30,0,1,30.72,33.37,0,118,0,78,0,5,40.81,32.16,
       3,116,-88,17
```

Listing 1.3. Example of measurement system output file contents

3.3 Processing the measured data

Results provided by the evaluation system need to be further processed to provide meaningful data for experiments. The processing is performed using a function programmed in MATLAB. In each measurement there are data from 30 seconds. The data are averaged and maximal values are selected. The script outputs results to a `csv` file.

4 Sample results

The figures 9, 10 and 11 show sample data output from the evaluation system, more specifically comparison of data from two evaluations - of 2.4 GHz and 5 GHz 802.11 communication links. The evaluations were performed in four points with different distances and different numbers of obstacles.

The first figure - figure 9 shows comparison of average latencies. The second figure - figure 10 shows comparison of latency deviations. The third figure - figure 11 shows comparison of drop rates.

It can be seen that generally 5 GHz link performs worse than the 2.4 GHz one. There are some irregularities in the data that could most likely be avoided using repeated measurement and by measuring in areas where frequency spectrum on these frequencies is not used that much.

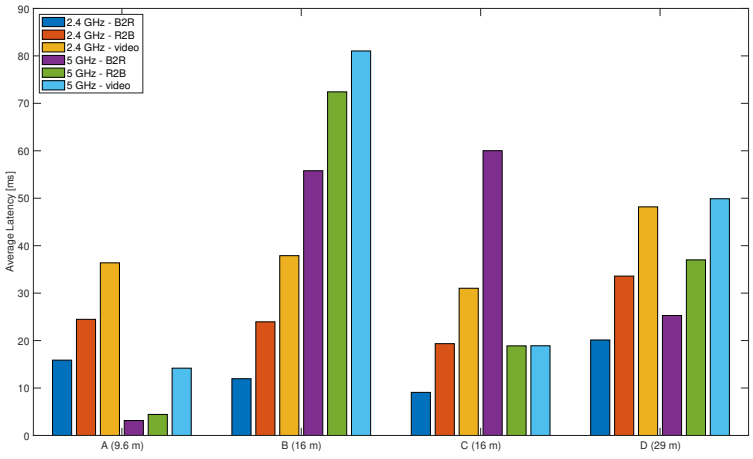

Fig. 9. Comparison of average latencies from evaluation of 2.4 GHz and 5 GHz 802.11 communication links

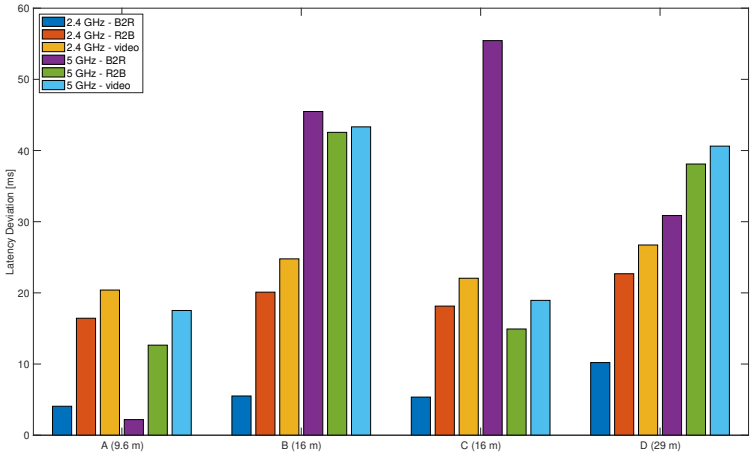

Fig. 10. Comparison of latency deviations from evaluation of 2.4 GHz and 5 GHz 802.11 communication links

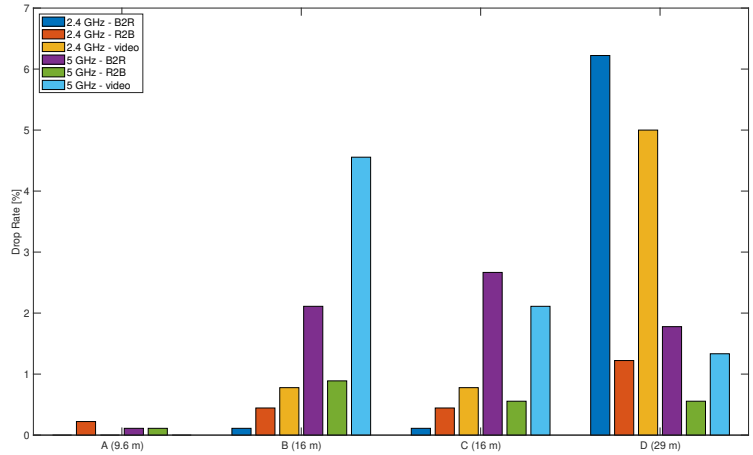

Fig. 11. Comparison of drop rates from evaluation of 2.4 GHz and 5 GHz 802.11 communication links

5 Further work

There are a few ways of further enhancing the evaluation system. The most significant of them is adding a touch screen display to the robot simulator Raspberry Pi, thus adding a graphical user interface. This would bring significant simplification to the usage of the evaluation system as the need for an external computer starting the simulation would be eliminated. Another advantage would be the possibility to show the measured data in real time and measure again in case of any fatal error. Showing the data in real time also means that the results processing would be done directly on the Raspberry Pi, therefore eliminating the need for external scripts for data processing, but there would have to be a way to upload the results to an external storage, such as a file server or an USB flash drive.

Another way of improving the evaluation system would be allowing for onboard network interface configuration. This proposes significant challenge as improper configuration causes the entire evaluation system to become unusable. This could also be configurable using the aforementioned graphical user interface.

Another problem that needs to be solved is simulation of a moving vehicle because there will be more noise and the demodulation might be problematic.

6 Conclusion

The aim of this thesis was to provide an insight into how to develop an evaluation system for communication links in field robotics. Specifics of communication model in field robotics was examined alongside with video feed data. Based on these principles an example evaluation system was developed. The evaluation system is very simple but has many drawbacks, mostly caused by complicated setup and usage because there are many steps to be done before the evaluation itself starts. Many of these drawbacks will hopefully be solved by implementing the graphical user interface mentioned in the section 5.

The evaluation system was tested in [1] where 2.4 GHz and 5 GHz 802.11 communication links were evaluated. Comparison of the two communication links proved that the 5 GHz link had lower range and higher latency. It was proven that the evaluation system works and it can be used for future comparisons of communication links and more importantly of their different configuration. The evaluation system targets not only 802.11 based systems but also any radio communication systems equipped with ethernet and TCP/IP stack (for example radios based on 433 MHz ISM frequencies). For comparison of the communication links it is also more important to tell which is better than to provide absolute values of measured data.

Acknowledgments. This work was supported by the European Regional Development Fund under the project Robotics 4 Industry 4.0 (reg. no. CZ.02.1.01 /0.0/0.0/15_003/0000470) and the completion of this paper was made possible

by the grant No. FEKT-S-17-4234 - "Industry 4.0 in automation and cybernetics" financially supported by the Internal science fund of Brno University of Technology.

References

1. Hybl, M.: Communication in Mobile Robotics. Brno University of Technology, Brno (2019)
2. Nejdl, L., Kudr, J., Ruttkay-Nedecky, B., Heger, Z., Zima, L., Zalud, L., Krizkova, S., Adam, V., Vaculovicova, M., Kizek, R.: Remote-Controlled Robotic Platform for Electrochemical Determination of Water Contaminated by Heavy Metal Ions. International Journal of Electrochemical Science 10(4), 3635–3643 (2015)
3. Zalud, L.: ARGOS - System for Heterogeneous Mobile Robot Teleoperation. IEEE/RSJ International Conference on Intelligent Robots and Systems, 211–216 (2006)
4. Zalud, L.: Orpheus – Universal Reconnaissance Teleoperated Robot. In: Nardi D., Riedmiller M., Sammut C., Santos-Victor J. (eds) RoboCup 2004: Robot Soccer World Cup VIII. RoboCup 2004. Lecture Notes in Computer Science, vol 3276. Springer, Berlin, Heidelberg
5. Precision Time Protocol, `http://www.ncbi.nlm.nih.gov`
6. API In CPP `https://wiki.mikrotik.com/wiki/API_In_CPP`