

Fontys Hogescholen

Programming and Visualizing Industrial Robots with a Web-Based GUI

Zirui Song (2652528)

INTERNSHIP REPORT

FONTYS UNIVERSITY OF APPLIED SCIENCES

HBO-ICT: English Stream

|  |  |
| --- | --- |
| Data student: | |
| Family name , initials: | Z Song |
| Student number: | 2652528 |
| assignment period: | 30-08-2017 to 05-02-2018 |
| Data company: | |
| Name: | Fontys Hogeschool Engineering |
| Department: | Mechanic & Robotics Laboratory |
| Address: | R1 0.204-0.205 |
| Company tutor: | |
| Family name, initials: | PC Negrete Rubio |
| Position: |  |
| University tutor: | |
| Family name , initials: | E Thaqi |
| Final report: | |
| Title: | Programming and visualizing Industrial robots with a web based GUI |
| Date: | 17-01-2018 |

Approved and signed by the company tutor:

Date: 17-01-2018

Signature:

# Foreword

I was invited for an internship opportunity from the Mecanics & Robotics Labortary of Fontys Hogeschool Engineering. The internship project is to develop a web-based GUI for an Industrial Robot that the robot can be visualized and programmed in the browser.

This project is based on DOT framework and strategies used are library, workshop and showroom.

Contents

[Summary 1](#_Toc503900486)

[Glossary 1](#_Toc503900487)

[1. Introduction 2](#_Toc503900488)

[2. About the company 2](#_Toc503900489)

[3. Project overview 3](#_Toc503900490)

[3.1. Facts 3](#_Toc503900491)

[3.2. Target of Project 5](#_Toc503900492)

[3.3. Analysis to the project 6](#_Toc503900493)

[4. Project Progress 7](#_Toc503900494)

[4.1. Prepositive Prototypes 7](#_Toc503900495)

[4.2. Development of Final Product 9](#_Toc503900496)

[5. Conclusion and Outlook 12](#_Toc503900497)

[5.1. Project Summary 12](#_Toc503900498)

[5.2. Further Functionalities 12](#_Toc503900499)

[Evaluation 13](#_Toc503900500)

[References 13](#_Toc503900501)

[Attachment A: Project Plan (Excerpts) 14](#_Toc503900502)

[Assumptions, Constraints and Risks 14](#_Toc503900503)

[Project Management Approach 14](#_Toc503900504)

[Attachment B: Links of Source Codes 15](#_Toc503900505)

[Attachment C: Links of Videos 15](#_Toc503900506)

# Summary

This project researched on web application of ROS. The goal of this project was to develop a web-based GUI for Universal Robots UR5 Industrial Robot that the robot can be visualized and programmed in the browser.

To finish this project, some problems should be solved. How the web application communicate with ROS? How to visualize the robot in the browser? How to realize MoveIt! functionalities for robot’s motion planning with Javascript? This required to find the way to achieve the goal from the internet and practice those methods. The application was to be finished by solving the problems before.

# Glossary

|  |  |
| --- | --- |
| **Term** | **Explanation** |
| GUI | Graphical user interface |
| HCI | Human-Conputer Interaction (usually means programming tool of robot) |
| Main viewer | The section in the application that show the scene with the robot |
| Node | An ROS process |
| Parameter | A set of data with key stored in ROS |
| ROS | Robot Operating System |
| Service | An ROS request/reply message pair |
| Topic | The bus where ROS nodes communicate with each other |
| URDF | Unified Robot Description Format, XML format describing the robot |
| Waypoint | A specific posture of the robot, containing all the value of joints |

# Introduction

This project is to build up a control-box web application for an industrial robot with JavaScript, which is to be run in any devices and compatible with any industrial robots.

Universal Robots UR5 is equipped in the labortary so it is to be used for this project. So the substance of this project is making a web-based GUI for UR5 with functionaity of:

* Real-time visualization;
* Moving the robot to predetermined place;
* Simple programming for the robot.

Chapter 2 provides information about the company.

Chapter 3 gives all the details about the project and its related things.

Chapter 4 describes how the project is done.

Chapter 5 concludes this project.

# About the company

The Fontys mechatronics laboratory main goal is to recreate and solve industrial challenges while contributing to Fontys students education. The challenges that students face in the lab, are developed following stricted use case scenarios that company partners face in a daily basis, because of that, students get to work in the most innovative topics, giving to our company partners a proof of concept before implementing new technologies.

Engineering students in semester 7 will attend the automation & robotics class and do their practical in the robotics laboratory. The practice content is shifting from the production of dedecated production lines with a fixed layout and processing order for a product or product family, towards intelligent, more or less autonomous, processing stations that can be combined in an ad-hoc way depending on the desired product.



Figure 1: Students working on their projects with UR5 robot

# Project overview

## Facts

### Universal Robots UR5

Universal Robots is a Danish robotics company founded in 2005. It provides UR3, UR5 and UR10 series of industrial robots to meet worldwide industrial automation requirements in different levels. UR5 is a typical 6-axis industrial robot. The maximum payload of UR5 is 5 kg and its work radius is 850 mm. (Universal Robots A/S, 2009)

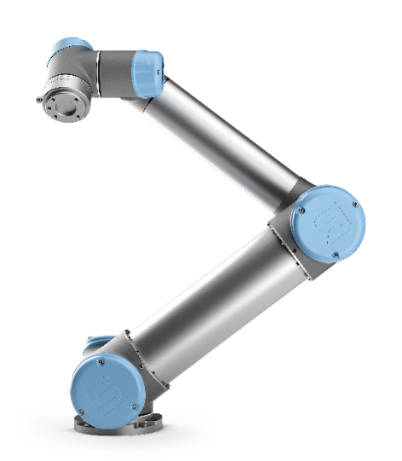


Figure 2: Universal Robots UR5

Fontys Robotics Laboratory equips two UR5 robots, one of which is used for this project.

### Robot Operating System

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. (About ROS, n.d.)

There are many ways to communicate with ROS:

* Call services,
* Subscribe to or publish on topics, and
* Get or set parameters.

### URDF and TF

The Unified Robot Description Format (URDF) is an XML specification to describe a robot. The specification covers kinematic and dynamic description of the robot, visual representation of the robot and collision model of the robot. The description of a robot consists of a set of link elements, and a set of joint elements connecting the links together. (XML Robot Description Format (URDF))

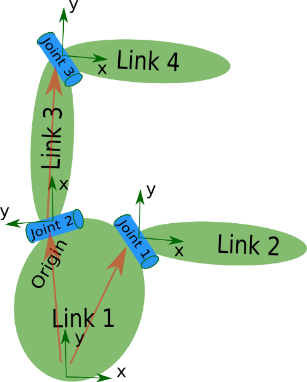


Figure 3: The Unified Robot Description Format (URDF)

So a typical robot description looks something like this:

1. <robot name="pr2">
2. <link> ... </link>
3. <link> ... </link>
4. <link> ... </link>
5. <joint> .... </joint>
6. <joint> .... </joint>
7. <joint> .... </joint>
8. </robot>

The tf library was designed to provide a standard way to keep track of coordinate frames and transform data within an entire system such that individual component users can be confident that the data is in the coordinate frame that they want without requiring knowledge of all the coordinate frames in the system. During early development of the Robot Operating System (ROS), keeping track of coordinate frames was identified as a common pain point for developers. (Foote, 2013)

### MoveIt!

MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains. MoveIt! is the most widely used open-source software for manipulation and has been used on over 65 robots.

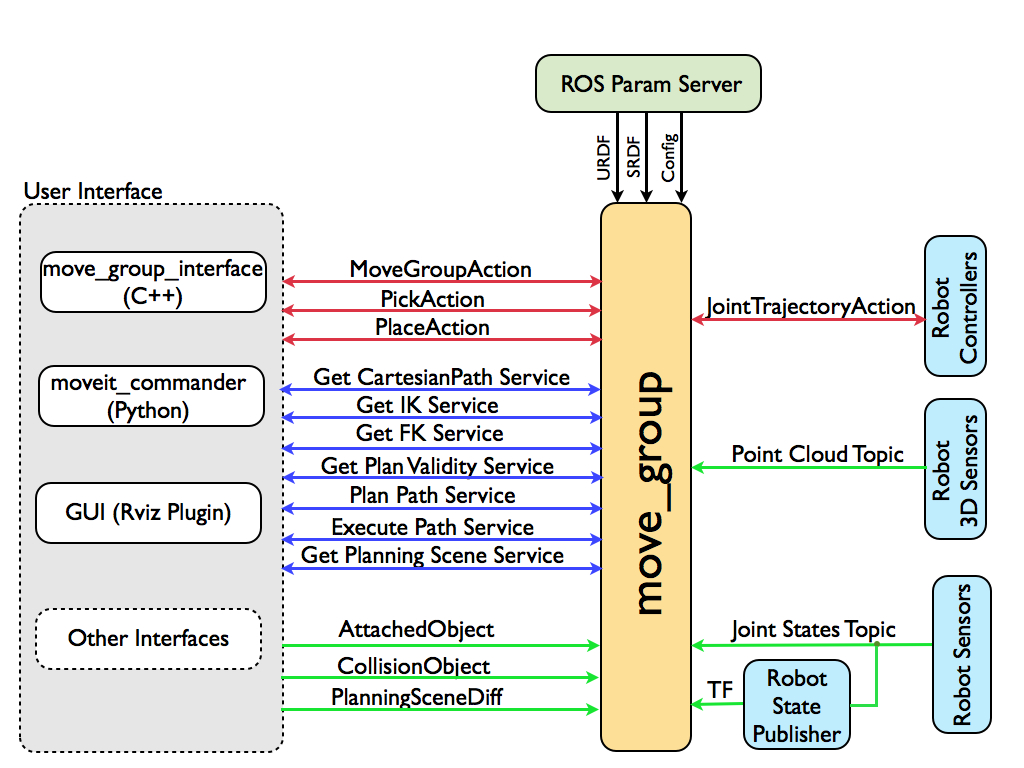


Figure 4: The system architecture of MoveIt!

Figure 4 shows the high-level system architecture for the primary node provided by MoveIt! called move\_group. This node serves as an integrator: pulling all the individual components together to provide a set of ROS actions and services for users to use.

move\_group is a ROS node. It uses the ROS param server to get the URDF, SRDF and MoveIt! configuration. move\_group talks to the robot through ROS topics and actions. It communicates with the robot to get current state information (positions of the joints, etc.), to get point clouds and other sensor data from the robot sensors and to talk to the controllers on the robot. (Sucan & Chitta, n.d.)

### RobotWebTools (including roslibjs and ros3djs)

RobotWebTools is a collection of open-source modules and tools for building web-based robot apps. Their tools allow web applications to interface with a variety of robots running middleware like the popular Robot Operating System (ROS) using the latest in HTML5 and JavaScript. roslibjs and ros3djs are used in this project with the rosbridge on server-side.

rosbridge communicates ROS data messages contained in the JavaScript Object Notation (JSON) for straightforward marshalling and demarshalling. Through its use of WebSockets (a protocol built on top of HTTP), rosbridge can be readily used with modern web browsers without the need for installation. This fact combined with its portability and pervasive use makes the web browser an ideal platform for human-robot interaction. (Toris, et al., 2015)

roslibjs uses WebSockets to connect with rosbridge and provides publishing, subscribing, service calls, actionlib, TF, URDF parsing, and other essential ROS functionality. Functionalities of connection, publishing, subscribing, service calls and parameter updating are used in this project.

ros3djs is the standard JavaScript 3D visualization manager for ROS. It is build ontop of roslibjs and utilizes the power of three.js. Many standard ROS features like interactive markers, URDFs, and maps are included as part of this library.

### Vue.js

Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries. (Introduction)

## Target of Project

There are many industrial robots manufactures in our planet. Each brand of industrial robots has its pattern of controlling and programming. This makes trouble for some enterprises that need purchase industrial robots from different manufactures because they must recruit more employees to use and maintain them. So, it is good to make a unified graphical user interface that compatible with all brands of industrial robots. It is also possible to add another robot into the system. And the best thing is, you will be able to control your industrial robots at anywhere (via secured local network of your enterprise) with any device (PCs, laptops, tablets, smartphones, etc.).

The project is aimed to research, design and develop a web based graphical user interface (GUI) that lets the users communicate, program and visualize Industrial robots from different brands that are connected to it. The web based GUI is developed using Javascript, HTML5, CSS and roslibjs for the front-end and it must connect to rosbridge as the server running in ROS (Robot operating system).

Industrial robots have its own GUI to let users program and interact with the robots. Each brand of Industrial robots has its own GUI and they can be very different.

The assignment requires the design and development of a GUI that lets any possible user easily program and interact with any industrial robot connected through rosbridge in ROS to the GUI.

### Goals and Objectives

* Design following Human-computer interaction (HCI) patterns for industrial robots and that  
  communicates to the ROS server using roslibjs.
* Let the user visualize in 3D the movement that the industrial robot will perform.
* Make sure the app can run efficiently in different modern browsers and devices (laptops,  
  tablets and phones).
* Document the product well and make it easily maintainable and extendable.
* Follow the Agile method and meet in a weekly basis with trainee’s company tutor to inform  
  any new challenge and the state of the project.

## Analysis to the project

### Project Architecture

The system where the application is inside can be splitted into 3 parts: the web application itself, ROS and the robot. ROS plays a middleman role in this system. ROS connects to the robot, read the joints and publish to the browser. The browser also sends the commands to ROS and ROS talks to robot about that.

In this system, rosbridge provides an interface to the browser in ROS. The webapp uses roslibjs to connect to ROS and ros3djs for visualization. ROS connects to the robot with UR Driver and carries out motion planning by MoveIt!. The webapp part with its connection to ROS is to be done in this project.

Figure 5: The system map of the application

### Research Methodology: DOT Framework

In the project, the DOT framework is applied during research phase. The procedure of each research follows Rigor cycle. Just like below:

1. Follow online tutorials and documents (*Library*, *Lab*)
2. Try to build prototype, then the final product (*Workshop*)
3. Show the result on request (*Showroom*)

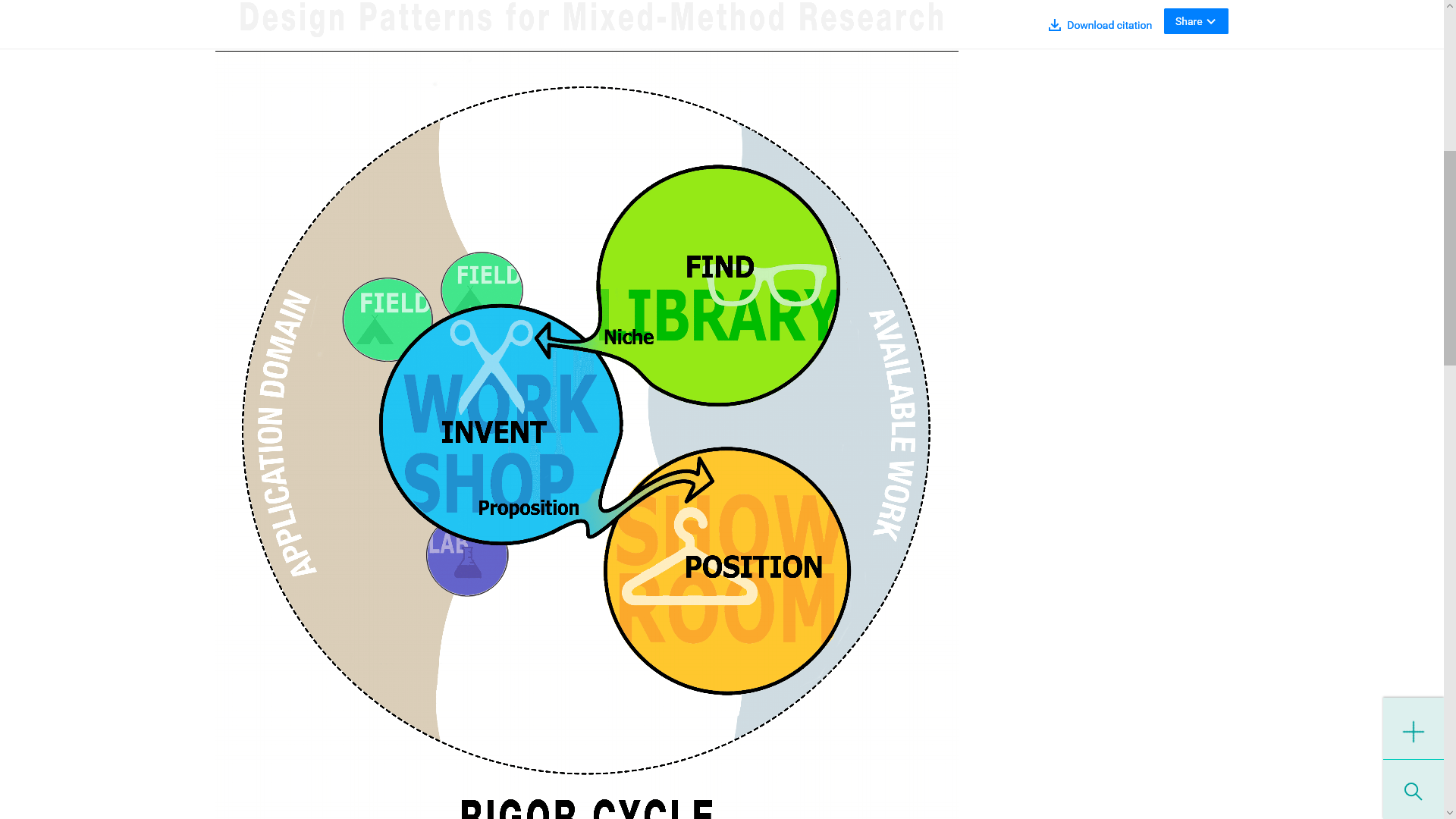


Figure 6: Rigor cycle of DOT framework

# Project Progress

In this chapter you describe **what** you've done and **how**, and especially **why**. You describe what systematic way you have followed from the assignment to the end product(s). Usually this chapter is called Results, but you are free to choose your own title.

You used a lot of research strategies (e.g. Field, Library) to get to the end result. Explain the techniques you used to execute your strategies. For example describe surveys you conducted, experiments you carried out, comparisons of possible techniques, tools and solutions, used literature found on the Internet or in books.

The progress of this project was splitted into two parts: prototyping and final product developing.

## Prepositive Prototypes

Before the final project was done, three prototypes were made for preparation as following:

* turtlesim on the web
* URDF online viewer
* MoveIt! web-based planner

These prototypes helped to master the necessary technologies that was to be used in the development of final product.

### Prototype 0: turtlesim on the Web

turtlesim is an ROS application that is used in ROS official tutorials. This application helps beginners know the basic knowledge of ROS. This prototype is aim to get a brief overview of ROS and how to communicate between browser and ROS with connection of rosbridge and the driver roslibjs. It simulates the interface of turtlesim and provides some functions by communicate to specific topics and services.

turtlesim has a 500 by 500 background color-changeable field and a turtle inside. The turtle has X, Y, Theta properties for its position in the field, and when it moves, it leaves its trace in the field. The brush width and color can also be changed. To move the turtle, linear and angular velocity values are to be sent for the movement of turtle, which will drive turtle to move for 1 second.

This prototype reproduces the field and the turtle in it, which is built in such structure:

* div#canvas: main field with background color set by turtlesim
  + canvas#drawing: records pen track of turtle
  + div#turtle: sets the position and rotation, contains the image of turtle
    - img#turtleImage: the turtle that can be seen in the webpage

The background of the field is defined by getting the parameters background\_b for blue channel, background\_g for green channel and background\_r for red channel. By subscribing turtle1/pose topic the display refreshes to show the state of turtlesim in real-time.

The control panel provides the functionalities of moving 1 step, moving constantly and stop, changing pen color including turning pen on or off and change background color. These functionailties are achieved in different ways:

* **Move**: Publish linear and angular velocity to the topic /turtleX/cmd\_vel;
* **Change pen color**: Call the service turtleX/set\_pen with pen color, width and on or off;
* **Change background color**: Set parameters background\_b, background\_g and background\_r, then call service clear to make changes effective.

roslibjs is introduced to let the prototype communicate with ROS. It is used to connect the prototype to ROS, subscribe the topics of turtle’s position, publish the topic of move, call the service of clearing field or setting the pen properties, and get or set background color. The prototype also uses setInterval() to achieve letting the turtle move constantly by publishing the same topic per every second and stops it by using clearInterval() when user clicks “Stop” button.

This prototype was done in the first month of the project period. By working on it, we had a general understanding of ROS and how to use roslibjs to communicate between ROS and the browser, which is helpful for working on further things.

### Prototype 1: URDF Online Viewer

This prototype is an application of usage of ros3djs. ros3djs provides a 3D visual of robot in collaboration with THREE.js. In this prototype it was used for showing the 3D-model in order to present the status of connected robot in real-time in the browser.

First, the connection of ROS is established by roslibjs. Meanwhile, the prototype creates a ROS3d viewer with ros3djs. The id of element, width and height is set when creating it, which will let ros3djs create a main viewer with specific size at the designated element of the webpage. A grid is added into the viewer to make it looks better. Then, a TF client is created for the ROS client with its default preferences. Lastly, the URDF client is created. The ROS client, TF client, path of models and the main viewer is defined in this procedure. Collada Loader in three.js is used for loading the model of UR5.

Then main viewer is a seperate web page and is contained with iframe in the main page. When the connection is opened successfully, the prototype will send a http GET request to the main viewer. The main viewer decodes the request with URL of ROS and robot model before loading itself.

Besides, the prototype also provides the value of joints by subscribing `/joint\_states` topic which contains the name and the value of all joints of the robot. In the message of the topic, the names and values are separately set in two arrays. These arrays are copied into the data in Vue pattern. With Vue.js these values can be shown in the page easily:

1. <div class="form-group">
2. <label>Joints' State</label>
3. <p v-for="(name, index) in nameJoints">{{name}}: {{valueJoints[index]}}</p>
4. </div>

In the code above, `v-for` is used for a for-loop of the `p` element in the webpage with each value and index used in array `nameJoints`. The index of `nameJoints` is used for get the corresponding value of joint in array `valueJoints`.

### Prototype 2: MoveIt! Web-Based Planner

The purpose of this prototype is to achieve controlling the robot within the browser.

MoveIt! is a very popular motion planning tool in ROS, which is also used in this project. This prototype lets the browser contact MoveIt! to make movement to the robot.

A similar webapp that already exists is found for this project. This webapp is called rwt\_moveit, which is part of visualization\_rwt created by Tokyo Opensource Robotics Kyokai Assosiation, and based on Denso VS060 industrial robot. This application has a webpage and some nodes written in Python. The nodes provides the topic subscription for MoveIt! motion planning. The webpage has full functionality of MoveIt! with:

* Manipulate start state or goal state
* Move the robot with interactive marker
* Plan and execute the motion
* Change the move group and single joint's value

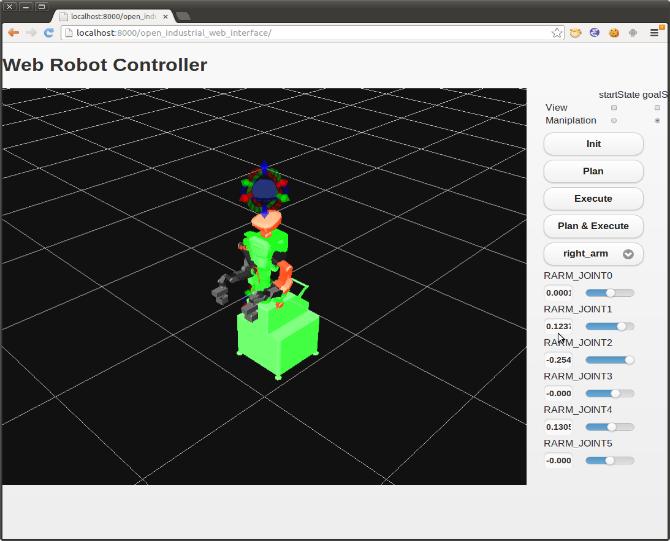


Figure 7: The interface of rwt\_moveit webapp

However, this webapp could not be used at first. We contacted the authors for the solution of the problem, finally found the way to solve it with the help of them and colleagues in the labortary and made the application compatible with UR5.

This prototype reduced the time and workload of researching on the problem, and made more time for developing the final product.

## Development of Final Product

The final product was expanded from the prototype 2 with a lot of changes applied on it. The major changes were rebuilding user interfaces, login page, simple programming based on waypoints and response of robot's abnormal stop.

### Rebuild User Interface

The application is designed for running on any device such as PC, laptop, tablet, phone, etc. So the user interface should be fit within all devices. However, the UI for prototype 2 couldn't meet this requirement. So we rebuilt the UI with Bootstrap.

Bootstrap provides a responsive gridding system for webpages to make the website fit with any device.

1. <div class="row">
2. <div class="col-xs-12 col-sm-6 col-md-9" id="urdf-container">
3. ..
4. </div>
5. <div class="col-xs-12 col-sm-6 col-md-3">
6. ..
7. </div>
8. </div>

The code above means:

* When the width of window is less than 575 px, the main viewer fills the window in width, setting the controls below the main viewer. On the other hand, the height of main viewer is set to 50% of window height (this is done with JavaScript).
* When When the width of window is more than 576 px but less than 767 px, the main viewer at left takes up 50% of window width, remaining half width for controls at right side. This ensures the right side is wide enough to fit the controls inside when using a device with small screen.
* When the width of window is more than 768 px, the main viewer at left takes up 75% of window width to maximize the main viewer.

Bootstrap also contains different useful controls and a layout system for webpages. These functions made the web page better-looking but less workload.

To make a better viewer, we also changed the main viewer to light theme by setting the color of background and the grid:

1. var viewer = new ROS3D.Viewer({
2. divID : 'urdf',
3. width : width,
4. height : height,
5. antialias : true,
6. background: '#cccccc'
7. });
8. viewer.addObject(new ROS3D.Grid({color: '#666666'}));

### Login and App Loading

A login page is introduced into this application. It prevents unauthorized personnel from improperly using this application and any personal injury or property damage caused by that.

TThe code of this part is under Vue.js pattern. That means the inputs of username and password are linked with the related variables and the latter updates when the inputs change. When user clicks “Login and connect” button, It checks if the user exists and the password is correct. Currently, as a prototype, the user data is hard-coded into the script file. If the username and password match, Vue will hide the login interface to show the main interface and establish the connection to ROS at the same time.

During the initialization of the application, the following parts were changed from the prototype.

The start state didn't being used for motion planning in the prototype, which means the model and the interactive marker of start state is useless. But the code of hiding them didn't work. In order to avoid user's confusion, we commented out the code of generating the URDF client and interactive marker of the start state to make them disappear from main viewer. It is also necessary to prevent the errors from removing them.

Manipulator is the only useful move group of UR5. In the prototype, manipulator is not default move group; user should change it to “manipulator” manually, which is inconvenient. Applying $('select#group').val('manipulator').change(); in the code to let the move group switch to manipulator automatically and trigger the event of the select box to make this change effective. With this action, user will no longer need to change the move group and the select box can be hidden (for UR5).

### Adding, going to, Saving and Loading of Waypoints

When the goal state of MoveIt! planning executed successfully, the “Add Waypoints” button turns enabled and user can click to save the waypoint into the list (array). The array preserves all waypoints user saved. The waypoints will show as buttons with the name of each waypoint. By clicking it, the robot will move to the waypoint immediately. User can also let the robot moves via all waypoints in order by clicking “Run All Waypoints”.

The information of joints is extracted from the message of /joint\_states with each waypoint’s name, its joints’ names and positions. This is done by set the joint values of sliders as of waypoint, make a MoveIt! planning and execute it in one step. which is done by planning and executing all the waypoints one by one. In this procedure, the completion of robot's each step is detected by subscribing from topic /follow\_joint\_trajectory/result or /arm\_controller/follow\_joint\_trajectory/result for virtual environment, following with actions of planning and executing next waypoint in the list or step out after last waypoint is executed.

The list of waypoints can be saved as a JSON file by clicking “Save Waypoints” button. The array of waypoints is converted to JSON and taken over by FileSaver.js for saving it to external file. The loading of file is splitted into two steps. The first step is trigger click event of a hidden file input to call the open file dialog when user clicks “Load Waypoints” button. The second step happens when a file is selected, which means the file input has changed. FileLoader in JavaScript is used for load the JSON file. After that, the JSON will be converted to array and overwrite the current one.

### Receiving E-Stops and Protective Stops

An emergency button is equipped on UR5 robot. When something emergency happens, the operator can press this button to force the robot stop and lock it. The protective stop is triggered when the robot unexceptedly deviates from its establised track in robot moving to help protective the robot from being damaged. These measures protect the safety of the robot, the operators and the production line.

The UR driver can also detect emergency stops and protective stops. When an emengency stop or a protective stop occurs, ROS will receive one of these following messages:

1. [ERROR][T] Emergency stop pressed!
2. [ERROR][T] Robot is protective stopped!

According to the result of monitoring the topic /rosout\_agg, these messages were found from a node called driver. With this, the topic /rosout\_agg is subscribed; However, only the messages which are typed as “Error” and come from driver will be responsed. The messages will be popped as a message box to tell the user the robot is emergency or protective stopped, stop the running waypoints and prevent user from adding new waypoint here.

# Conclusion and Outlook

## Project Summary

This project researched on the posibilities of programming and visualizing an industrial robot using a web application. The work of RobotWebTools and other ROS contributors make it possible, and the effort in this project realizes it. This allows user monitors their robot at anywhere with any device. Currently, most insudtrial robots have their own HCI pattern for programming the robots. The application made in this project also make the unifying the programming language of all types of robots, which can reduce the cost of the enterprises.

## Further Functionalities

Because of time limit of the internship, some problems and functions were not solved in the project. The following aspects can be the goals of further project(s) based on this product.

### Move robot by using arrow buttons (with kinematics.js)

Buttons for move and rotate are placed into the GUI, but functions are not implemented. To make the robot move by arrows possible, the X, Y, Z values of the position and rotation of the robot end effector should be calculated. kinematics.js is a utility that can calculate the posture of end effector with the positions and values of the 6 joints. It is included in the product and can be used for position calculation.

### Work with Node.js for login and database of waypoints

The login function is just a prototype and there is no user management function provided. It is suggested to use Node.js and mongodb to save user data better and safer. The login page should also based on the user data in the Node.js database. In addition, the app can do more such as saving the program made by specific user with the database.

### Enhancement functionality of programming

The simple programming tool only provide the function of adding the waypoint to the list and save/load the list to/from external file. It can be extended to be more functional by adding waypoints removal and reordering function, simple logical and loop statements and variable operations.

### Stability of robot motion

When using the real robot, protective stops often occur. This is because the robot moves too fast and sometimes it misses the target point. To protect the robot, a protective stop is triggered by itself; then the robot is to be unlocked from its HCI. It is necessary to improve the driver module to avoid the robot from moving fast, stopping heavily and triggering protective stops.

### Compatible with other types of industrial robots

This web-based GUI app is compatible to Universal Robots UR5. However, it may also suitable for any type of 6-axis industrial robots or even any robots that using ROS. This product can be splitted into two parts: common part that is same to all robots, and special part that differs to adapt each robot. When adapting a new type of robot, the common part should not change, while the special part can be copied from sample and modified to fit the data and settings of the new robot.

### Make good use of Vue.js

Only part of the code is under Vue.js environment. Putting everything inside the Vue object in app.js file can make good use of Vue.js functionalities and make further coding easier.

# Evaluation

I learned a lot of things when working on this project in this 5-month internship period. Apart from the knowledge of JavaScript, this project also requires the understanding of Robot Operating System and plenty of its concepts, which was strange to me firstly. This project helped me to get more experience on web programming and these new knowledges about robotics that was not learned from school. I also tried finding the answers by myself to get the answers and solve the problems. The tutor was really nice and also gave me a lot of help and hints. After all, I enjoyed the experience of doing internship here.

# References

*About ROS*. (n.d.). Retrieved from ROS.org: http://www.ros.org/about-ros/

Foote, T. (2013). tf: The transform library. *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, (pp. 1-6).

*Introduction.* (n.d.). Retrieved from Vue.js: https://vuejs.org/v2/guide/

Sucan, I., & Chitta, S. (n.d.). Retrieved from "MoveIt!": http://moveit.ros.org

Toris, R., Kammerl, J., Lu, D., Lee, J., Jenkins, O., Osentoski, S., . . . Chernova, S. (2015). Robot Web Tools: Efficient Messaging for Cloud Robotics. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Universal Robots A/S. (2009). *UR5 User Manual.*

*XML Robot Description Format (URDF).* (n.d.). Retrieved from ROS.org: http://wiki.ros.org/urdf/XML/model

# Attachment A: Project Plan (Excerpts)

## Assumptions, Constraints and Risks

### Project Assumptions

The following assumptions were made in preparing the project plan:

* A laptop with Ubuntu 14.04 and ROS Indigo installed is provided as work computer.
* The robot that will be part of the assignment, including their respective ROS (robot operating
* system) 3D models and drivers is provided and always accessible.
* The trainee will abide by the guidelines identified within this plan.
* The project plan may change as new information and issues are revealed.

### Project Constraints

The following represent known project constraints:

* No related projects are known for reference.
* The period of internship is limited.

### Project Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **Likelihood** | **Impact** | **Measures** |
| 1 | Accident that leads to damage of equipment | Unlikely | High | OPERATE THE ROBOT SAFELY |
| 2 | Work computer stolen, damaged or data inside damaged | Unlikely | High | Store the laptop firmly in the drawer and lock when leaving, backup codes on GitHub |
| 3 | Robot malfunction | Somewhat likely | High | Report to tutor immediately |
| 4 | Robot is occupied by other students when needed | Certainly | Low | Adjust daily plan, work on things does not need to operate robot |

## Project Management Approach

### Project Approaches

|  |  |  |
| --- | --- | --- |
| **#** | **Title** | **Action** |
| 1 | Preparation | Learn basic knowledges of Vue.js and ROS |
| 2 | Research | Visualize and control the robot in the browser |
| 3 | Development | Develop the end-product |
| 4 | Finalizing | Finalize the project and write final reports |

### Milestones

|  |  |  |
| --- | --- | --- |
| **#** | **Milestone** | **Deliverables** |
| 1 | Preparation | Prototype that shows basic communicate between browser and ROS |
| 2 | Research on visualization | Prototype that visualize a remote robot |
| 3 | Research on controlling | Prototype that can control a robot remotely |
| 4 | Development | End-product with full functionality |
| 5 | Finalizing | End-product tested and all documents of the project |

# Attachment B: Links of Source Codes

All the source codes are placed in GitHub.

Code of Prototype 0:   
<https://github.com/CRH380B-6216L/webROSGUI_prototype0>

Code of Prototype 1:   
<https://github.com/CRH380B-6216L/webROSGUI_prototype1>

Code of Prototype 2:  
<https://github.com/CRH380B-6216L/visualization_rwt/tree/hydro-devel/rwt_moveit>

Code of final product:  
<https://github.com/CRH380B-6216L/ur5_webgui/>

# Attachment C: Links of Videos

Videos are made during working period for showing the progress to the company.

Demonstration of Prototype 0:   
<https://youtu.be/MVroEbGX5Gc>

Demonstration of Prototype 1:   
<https://youtu.be/7M6ThsQKKKE>

Demonstration of Prototype 2:  
<https://youtu.be/Pqc1IhRfw_w>

Demonstration of final product:  
<https://youtu.be/iIB0xDJjZNA>