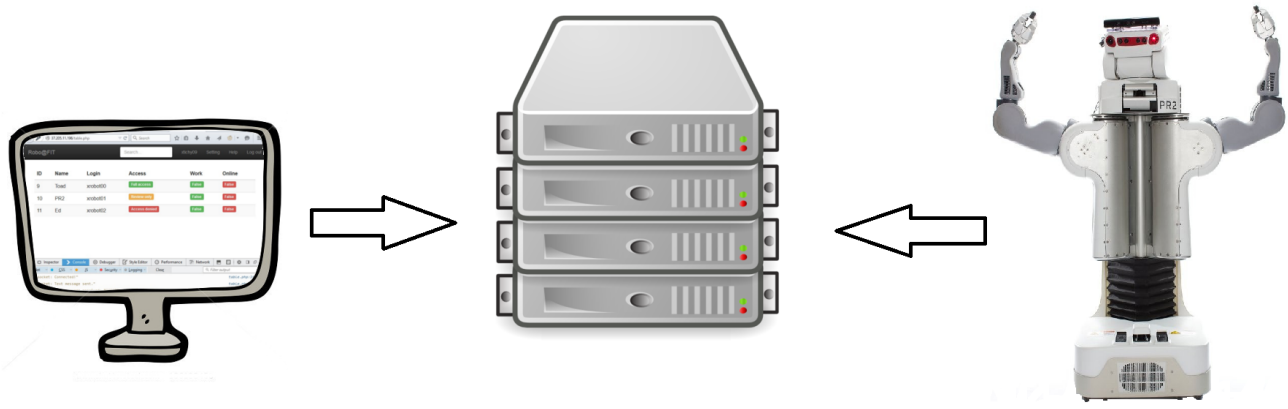
Excel@FIT 2015
http://excel.fit.vutbr.cz

Communication platform for management and control of robots

Andrej Tichý



Abstract
Nowadays, there are no cheap and easy robot management systems which would be able to provide fast and secure communication over a public network. This work deals with creating a communication platform which enables secure global communication between a client and a robot and ensures management of robots. To solve the problem I have designed a communication layer based on client-server architecture allowing non-ros applications to communicate with platforms based on ROS, using a specified protocol. When solving the problem I divided the application into three different subsections: a web client, a client for the robot and a server. The result of this work is the presentation of a functional system that can globally manage and communicate with the selected robot. The system provides an overview of the current status of selected robots. In already existing solutions I mentioned above the robot acts as a server in architecture, which is not an appropriate solution in practice. The main contribution of this work consists in exchanging roles, where robot acts as a client. Therefore it is possible to provide the robot with cheap computing power of server and thus to save valuable resources of the robot. One another benefit of this work is constructing a tool that allows global communication between the client and the robot and ensures the safety of this communication. It also represents the interface between non-ros applications and robots based on the ros platform. It provides advanced management of robots and clients, solves client's access rights to robots and helps to increase the efficiency of the robot.

Keywords: Remote control robot — Global communication — Robot Operating System

Supplementary Material: Video is not here yet — Code

*xtichy09@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

At this time when the need of having robots is increasing and there are tens of robots in many companies, a problem with confuseddnes a small efficiency of robots using occurs. The aim of the application beeing deveoped is to streamline working with robots and to give the user quick and easy overview of the status of individual robots. This will ultimately have an impact on reducing the cost spent on running of robots. Automating production lines, introduction of robotic employees or just purchasing one single robot are a quite big investment, so it is important that the robot is beeing used efficiently in order to arrange the shortest payback period for its owner.

For example the well known company Amazon completed the construction of its revolutionary warehouse "Amazon warehouse robots" [1], where they use around 15,000 robots for sorting, transfering and packaging of shipments. Using such a large number of robots it is necessary to have an automatized system of management and control.

According to the research [2] a robotic time comes, when intelligent robots will be common part of our everyday life. Industrial countries face the problem of ageing population, which is caused by the fact, that young families prefer career to children. One possible solution could be a robotic household providing babysitting or guarding property based on remote management and control.

The aim of my work was to design and implement a system, that will provide fast and secure communication between the web client and the robot as well as an overview of selected robots over a public network. It also should represent the interface between non-ros applications and systems based on ROS [3]. The problem is that there is no cheap and simple solution of the problem at this time, which could provide fast and secure communication over a public network.

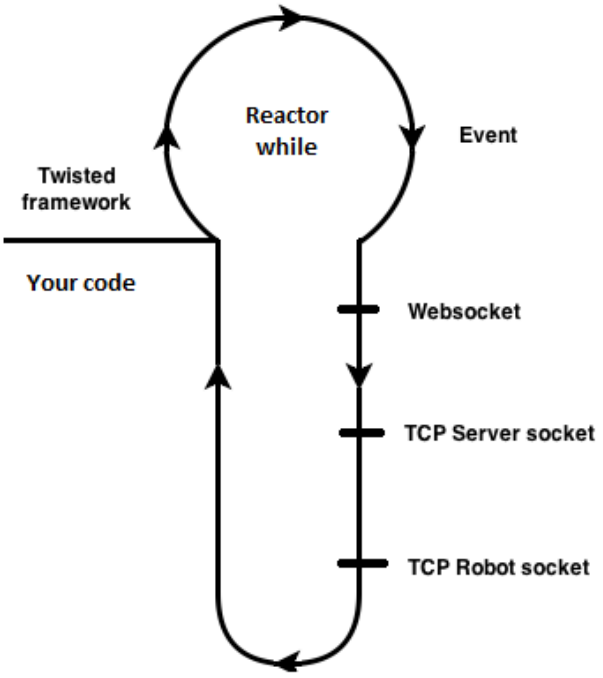


Figure 1. Reactor cycle, non-blocking waiting in usage of multiple sockets

There are only partial solutions or specialized systems, implementation and installation of which exceed the costs necessary for running my application. When dealing with the problem, I came across some interesting solutions that are technologically similar to my results or address a special part of the defined problem. All of them are beeing developed by a group of robotwebtools. [4]

Rosbridge 2.1 is a tool developed for communication between systems based on ROS and non-ros applications. MJPEG server 2.2 is a tool that provides transmission of video from ros-base system and publishes it in a form of a video stream. Robot management system 2.3 is a system that provides a graphical user's interface for managing robots by using tools mentioned above. Using of these tools we get a comprehensive system for managing robots, which has great use in a local network. The aim of my work wasn't to overcome the quality of control offered by the RMS, but forward the idea of robots management a step further.

One problems of the existing solutions is serving the robot as a server in the architecture, which brings along several disadvantages. Such robots waste their limited resources on the server overhead. If we wanted to perform communication over a public network, the robot would need to have a public IP address which might be a problem when we are thinking of the enormous number of robots in practice. Most existing solutions is designed for communication in a local network.

The biggest advantages of my solution compared to existing solutions is an exchange in roles the robot plays. In my architecture a robot represents a client that connects to the server 3, which implicates several positive facts. First of all my robot saves it expensive, limited resources and it can let the server solve difficult calculations, whose resources are much cheaper. Therefore the robot can be considered "only" as a data collector, which publishes data to the server, the brain. Another advantage of my solution is that the robot can communicate globally across the Internet without losing time spent on direction. Another significant benefit of my application is in the area of research where the

application allows you to process and distribute data to various non-ros platforms easily and thus to extend the spectrum of possible experimentation and using of robotic platforms.

2. Theory

2.1 Rosbridge

Rosbridge is one of many robotic tools which are beeing developed by a group called Robot Web Tools [4].

The basis of the robotic platform ROS [3] is roscore which is able to publish individual robot's sources to various local ports. In case we want to connect to these sources, we need to know for what the individual ports are. This could be difficult due to the dynamic port allocation. From a logical point of view of the Internet and client-server topology robot behaves as a server in this connection, and it often has no allocated static global IP address. Thus the possibility of our communication is limited to a local network. The aim of rosbridge [5] tools is to fix the problem of dynamic port allocation to robot's sources from roscore. Therefore Rosbridge unifies all the ports assigned by roscore, manages them and integrates them into a single port, which acts as a single point of access to the resources of the robot. This port can also be used in a local network with no-ros applications to communicate by using rosbridge protocol. [6]

Rosbridge is a quality tool that can be used for communication within the local network. When communicating it uses authentication and the communication can take place via https - therefore it contains the basic elements of security. When transferring resources from the robot, rosbridge provides many opportunities how to work with the data and it also allows you to work with a broad spectrum of data. One of these data is also video stream, that can be transmitted using rosbridge. However the transfer of the video stream is very unoptimized and slow in rosbridge.

Rosbridge is based on client-server architecture, the robot serves as a server. This distribution of architecture reveals a major problem. If the communication should take place globally, robot would need to have a public IP address. This is not a big problem for one robot, but if there were more robots, a problem with the lack of public IP addresses and increasing operating costs occurs. One another aspect is the unnecessary use of robot's resources which are considerably more expensive than the resources of the remote server. One another aspect is the view of safety, outlined in chapter 4.3.

2.2 Mjpeg server

Mjpeg Server [7] belongs to a group of tools that are beeing developed by group Robot Web Tools [4] like rosbridge, but because the transmission of video over rosbridge is still not optimal, the tool mjpeg server was implemented. Mjpeg server is optimized for transmitting data stream from a robot into a non-ros applications. The most commonly used data streams include video, but there is also the possibility of transmitting other sources of similar type. When transferring video stream MJPEG server allows a wide range of options to edit video "on the fly".

Significant deficiency like in rosbridge is that the application is designed only for the local communication, and thus it does not allow the global transmission of the data stream without any external intervention.

2.3 Robot Management System

RMS [8] is a tool for managing groups of robots based on the ros platform. RMS was developed by a group of Robot Web Tool. It is basically a tool that presents a graphical user interface and the background that allows you to manage robots. RMS uses the aforementioned tools and rosbridge 2.1 mjpegserver 2.2 to communicate with the robot and to video transmission. RMS enables authentication of user and access control. It contains graphical user interface for managing content and robots, remote control of robots and other gadgets.

2.4 Twisted framework

Twisted [9] is an event-driven framework written in Python and licensed as open source. It supports different types of protocols. Particular interest for us are TCP, HTTP Websocket. Twisted framework is based on an event driven programming, which means that the user of the framework writes a brief callback function. The main reason why I chose twisted framework is that it can listen actively to a greater number of sockets and sockets can be of various kind. The tool that makes this possible is called reactor and its principle is shown in a picture. 1.

3. Architecture of the system

The aim of my work was to create a system that will enable global and secure communication between the robot and the client, it will provide different levels of permissions and advanced management of users and robots. For this purpose, I have designed a communication layer 2 based on client-server architecture, which will allow non-ros applications to communicate with ROS [3] based platforms, using the specified protocol. [10]

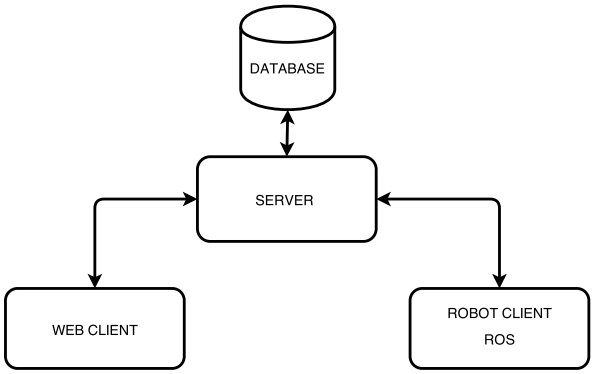


Figure 2. System diagram

Communication layer can be divided into four basic parts. Each individual layer solves a specific task and from a spatial perspective, each subsection of applications can run on independent and globally remote machines.

3.1 Server

The main part of the system is a server whose activities are divided into three subprocesses. 3 The main thread handles registration requests from clients, collects and distributes meta-data regarding clients and robots, deals with authentication and authorization of the client and the robot, handles rights and launching the communication channels between the client and the robot. This thread is always released only once by the machine, and persists throughout the operation of the server. The control cycle of the main thread is based on twisted framework [9] and it operates on reactor 1.

Communication thread represents a proxy server between the client and the robot completed with the management and control of communication. The proposed protocol [10] ensures the communication between the web client and communication thread where messages are transmitted via websockets. Communication between the communication thread and the robot is designed according to the same protocol and messages are transmitted using TCP/IP sockets. Messages for the robot are based on rosbridge [6] and are saved in JSON format.

The last part of the server is a thread that provides video stream. It requires HTTP/GET header [11] with precisely specified node and parameters necessary for proseccing the video from the client. In the next step the server sends this header to the process ensuring distribution of the video stream and its edit on the robot.

3.2 Web client

The logic of the communication layer is that it connects non-ros application with a robot based on ros platform. As a non-ros applications we can generally consider any application in which it is possible to create a graphical user interface, while this platform has the ability to communicate with the server using web sockets or TCP/IP protocol. I implemented a GUI using a web interface, which was used in the library roslib [12], using rosbridge protocol [6]. The application uses my proposed protocol [10] to communicate with different parts of the server described above. Messages for the robot are constructed using prescription of rosbridge protocol and formatted in JSON notation.

3.3 Robot client

This part of the system is the interface between roscore and communication layer. Its aim is to understand and realize the message received from the communication thread. It is divided into three basic threads 3. The main thread collects meta-data regarding the robot and registers the robot into the system. Based on the request from the server this thread initiates startup of two other threads. Client's communication thread based on the input parameters connects to the communication server thread. Video-stream thread like communication thread connects to the server video-stream thread on the basis of information received from the server. This step creates a bridge between the server and the robot, which allows global communication and video transmission between them.

4. Communication and working

In this part of article I will explain the process of initiation communication between the robot and the client, how the registration of the robot and the client proceed and how to transmit meta-data.

4.1 Working

The first part which needs to run is the server process. As a server I consider a sufficiently powerful infrastructure that will be able to calculate even complex algorithms with cubic complexity O (N^3) in a reasonable time. Another requirement is the stability, availability, speed and low response of the network connection. The process does not use complex data structures, and does not store any large amount of meta-data. Therefore, if the system will be used at less than 5000 robots, it is not necessary to solve its spatial complexity. Scalability is directly proportional to the available performance of the infrastructure on which the process is running. Theoretically the server is built

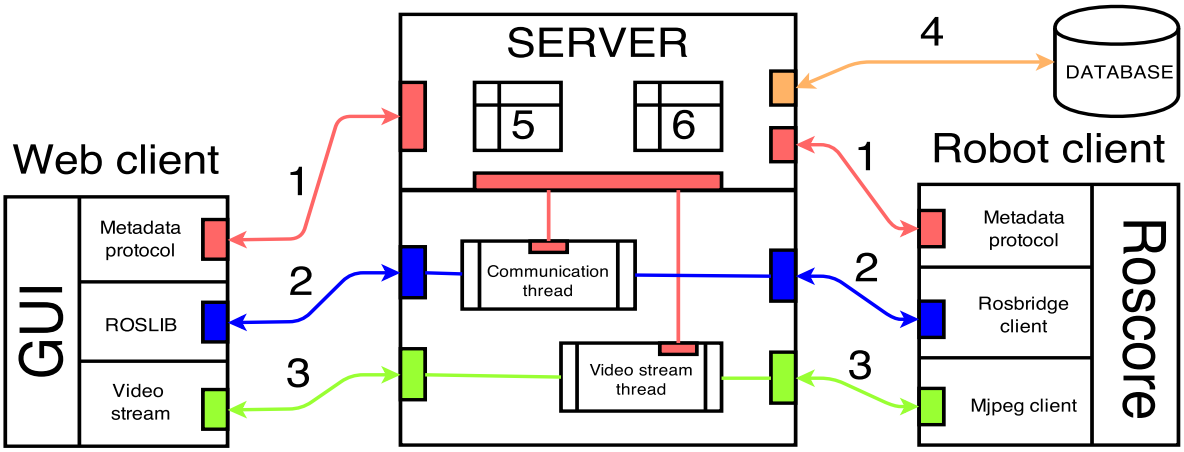


Figure 3

to operate infinite number of robots. Server must have allocated its own public IP address and the possibility to book its own port which is predefined for the 9090.

The second part is the robot, which must be based on a robotic platform ROS [3]. The robot must have installed the robot client 3.3 and each robot must generate its own ID and login, which should be used to log into the system. User of robot should ensure that the client process will have availavble enough processor's time and enough baud rate. The baud rate is directly proportional to the amount of data that are distributed to server 3.1. Particularly critical is the stability and response of data transfer between the robot and the server. At high response rates there is a problem with the actuality of available data at the web client 3.2, which can pose a significant problem especially if we want to control the robot using the video. In cases of connection instability it response can increase disproportionately or ot it can lead to a critical error and then to logoff the robot from the server.

The last part of the system is the web client 3.2, which generally can run on a variety of devices allowing the graphical user interface and supporting javascript. We assume a sufficiently fast and stable network connection.

4.2 Communication protocol

On condition that all the requirements that are mentioned in chapter 4.1 are fulfilled, communication protocol platform can continue.

Registration of robot begins by starting robot client with input parameters that are ID, login, IP address of the server and the port to which the server listens to. Starting robot client will be performed automatically while the robotic platform is beeing starting. A message based on the protocol is created [10]. Message contains meta-data regarding the robot.

After getting the registration requirement the server checks if the robot's metadata match with metadata stored in the database, especially a login ID. After a successful verification, the server creates an object of robot type and inserts it into the list of currently registered robots and sets its status parameters.

Login web client runs via the login gateway based on login and password. When logging in, the client displays the current list of robots for which it currently has access and robot's status parameters. It also creates a registration message based on protocol, which is sent to the server 3.1.

When server receives a registration message from the client it creates an object of type webclient, set its status parameters and inserts it into the list of currently connected clients.

If the robot is not working currently, and the user has permission to work with this robot, the client can initiate communication with the robot. The web client creates a message based on protocol [10] and sends it to the server 3.1. The server evaluates the parameters of the message and after successful testing, it creates a communication thread.

After successfull establishing a communication thread, the communication thread itself logs on the main thread of the server 3.1 and assignes to one of the state parameters of the robot. After registration of thread it needs to send a message with meta-data regarding the communication thread to both of clients.

After receiving the message containing meta-data regarding the communication thread, web client redirects itself to work graphical user interface and registers in the communication thread 3.1.

After receiving message containing meta-data regarding the communication thread, robot client creates a communications thread 3.1 and registers it in the communication thread on the server 3.1. After these

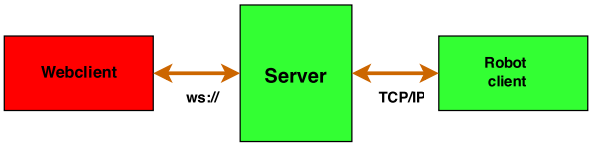


Figure 4. Visualization of levels of security

operations the system is ready for the communication between the web client 3.2 and the robot 3.3.

Initiation of video stream is based on the same principle as the launch of communication threads but with the difference that different metainformation and messages are used and video is displayed in the HTML5 tag <video>.

For more detailed specification of protocol and messages see technical documentation for communication platform. [10]

4.3 Safety communication

When implementing the application I've tried to make the communication between elements of the system as safe as possible in accordance with safety standards.

One commonly used solution dealing with the problem and using the rosbridge server [5] is that a robot is considered as server 3.1 and opens one of its ports to a client 3.2 interested in communication with the robot. In such architecture everyone, who knows on which port the server is running, can connect and communicate with the robot. Therefore, I consider such a solution as relatively dangerous for commercial use. In order to solve this problem and also the problem of the global communication it was necessary to insert layer between the client 3.2 and the robot 3.3. The layer manages the access rights of users to the robot.

For this layer we can considered the main thread of server 3. This thread has the task to receive authentication requests from clients and robots. Subsequently it registers these requests in the internal data structure. After authentication the client can work with the system whereas the field of possibilities is given by its authorization. Levels of authorization assigns the system administrator and they are checked every time a user enters into the system. [13]

5. Demo application

One of my aims was to create a demo application in the form of a web portal for remote management of ground robots at the Faculty of Information Technology in order to test the created communication platform. [14]

I started up the server on a virtual server with a public IP address, a fast enough hardware and connection. I also established a database with testing data on the virtual server. As reference robots I chose robotic laboratory at the Faculty of Information Technology, which belongs to the group Robo@FIT [15]. All robots are based on a robotic platform ROS [3].

For the client and graphical user interface, I chose Web technologies, where I used WebSocket communication for creating messages based on rosbridge protocol [6], roslib library [12] and GUI framework bootstrap. For testing, I created a functional management system, which is used for practical and research purposes Robo@FIT group.

My work is theoretically dimensioned to an unlimited number of robots and clients, where capacity is dependent on hardware performance of server 3.1 and speed of connection.

As a lack of the application I consider its security in connection with the use of websockets and javascript that I described in chapter 4. As another drawback I see a great range of using JavaScript, which is safety problem on one hand. On the other hand the need of the client 3.2 hardware performance begins disproportionately to grow with more complex work, which is highly undesirable problem. As a solution, I imagine minimizing the usage of JavaScript and keeping a big part of the work with the data on the server using PHP and frameworks.

6. Conclusion

This work thematizes the problem of remote robot control over a public network, and it explains why this issue should be solved. It points out one of possible solutions and explains why the the problem should be solved in this way. My work outlines the basics of safety communication in the system and then it demonstrates created demoapplication 5 and evaluates advantages and drawbacks of the application.

The result is a functional communication layer that enables the management of robots 3.3 and clients 3.2. It resolves security of communication 4.3 between them, supervises the authorization and authentication. It also allows you to create extensions to the communications layer and thus to create various applications dealing with administration and management of robots over a public network. The result of this work is also a demonstration of demoapplication, which is used to manage ground robots at the Faculty of Information Technology, Technical University in Brno and robotic group Robo@FIT [15].

The contribution of this work is in designing a tool that allows global communication between the client 3.2 and the robot 3.3 and ensures safety of this communication 4.3. It represents the interface between

non-ros applications and robots based on ros platform. It provides advanced management of robots and clients, solves client's access rights to robots and helps to increase the efficiency of the robot.

6.1 Future plans

In a further development of this application I plan to implement the M:N bond in the communication between the client and the robot. This means that one client can control multiple robots, or a robot can be handled by more clients. Another plan is expanding the graphical user interface to demo application and adding new functionality. One possible extension is also controlling of autonomous robot guard.

The possibility of expanding and experimenting with communication platform are wide because it combines two different platforms and thus it opens the door to a huge number of new applications. Another significant benefit is to provide cheap computing power of server to a robot and thus save limited resources of the robot. One of the possible expanding direction which I imagine could be an application that combines robot household and thus represents a centralized system where robots can share other informations.

Cooperation of two robots is performed that one robot uses data from sensors of second robot, which saves expensive hardware resources.

As one of the typical use cases we can mention management solutions of robot cleaners in the business chain. There is a chain store, which uses autonomous robots for cleaning their premises at night. Normally, the administrator would have to set each robot to its own specific role and then let it run. If it comes to a failure of some robot, the robot stops working, which reduces effectiveness of using it. Handyman realizes the mistake only when he sees the robot by his own eyes. When using my solution the administrator logs into the system, starts cleaning mode, and then he can see what the robot is doing. In case of any failure, the system informs the controller immediately and then he can solve the error either online or manually directly on the robot depending on the type of the error.

7. Acknowledgement

This way I would like to thank Ing. Vítězslav Beran, Ph.D. for professional guidance at work. I would also like to thank the Faculty of Information Technology and Robo@FIT group for providing me with space and robots.

References

[1] Amazon. Amazon warehouse robots, Dec 2014. https://www.youtube.com/watch?v=quWFjS3Ci7A.

[2] Aaron Smith Janna Anderson. Ai, robotics, and the future of jobs, Aug 2014. http://www.pewinternet.org/2014/08/06/future-of-jobs/.

[3] Open source. Roslib, Dec 2006. http://www.ros.org/.

[4] Russell Toris. Robotwebtools, Dec 2013. http://robotwebtools.org/index.html.

[5] Jonathan Mace. Rosbridge suite, Apr 2014. http://wiki.ros.org/rosbridge_suite.

[6] Jonathan Mace. Rosbridge 2.0 protocol, Apr 2013. https://github.com/RobotWebTools/rosbridge_suite/blob/groovy-devel/ROSBRIDGE_PROTOCOL.md.

[7] Benjamin Pitzer. Mjpeg server, Aug 2014. http://wiki.ros.org/mjpeg_server.

[8] Russell Toris. The robot management system, Apr 2013. http://wiki.ros.org/rms.

[9] Twisted matrix. Twisted matrix, Sept 2011. https://twistedmatrix.com/trac/.

[10] Andrej Tichý. Communication platform for management and control of robots, Apr 2015. https://github.com/xtichy09/Remote-robots-ROBOFIT.

[11] Burak Guzel. Http headers for dummies, Dec 2009. http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039.

[12] Ken Conley. Roslib, Dec 2013. http://wiki.ros.org/roslib.

[13] Heroku. Websocket security, Dec 2012. https://devcenter.heroku.com/articles/websocket-security.

[14] Andrej Tichý. Robo@fit portal login, Apr 2015. http://37.205.11.196/login.html.

[15] Faculty of Information Technology VUTBR. Robo@fit, Sept 2010. http://www.fit.vutbr.cz/research/groups/robo/.