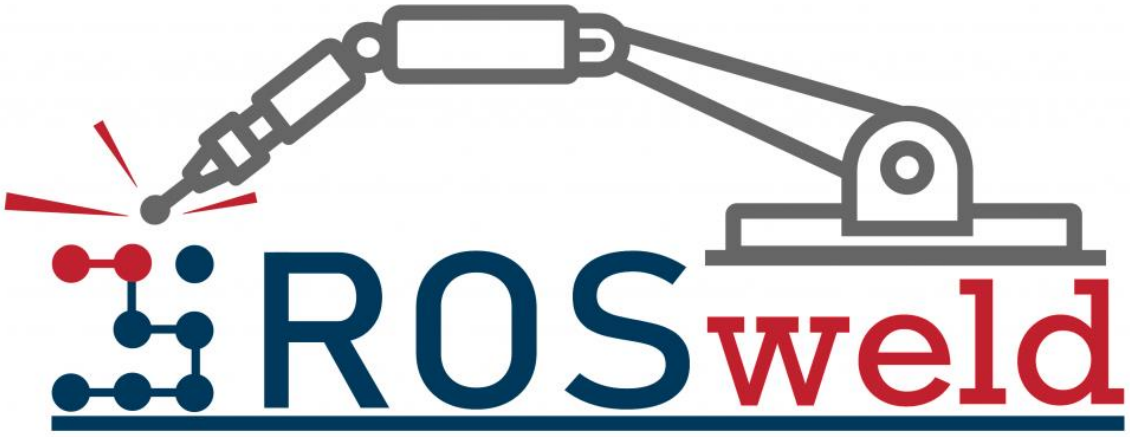


USER DOCUMENTATION

Content

1 Introduction

ROSWELD is a planning, monitoring and control software suite on an industrial quality level, for heavy industrial robot applications. The ROSWELD framework is built upon the synergy of available ROS components, being upgraded to the industrial quality level; with the new, high-tech modules for heavy robot applications developed within ROSWELD.

Additionally, the graphical multi-platform user interface FlexGui 4.0 with the Technology Readiness Level (TRL) 9 (official Horizon 2020 TRL scale), is already a standard ROS tool to create easy-to-use interfaces to ROS-based robot and software applications.

A part of ROSWELD will be programming and planning software on TRL6 and will be adapted and exported to general ROS components. In addition, ROSWELD will use the following ROS components:

- algorithms for path planning
- *roscore* for basic messaging
- *rosbridge* for platform-independent communication
- *rosbag* for logging and debugging
- *MoveIt*! for robot-independent movement planning
- *FlexGui 4.0*

Additionally, ROSWELD will provide interfaces to robots and standard CAM systems, to ensure the portability between different platforms, and thus, preparing the ROSWELD technology to be applied in various industrial environments.

To ensure an optimal use of the ROSWELD framework, , knowledge of the following components and areas is required:

- ROS and *MoveIt!*
- *FlexGui 4.0*
- Basic knowledge related to NACHI, Hyundai, or OTC industrial robots including:
    - Robot system configuration and setup
    - Robot operation
    - Programming and multi-threading
    - File operation and remote access
- Basic knowledge of the welding processes and related equipment

2 Installation

The ROSweld framework is a ROS compatible application requiring additional software components beside the standard installation. In order to utilize the full capability of the ROSweld framework, or modify its source code, the following versions of the components need to be installed on the PC running the ROS server:

- Ubuntu 16.04
- Catkin (python-catkin-tools)
- git
- qt5 default
- ROS kinetic
    - trac-ik
    - mjpeg_server
    - tf2-web-republisher
    - rosbridge-suite
    - libuvc_driver
- MoveIt!
- Apache2
- Python 2.7
    - PIP
    - numpy + ROS numpy
    - jsonpickle
    - pydispatcher
    - pyros_setup (rospy)
    - *websocket-client* [1]
    - *configparser*
    - *mock*
    - *pytest*
    - *Sphinx*
    - *pytest-cov*
- *vscode:*
    - *Python*
    - *autoDocstring*
    - *pylint*
    - *svn*
- ROSweld:
    - FlexGui 4.0
    - ROSweld_tools
    - ROSweld_drivers
    - ROSweld_robot
    - ROSweld_tools_examples [2]
    - ppm_labor

[1] Items marked with *italic* are necessary only for modifing and/or testing the code
[2] Underscored ítems are only necessary for the demo setup

- ppm_labor_control

2.1 Installation manuals:

- Ubuntu 16.04: https://www.tecmint.com/ubuntu-16-04-installation-guide/
- ROS: http://wiki.ros.org/ROS/Installation
- ROSbridge: http://wiki.ros.org/rosbridge_suite
- FlexGui 4.0: https://www.ppm.no/FlexGui4-Home/Index/downloads
- libuvc_camera: http://wiki.ros.org/libuvc_camera
- MoveIt!: http://moveit.ros.org/install/
- Apache2: http://httpd.apache.org/docs/2.4/install.html
- Python 2.7: https://tecadmin.net/install-python-2-7-on-ubuntu-and-linuxmint/
- PIP: https://pip.pypa.io/en/stable/installing/
- PIP/Python 2.7 packages: https://packaging.python.org/tutorials/installing-packages/
- VSCode: https://code.visualstudio.com/docs/setup/linux

Please note: in our system, only the following command was working to set up the libuvc_camera:

```
sudo chmod o+w /dev/bus/usb/001/024 <-- where 024 is port# from the error code
```

It is recommended to use Ubuntu 16.04 and ROS Kinetic, but most of the components are continuously supported on newer versions as well.

The required ROS packages can be installed with the following commands:

```
sudo apt-get install ros-kinetic-trac-ik-kinematics-plugin

sudo apt-get install ros-kinetic-tf2-web-republisher

sudo apt-get install ros-kinetic-rosbridge-suite

#Assuming, that you have a catkin workspace, called catkin_ws in your home folder

cd ~/catkin_ws/src

git clone https://github.com/RobotWebTools/mjpeg_server.git

cd ..

catkin_make
```

Please note, that *VSCode* is only needed to modify the code or finetune the settings.

It is strongly advised, but not required to install the add-ons for VSCode. These extensions available at the Extension Marketplace: https://code.visualstudio.com/docs/editor/extension-gallery

2.2 Apache2

In order to use FlexGui 4.0 on the client device (PC or mobile devices with HTML5 browser), an *Apache2* web server is needed on the ROS server machine. It is recommended to add FlexGui 4.0 to your catkin workspace, so the web server will be able to provide the URDF and 3D model files for the 3D Fidget of

FlexGui 4.0, based on *ros3d.js* framework. *Apache2* is intended to host FlexGui 4.0, and you can connect to the application remotely, using the host PC IP in the browser.

This way, there is no need for an additional Virtual Directory or a simple pythonic web server (*ROSweld_tools/src/webserver.py*).

To change the default directory, please modify the paths to your workspace in both *apache2.conf* and *000-default.conf* configuration files.

```
sudo gedit /etc/apache2/apache2.conf
sudo gedit /etc/apache2/sites-available/000-default.conf
```

2.3 Installing ROSWELD packages

The installation of the necessary ROSweld packages needs to create a catkin workspace on the ROS server machine and download the packages from *git*:

```
#Create a new folder in your home or use the already existing workspace:
mkdir catkin_ws
cd catkin_ws
#clone our git repositories to your machine
git clone http://git.ppm.no/ROSweld_tools.git
git clone http://git.ppm.no/ROSweld_tools_examples.git
git clone http://git.ppm.no/ppm_labor.git
git clone http://git.ppm.no/ppm_labor_control.git
git clone http://git.ppm.no/ROSweld_robot.git
git clone http://git.ppm.no/ROSweld_drivers.git
git clone https://github.com/ros-industrial/FlexGui_industrial.git
#Initialize the catkin workspace
catkin init
#make the workspace
catkin_make
#source the setup.bash in your .bashrc if it's not there
echo 'source ~/catkin_ws/devel/setup.bash' >> ~/.bashrc
```

At this point, if the *catkin_make* compiles were successful and without error, that means everything is installed correctly and ROSWELD is ready to be used on your ROS server machine.

3 Configuration

The configuration of the virtual environment according to the real one requires consideration of the ROSWELD framework settings. The *unified robot description format* (URDF) is defining the positions of

the objects in the virtual environment, where it is advised to place the robot into the origin of the virtual environment's coordinate system.

The welding process CAM path defined in the coordinate system of the workpiece model is also referred to as the *Model space*. The coordinate system is defined during the export of the model from the CAD modelling program or is directly generated in our 2D path generator (included in the FlexGui 4.0 project).

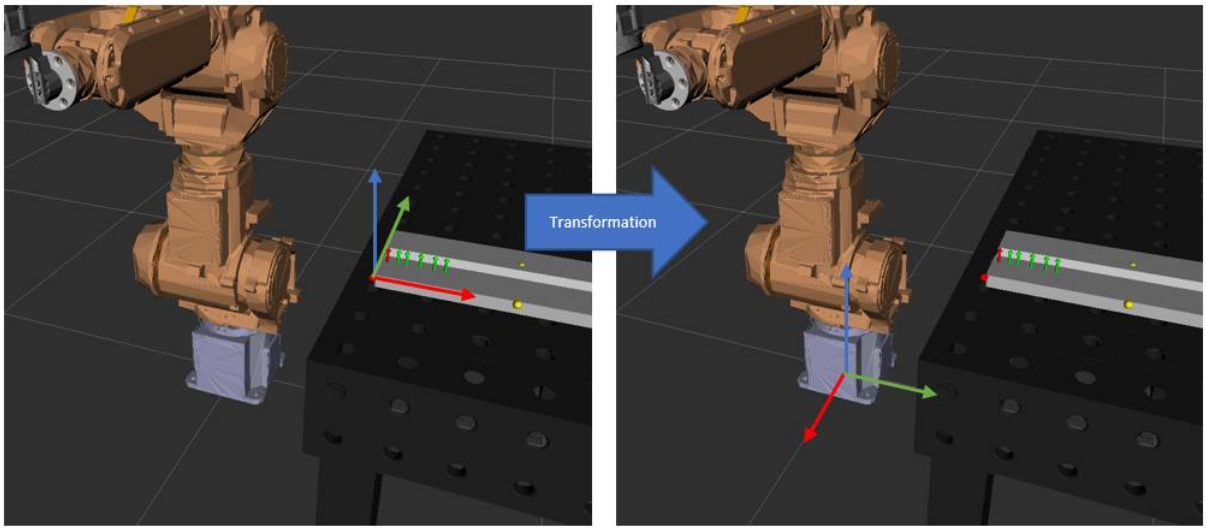


Image 1 Coordinate transformation

With the given calibration procedure, the transformation matrix from the model space to the robot base can be defined; therefore the workpiece features can be described with robot base coordinates.

The CAM file's format is the following: Each path point is defined in separate lines as $x\ y\ z\ r\ p\ y$ sixes, where the values are separated by tabulators and ends with a line breaks. The file will be saved with **.wmcam* extension.

The points in the welding process path contain position and orientation information. After the calibration procedure performed, the path becomes executable providing the position for the robot tool. In the ROSWELD project, the initial orientation of the tool is defined along the following coordinate axis: the direction of the welding defines the X-axis, that points towards the next path even if there is a curvature in the path definition. The Z-axis points outwards the groove, aligned with the bisection of the groove angle or the surface normal vector. The $Y - axis$ is defined respectively to the other two axes.

The tool is defined as the electrode axis is the $Z - axis$, where the positive direction is the direction of increasing distance between the electrode tip and the workpiece. The $X -$ and $Y - axis$ are defined arbitrarily, following the common practice. For example, in Gas Tungsten Arc Welding, the $X - axis$ is pointing from the tungsten electrode tip towards the filling wire entry point.

4 Usage

With *ROSweld_tools_examples* a test environment can be run, where the workpiece is a V groove plate and the robot type is NACHI MR20 seven-axis industrial robot arm.

Please note: by default, this demo works with *MoveIt!* as the simulation interface and a NACHI MR20 robot.

To run the example, use the following script:

```
roslaunch ROSweld_tools_examples start.launch
```

By default, all camera connections are disabled. If the cameras are needed, please edit the ppm_cameras.launch file to match with the required camera properties and enable them in the launch file:

```
roslaunch ROSweld_tools_examples start.launch cameras:=true
```

What should happen if you run the following script? – First of all, ROS will start with *rosbridge* server over *websocket*. Then, *ROSweld_tools_examples* will run *moveIt!* and *Rviz* to visualize our virtual environment (Image 2 Example setup).

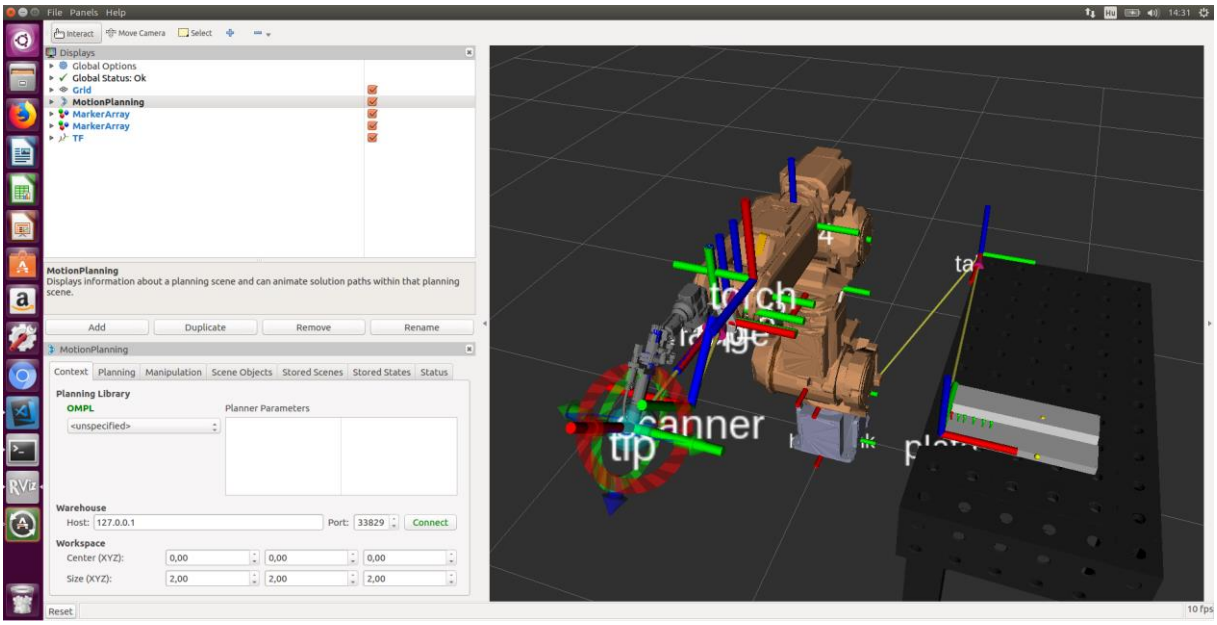


Image 2 Example setup

The example project is made with the wizard of *MoveIt!.* With some minor modifications, it can be easily configured for your own robot with your own workspace in the demo.

Please check lab.launch file in the ROSweld_tools_examples package as a reference and modify the demo.launch file (output of the moveIt! wizard) and add the roswled specific lines according to the ROSweld.launch file.

The following components will run:

ROSweld_tools/src/**app.py**	The main file of our application
ROSweld_drivers/src/drivers/**move_it_robot.py**	Our MoveIt! robot driver
ROSweld_drivers/src/drivers/**nachi_robot.py**	NACHI MR20 7-axis robot driver (port: 8000) *
ROSweld_drivers/src/drivers/**uEpsilon.py**	MEL M2D laser scanner driver (port: 3000) *

ROSweld_drivers/src/drivers/**otc_wps.py** OTC WPS driver (port: 8001) *

You can disable real robot / equipment communication, if you remove these lines from the *start.launch* file.

At this point, an initial project is loaded, with an initial calibration. All the services and topics are published and can be used from FlexGui 4.0 or directly from ROS.

4.1 Services

ROSWELD is providing services to control the welding process. The available services can be listed by the *rosservice list* command. The list of the available services is collected and documented at the description of each node.

SetService is a service type in ROS, provided by the ROSWELD framework. It has two parameters: a command and a *JSON* argument. It is possible to send commands to the ROSWELD framework's different interfaces to perform the desired task. Some of the commands without-, but more often requires a *JSON* object to define the command parameters. Please see the details in the following chapters. The reason for using the *SetService* as the primary input of the framework is to limit the advertised services. The advanced ROS Welding Planner will provide separated services.

4.1.1 Calibration

The calibration service is intended to define and record the calibration points to create the transformation matrix from the model coordinate system to the robot coordinate system. The coordinates of the calibration points are defined in the *Model space* (mostly on the workpiece or fixture). Then, the corresponding points positions on the real objects are recorded using the *tool center point* (TCP). Read more about the used calibration procedure in Chapter 3 Configuration.

The *set_calibration* service is a *SetService* type service in ROS. Please see the list below for the available commands and their parameters for this service:

add the command appends the given (x, y, z) point as a new calibration point. The (x, y, z) point is defined as a JSON object, with x, y and z parameters in model space coordinates, the point is on the workpiece.

set_selected_point the command sets the selected calibration point. The index of the calibration point is defined in the *JSON*, with *index* parameter.

goto the command sends the robot to the previously recorded robot position of the selected calibration point. JSON parameter has an optional *speed* argument with a default value *1* in mm/s defining the TCP speed of the robot

safe_goto the command is like the goto command, but with an additional *distance* JSON argument with default value *0.02* in z-direction in meters, defined in model space.

save the command saves the calibration to the storage. The JSON argument is empty.

cancel the command terminates the current calibration and loads the previous one (default if none).

record the command records the current robot position and orientation reported by the robot to ROS, and pairs it with the selected calibration point, making a model space – robot coordinate pair definition. The JSON argument is empty.

remove the command removes the selected calibration point from the list. The JSON argument is empty.

4.1.2 Project control

The *project_control* service is a *SetService* type service in ROS. The available commands and their parameters are as following:

play the command sends a play command to the current task in the plan. It is possible to add custom parameters, such as: *type* to define the type of the movement (goin, verify, etc). The play command is performed on the current task.

pause the command stops the robot motion execution when going on a given path, the pause command is forwarded to the current task in the plan.

save the command saves the project to the storage, the project will automatically load on next startup.

cancel drops all the changes in the project made since the last saving and loads it back if possible (default project is loaded otherwise).

input the command forwards the input to the selected task with the given JSON. The parameter and the value must be defined in the JSON as *param* and *value* parameters. The possible params are the following:

- **path** loads the given path to the task if possible. The path is a cam file, defined in Chapter 3 Configuration.
- **step** with the *step* parameter, the *current step* selected on the path. If the *current step* is a reachable point on the path the robot will move there, otherwise returns with an error.
- **modification** this parameter is connected to the current step. ROSWELD will obtain the modification list for the currently selected step or create a new one and include the new values: z, y, $angle$, $delta\_r$, $amperage$, $voltage$, $filler\_speed$. The properties and the definitions of the parameters are given in 4.4.4 Bead editor.
- **bead** this parameter is used for setting up the current weld bead's default offset and angle, to the given $y, z, angle$ values.

Example code for the *SetService* type calls:

```
rosservice call /ROSweld/project_control "command: 'input' json: '{\'param\
': \'step\', \'value\':5}'"
```

4.1.3 Move

Command the robot (or the simulation) to the desired position or to move along a path, **/move_along** service has to be called with the Moves (ROSweld_tools/Move[]). Each move has its own Pose (position and orientation), welding parameters (ROSweld_tools/WeldingPrameters), accuracy, acceleration and tool definition.

> **Please note,** that move_along service will override the stored poses on the robot / simulation and will use the given list to calculate the current step number on the path.

Alternatively, if the robot / simulation needs to be moved to a specific single pose without modifying the path, the **/move_pose** service can be used with the same parameters.

4.2 Topics

ROSWELD publishes the following topics:

plan	the currently loaded project as a JSON object
calibration	the currently active calibration points (*ROSweld_tools/Calibration* message, with *ROSweld_tools/CalibrationPoint[]*)
robot_state	the current position of the robot with orientation (*geometry_msgs/Pose*) and many other useful data, for example, joint angles, RPY, and step number. The **robot_state** topic is published by *ROSweld Core*, the data originated from the real or virtual robot, based on the *isSimualtion* flag.

4.3 Message and service types

ROSWELD is providing the following message and service types to ROS:

ROSweld_tools/Calibration	CalibrationPoint array
ROSweld_tools/CalibrationPoint	model point (geometry_msgs/Point) and measured pose (geometry_msgs/Pose) pair
ROSweld_tools/Project	project_json, default_settings and current_path
ROSweld_drivers/EulerAngles	rx, ry, rz
ROSweld_drivers/RobotState	error, isSimulation, mode, isMoving, robotProgramState, storedPoses, speed, step, euler_angles, joints and pose
ROSweld_drivers/Move	pose, accuracy, acceleration, tool and welding parameters
ROSweld_drivers/WeldingState	containing the following parameters: voltage, amperage, filler_speed, default_arc_length, mode, speed, is_arc_on, error, job_number
ROSweld_drivers/WeldingJobs	configurations, current_index, auto_update

4.4 FlexGui 4.0

The ROSWELD tools examples package provides a graphical user interface for FlexGui 4.0. Its source is located at *ROSweld_tools_examples/FlexGui/ROSweld.fgproj* and runs in a browser. The project has

seven screens, which are described in detail in the following sections. The FlexGui 4.0 manual can be accessed through the website https://www.ppm.no/FlexGui4-Home/Index/downloads.

4.4.1 Welcome

A simple welcome screen loads as the default screen. By pressing the START button you will be redirected to the Monitoring screen.

4.4.2 Calibration

On this screen, the calibration of the robot to the virtual environment can be performed. Three calibration points must be defined by moving the robot to the chosen calibration points' location on the real object and by pressing Record. After the recording of all three points, the transformation matrix will be calculated and used for the model space – robot space coordinate system conversion.

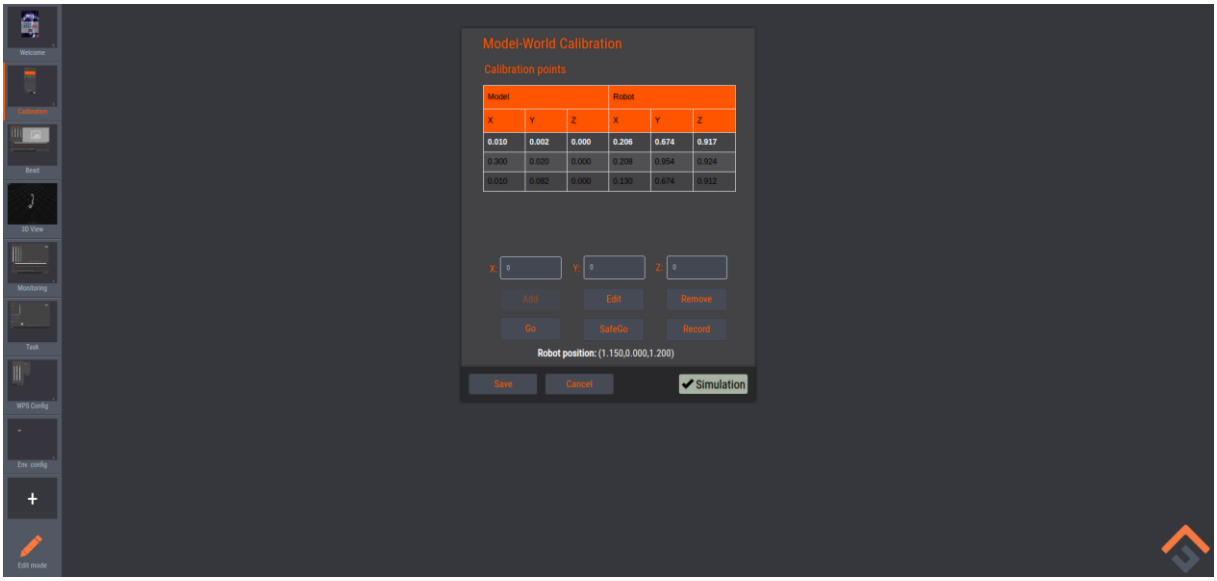


Image 3 The FlexGui 4.0 - Calibration screen

It is also possible to define more than three points through console commands, although

> **please note,** only the first three will be used for the calculation, therefore adding the over three points is disabled on the GUI.

The existing model points can be deleted from the point list by pressing the *Delete* button press. Their coordinate points can be redefined by selecting the calibration point pressing the *Edit* button. The new values (inserted into the *x, y, z* input fields) will be updated, and the transformation matrix will be recalculated on every change.

The calibration points displayed in RViz, as a MarkerArray, are represented as yellow spheres and changes to red upon selection. The same visualization is valid for the 3D View in FlexGui 4.0.

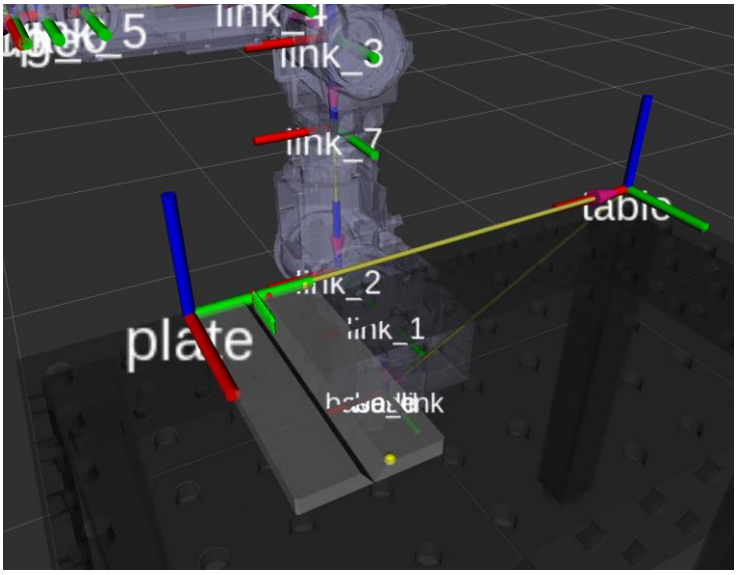


Image 4 Calibration markers on the workpiece and coordinate system definitions for the components of the virtual environment in RViz

Pressing the Save button will write the calibration to the storage (overwriting the existing one, if there is any), so that after restarting ROSWELD the saved calibration file will load and be used automatically. Cancel will load the last – saved – calibration file, or if there is none, load the default calibration.

4.4.3 Task editor

The Task editor screen is shown in Image 5. First, a **CAM path** should be uploaded by pressing the file input field (8) and by selecting a file with **.wmcam* extension.

Please note: uploading a path will override the current path for the selected task.

ROSWELD will use this CAM file for the *current task* as a *reference path* definition. For each weld bead can be defined **y and z offset and angle** (2), what translated into a general modification valid for all path points. Those general modifications will be not listed in the *Bead editor*. Further weld beads (4) can be added along the reference path, and the modifications applied for each one, separately. Predefined welding jobs can be added for each weld bead individually and used during welding as default. The modifications in the jobs can be performed on the **Bead editor** screen.

The default torch speed (1) definition can be made using the slider next to the camera image (5), where the unit of the speed is in mm/s.

Please note: ROSweld MS1 has a simplified planner, without a connection between the weld beads.

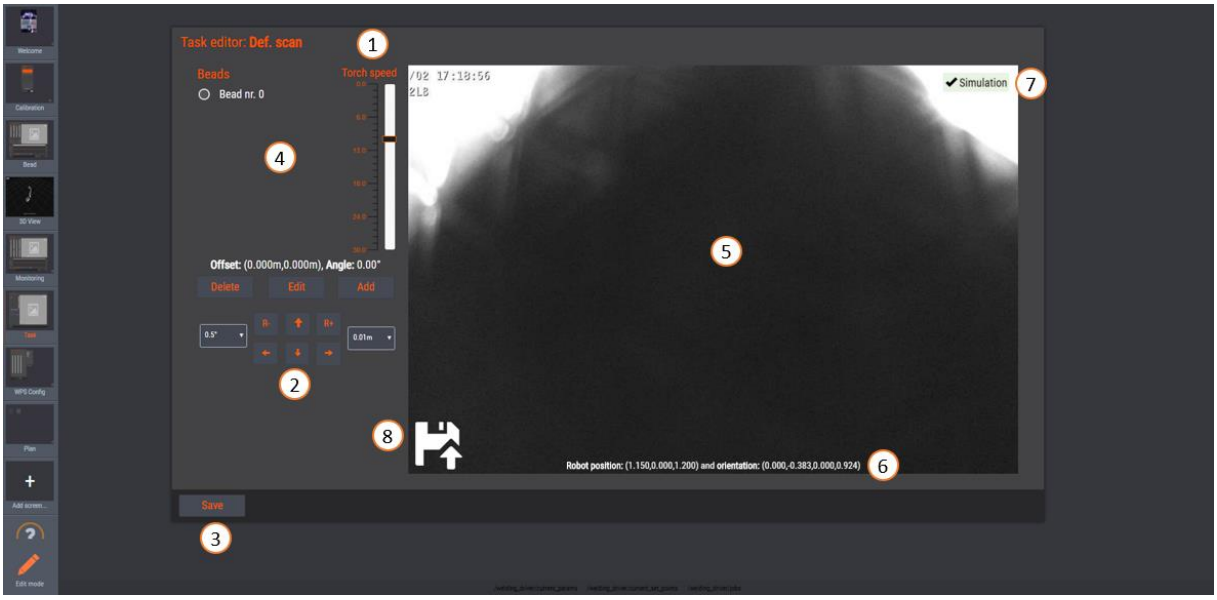


Image 5 The Current task editor

4.4.4 Bead editor

The bead editor screen, shown in Image 6, is designed to provide an interface to edit the current weld bead parameters in the selected task. The screen built up from the following elements: Progress bar (1), Welding parameter editor and control bars(6), Control keys for path modifications (2), Camera image (5), Simulation/robot switch (7), Robot pose indicator (8), Modification visual on/off (9), Save (3), Movement controls (4).

Image 6 The Bead editor screen in FlexGui 4.0

Welding parameters

To edit the welding parameters on the selected step, use the Amperage/Voltage/Filler speed/Torch speed control bars (Image 6, Item 6). The defined values from the first path point are valid for the whole length of the process, but changes can be applied at each step point. These values are limited

to min/max range, which can be changed in the *init script* of the FlexGui 4.0 project. Thehe scaling of the scrollbars will follow the changes.

Path modification parameters

The process path refers to the welding of one weld bead as a step-by-step parameterized path. It can be edited by using the control keys in the middle of the screen (2). The changeable parameters are the Y and Z offset, Angle and Delta R.

Defining the parameters: Y and Z offset (referred to the original path), working angle (R), z-rotation for the whole path (noted as D, global, point-by-point rotation along path ref. Z axis). After each change, the virtual robot on the right side will move accordingly. At the names of the paramters (9), the current values are shown at the selected step.

The **progress bar** represents the process path, where the markers are placed at the path step-points where the modifications can be added. A small badge will show, what modifications were made on that step. Hovering the badge with the mouse cursor will show a summary of the modification made in that step. Clicking on the *progress markers* will send the robot to the selected position.

At the bottom of the progress bar, a chart can be seen with the colour-coded welding and position parameters. By pressing the name of the parameter will toggle the hide or show of the modifications made in this parameter.

Pressing the **Save** button will write the project to the storage overwriting the latest one and will be loaded upon the next startup. The **Upload path** will transfer the current path with the presented modification but without saving it to the storage.

By pressing the **Go in** button, the robot is commanded to move to the first position on the path. On the **Check** button press, the robot will go back to the first position on the path and go along it until the end.

```
Please note: MoveIt! and the virtual robot will stop if there is a collision!
```

4.4.4.1 Move the robot

To move the robot, there are multiple options:

Go in	will send the robot to the 1st point of the path
Abort	will stop the robot on the current position
Check	will move the robot from the first step to the end of the path
Set step	scrolling the scrollbar will send the robot to the selected point. The last point of the generated path is the selected step and the path will go through on all the points between the last and the currently selected step, including the modifications

4.4.4.2 Modifications

A modification on the path is a change in the pose (y, z, angle, d rotation) or the welding parameters.

Modifying the pose will immediately be applied and incrementally added to the path from the modified index. The robot will move on the edited path automatically. The following actions are done, when modifying the pose:

- Create a new path modification
- Set y, z, angle, d rotation
- Recalculate path
- Upload to the robot
- Move to the new pose

Therefore, modifying the pose will start the robot/simulation to move.

The defined welding parameters will be applied during welding. When the robot reports its status with the step parameter, the ROSWELD framework updates the new welding parameters on the WPS, with the *set_params* service of the WPS driver.

4.4.5 WPS Config

On the WPS config screen (Image 7), the welding job can be edited to configure the basic reference parameters of the WPS in ROSweld. If the connected welding driver supports automatic update, the job will be uploaded to it; otherwise manually update of it on the WPS is needed, the edit is only a local representation of the job itself.

With the refresh button on the right, an updated list list of the WPS jobs can be obtained.

If some changes were made in the configurations, it must be saved by pressing the Save button or Cancel to drop the changes.

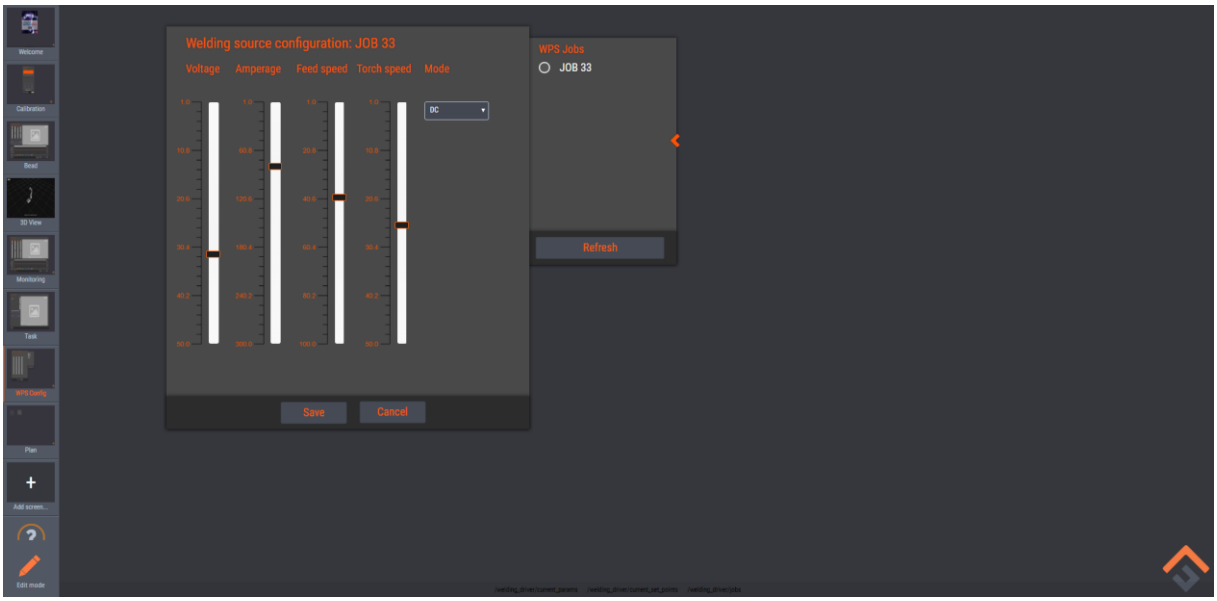


Image 7 The WPS job editor

4.4.6 2D Path generation

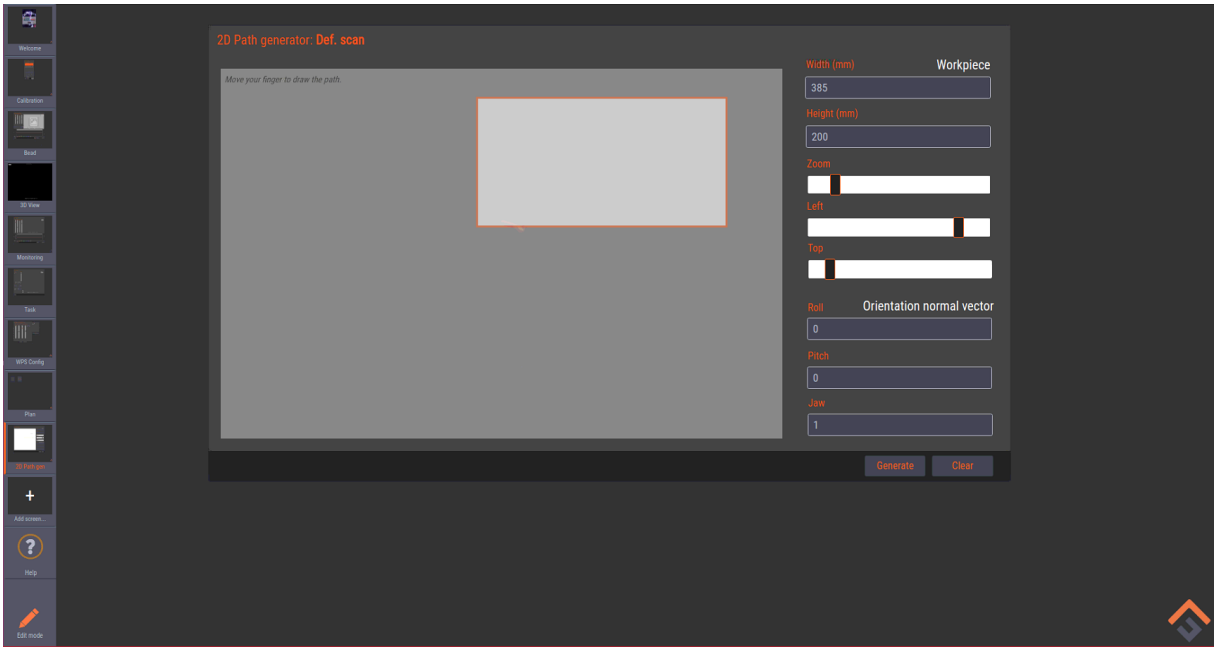


Image 8 2D Path generation screen in FlexGui 4.0

It is possible to generate 2D (x, y) path from FlexGui 4.0 (Image 8). It is necessary to set up the size of the workpiece by giving its dimensions, what will be the reference for the generation. If a top view camera is installed to the system, its image will be visible behind the light rectangle to help the path drawing.

Moving the mouse cursor on the screen will draw the path, which will be translated into model coordinates.

Pressing the Generate button will translate and add the path to the selected task and upload it to the robot.

4.4.7 Monitoring

On the Monitoring screen (Image 9), the welding process can be tracked. You can change the welding parameters by using the scrollbars and see the selected welding and path parameters compared to each other.

Pressing the *Abort* button will immediately stop the robot movement and disables welding equipment.

Image 9 The Welding monitor screen with the camera image

The robot speed can be changed with the Torch speed scrollbar. On the scrollbar, you will see the current position of the robot on the path with the current welding parameters. The welding parameters value are shown on the Voltage/Amperage/Filler speed sliders reported by the WPS as a reference value.

On the top right corner of the camera, you can see that the simulation is on or off.

Pressing the **Start Weld** button, the WPS becames enabled in the framework and the welding process will start. The robot is commanded to move to the first path point, ignite the arc then go through the current weld bead. During welding, it is possible to abort the current movement by pressing the **Abort** button. It will terminate the arc and disable the WPS status. When the termination process is over, the robot will stay in position.

It is also possible to show or hide the selected path modifications by pressing their name under the chart, same as described on Image 6 (9).

4.4.8 3D View

This screen shows the whole virtual environment in your client (Image 10).

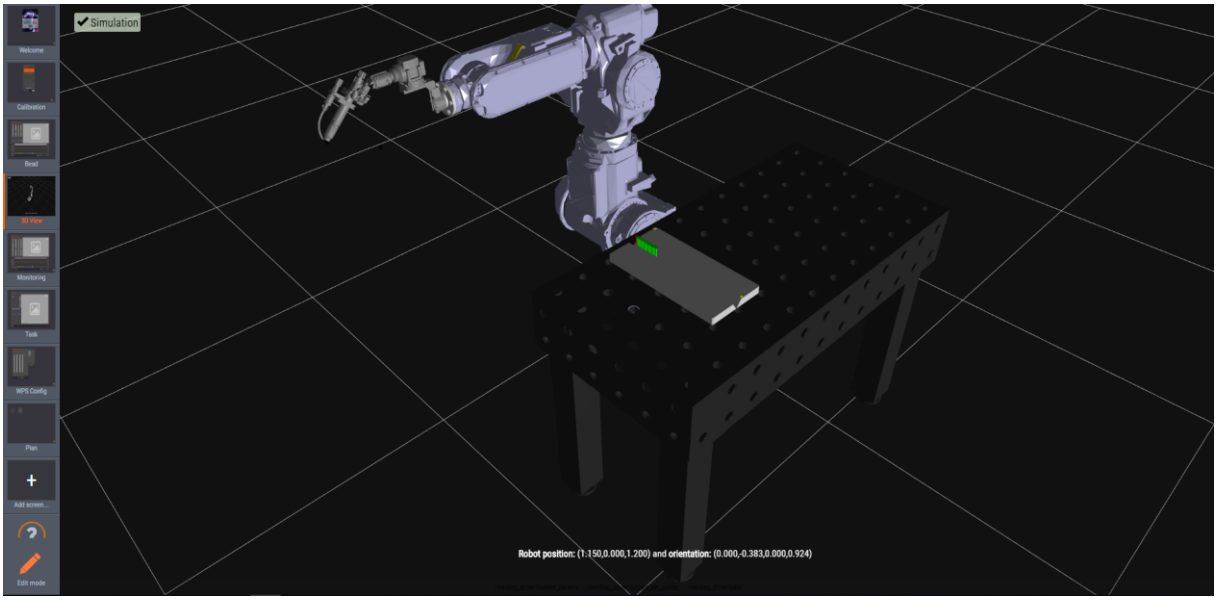


Image 10 The FlexGui 4.0 - 3D View of the virtual environment

5 Drivers

The ROSweld framework provides drivers for the devices in this section. The drivers are set up for the example project, but here you will receive the required information to customize them.

5.1 Robot

The robot drivers' task is to make a connection between the robot and the ROS framework. Each NACHI, OTC and Hyundai drivers include a robot and the ROS side program. The communication protocol is UDP. The following services are provided by the drivers:

move_along	moves along the stored path with the robot's current speed
move_pose	moves to a given pose
abort	aborts the current move
store_poses	stores the poses on the robot
set_speed	sets the current speed
move_between	moves from one step to another along the path

The drivers are providing the following information (rosweld_drivers/RobotState):

pose	including the current position and orientation
joints	joint values in radians
euler_angles	roll, pitch, yaw
step	the current step index on the path
storedPoses	number of the stored poses
robotProgramState	state of the robot program

mode	robot mode, such as teaching, playback or high-speed teach
isMoving	true if the robot is moving
isSimulation	this bit is not reported directly by the drivers but forwarded by the ROSweld core proxy. It shows, whether the current topic is the real robot or a simulation
error	contains any error message in case of any error

5.1.1 NACHI

The driver communicates with the NACHI robot over UDP. It is necessary to install two user tasks:

USERTASK-A.000	commanding / controlling the robot (port: 8000)
USERTASK-A.001	Receiving the status updates (port: 8001)

The nachi_robot.py driver starts two UDP connections to the robot. It is necessary to set up the robot's IP address and the two UDP ports (8000 and 8001 by default) and open the firewall if there is any. The used *rosparams* are the following:

ip	IP address of your NACHI robot
update_port	update port on the robot (communication with USERTASK-A.001)
command_port	command port on the robot (communication with USERTASK-A.000)
robot_name	name of your NACHI robot; prefix for the offered services and topics

All the features above are supported by the NACHI driver.

5.1.1.1 6-7 Axis configuration

The given driver is designed to operate with 7-axis NACHI robot. If the application of a 6-axis robot is necessary, only the robot side files need to be modified:

USERTASK-A.001:	Remove the 7th joint value (SETREAL 4, SYSTEM! (110), 76) and decrease the size of the package with 4.
USERTASK-A.000:	When storing to pose parameters, you must set a constant 0 for the REDUNDANT_JOINT (VAR.INC REDUNDANT_JOINT, V1! -> REDUNDANT_JOINT,0)
<ROBOT_TYPE>-A.009:	You have to remove the second mechanism from all move commands, MOVEX (,M2J ….)

5.1.1.2 Starting the NACHI driver

You can start the driver with the given example launch file: mr20.launch.

```
roslaunch ROSweld_drivers nachi.launch ip:=<your_robot_ip>
robot_name:=<robot_name>
```

5.1.2 Hyundai

The driver communicates with the Hyundai robot over UDP. It is necessary to install two JOBs and set up all the IP addresses correctly.

For the right installation, please follow the tutorial below (7.1 ROS Hyundai driver and simulator installation manual).

The hyundai_robot.py driver starts one UDP connection for commanding the robot and one UDP server to receive the updates. It is necessary to setup the robot's IP address and the two UDP ports (12340 and 12350 by default) and open the firewall if there is any. The used *rosparams* are the following:

ip	IP address of your Hyundai robot
update_port	update port on the robot (communication with 0002.JOB)
command_port	command port on the robot (communication with 0001.JOB)
robot_name	name of your Hyundai robot; prefix for the offered services and topics
local_port	to be able to receive feedback after commanding the robot, it is necessary to setup the local communication port, by default: 12349

The ports mentioned above must match the ones in the Hyundai project: 0001.JOB and 0002.JOB.

0001.JOB	handles the commands sent to the robot, ENET1.IP is the local address of the robot itself. ENET1.RPORT is the remote port, which must match with command_port.
0002.JOB	updates the robot state. It is necessary to set JOB 2 as JOB 1's subtask (in the Multi-tasking state window) to be able to run with JOB 1 in parallel. ENET2.IP is the ROSweld server's IP address and ENET2.RPORT must match with update_port.

To set up the network parameters, you must check config.hrs as well. The defined IpAddrExt must conform with the same subnetwork, as the ROSweld server IP and robot IP otherwise.

5.1.2.1 Starting the Hyundai driver

You can start the driver with the given example launch file: hyundai.launch.

```
roslaunch ROSweld_drivers hyundai.launch ip:=<your_robot_ip>
robot_name:=<robot_name>
```

5.1.3 MoveIt!

ROSweld contains a general MoveIt! driver, offering the same interface as the NACHI driver. The driver is only for visualization. In case the equipment is available, the robot controller can be included and the driver will control the physical robot as well.

The used *rosparams* are the following:

robot_name	is the name of the MoveIt! node; prefix for the offered services and topics
move_group_name	the name of the selected move group

5.1.3.1 Starting the MoveIt! driver

The driver started with the given example launch file: moveit.launch.

```
roslaunch ROSweld_drivers moveit.launch ip:=<your_robot_ip>
```

5.2 Sensors

5.2.1 Laser scanner

ROSweld provides the interface for 2D/3D Laser line profile scanners. The supported scanners are the MEL M2-iLAN and MicroEpsilon series.

The basic configurations for each scanner are the following:

laser_name node name and prefix for each topic and service provided by the driver, by default: laser_driver

laser_tcp name of the tool link, by default: scanner

The driver publishes the following topics and services to ROS:

start service to enable the laser scanner to run in two modes: continuous reading or robot triggered. During continuous reading, the driver triggers the scanner immediately after a reading is complete and stored. During the robot triggered measurement, the driver triggers the laser scanner when the robot state is updated.

stop service to disable the laser scanner

laser_scan topic which contains the latest available laser scanner measurement transformed into the previously set frame as a PointCloud2 message

Each launch file includes laser_dumper.py, which can dump the scan data to a text file. To set up dumping, you can use the following ROS parameters:

trigger_robot_name save the robot's current position

dump_size split the files after N measure

log_dir save the files to a selected folder

5.2.1.1 MEL M2-iLAN Laser scanner

ROSweld provides the driver for the MEL M2-iLAN laser scanner. The driver is communicating with the laser scanner via UDP.

Connection parameters:

laser_ip IP address of the laser scanner

laser_port UDP port of the laser scanner

You can download the factory documentation of the scanner:

http://www.sensores-de-medida.es/uploads/hb-m2-ilan-udp-e.pdf

During the initialization phase, when starting the driver, the following settings are done on the laser scanner:

- Setting up manual trigger mode
- Reset FIFO
- Requesting information

To start the driver, please use the following command:

```
roslaunch ROSweld_drivers mel.launch
```

5.2.1.2 MicroEpsilon scanner

You can control the MicroEpsilon scanners with ROSweld. To be able to use the package, first, you must run the install script inside ROSweld_drivers package.

To install the required packages, the following script must be run:

```
cd ~/catkin_ws/src/ROSweld_drivers/uepsilon_sdk #or find your ROSweld_drivers folder

sudo ./install.sh
```

The package will install *aravis 5.10* (you can use the latest as well) and install the latest 0.2.0 MicroEpsilon Linux SDK x86-64 version. If you would like to update the 0.2.0 SDK, all you have to do is to download it from:

https://www.micro-epsilon.com/2D_3D/laser-scanner/Software/downloads/?sLang=en

and copy the content to the SDK folder inside the package. It will eventually be necessary to update the install script if the hierarchy of the SDK changes.

The scanner uses the following ROS parameters to setup the connection:

serial — you must define the serial number of your laser scanner. If the specified scanner is not presented, the driver will use the first available one.

resolution — the resolution to use for your scanner

You can start the driver with the following command:

```
roslaunch ROSweld_drivers uepsilon.launch
```

5.2.2 Welding camera

The ROS framework provides the driver for UVC compatible devices, such as AV.io HD USB3.0, which support DVI/HDMI input, but with the right cable, can work with s-video as well.

The **libuvc_camera** driver can be used to grab the frames from the welding camera and forward it to ROS.

The driver publishes images to ROS, which can be recorded with *rosbag* or played with *web_video_server* or *mjpeg_server*.

5.2.3 IP camera

ROSweld IP camera driver is based on the already existing ROS IP camera package: https://github.com/ravich2-7183/ip_camera

The following new features are implemented to increase quality and reliability:

Authentication — most of the IP cameras require a username and password to authenticate the user, use *--user* and *--password* to set this up

Max FPS	if you wish to limit the maximum FPS, you can do it with the *--fps* flags, while the default used by the driver is 10 FPS
Topic	you can set the topic to publish with the *--topic* flag
Automatic restart	if the camera is not updating for 5 seconds, the driver will stop it and the camera supervisor application will restart it

You can start a camera node with the following command:

```
roslaunch ROSweld_drivers camera.launch
```

5.3 WPS driver

The Welding Power Source (WPS) drivers communicate with the hardware over UDP protocol and consists of two parts:

On a PLC/Robot there is a UDP connector, which collects all of the necessary information about the WPS and communicates with the ROS part driver, which publishes the information to ROS and controls the WPS. The implemented WPS are for OTC and Fronius. The OTC driver control is an integrated solution; therefore, the communication is carried out through the robot controller. The Fronius WPS interface is attached to a PLC, that translates the messages and publishes it to the network.

The ROS side driver offers the following functionality:

edit_config:	edits a selected welding job configuration locally or depending on the WPS, on the device itself
set_job_number:	sets the currently selected job on the WPS
set_params:	sets the currently used setpoint values
arc_start:	starts the arc
arc_stop:	stops the arc
jobs:	topic with the available jobs
current_set_points:	topic with the current setpoint values
current_params:	topic with the actual values

The drivers are offering the following parameters to receive and to set:

- Voltage
- Current
- Wire-feed speed
- Arc length
- Mode
- Auto job update
- Is arc on

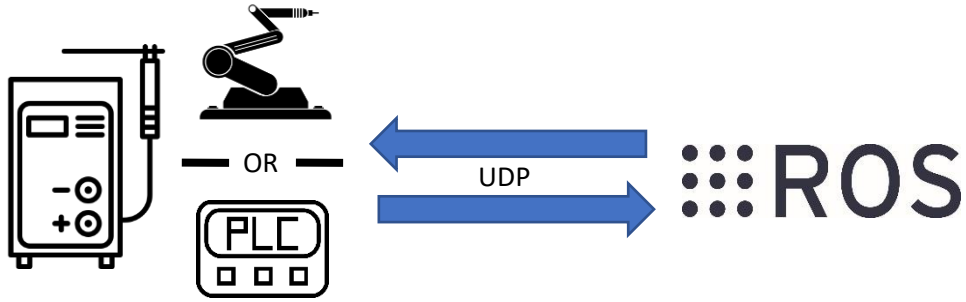
To use the WPS with ROS, both sides must be installed. The ROS side is the same for both. The selected driver must be run with python: [otc|fronius]_wps.py. Following ROS parameters must be set:

wps_ip	this is the IP address of your robot/PLC

wps_port this is the selected UDP port

When you start the driver, it will try to connect to the WPS and advertise the services and topics to ROS.

Please note: not all drivers can update and receive the jobs from the WPS itself, it depends on whether the interface is open or not.



5.3.1 OTC

The WPS side driver is running on a NACHI/OTC robot as a user task. To install the robot side driver, the user has to copy the user task to the robot, see NACHI documentation (Chapter 4.1):

https://www.globalindustrial.com/site/images/Nachi-PDFS/TFDEN-012-001_Robot_language.PDF

If the user task program is compiled, you can enable and execute it in the user task monitor.

Please note: to be able to use the driver, to compile it and to run it, you will require the Integrator user level on the robot.

If the driver is running, it will be ready to use from the ROS side.

You can modify the selected UDP port and variables to be used by the driver, if you would like to use them for other purposes. In that case, please modify the *WPSIO.inc* file and search for the PORT or the variables you wish to change. Please remember to use the same PORT number on both, ROS and WPS side.

To setup the OTC welding equipment, please see the manual:

1L21700F-E-27_ArcWelding.pdf

Chapter 2: Basic settings for Arc welding - setup of the robot-WPS connection, registering WPS, setup the necessary software environment

1L21700B-E-11_Setup.pdf

Chapter 6: Checking welding characteristic data, allocate I/O-s for the welding operation

6 Acknowledgements

The Focused Technical Project "ROSWELD – ROS based framework for planning, monitoring and control of multi-pass robot welding" is co-financed by the EU project ROSIN (www.rosin-project.eu) and ROS-Industrial initiative (www.rosindustrial.eu).

The FTP's coordinator and performer is PPM Robotics AS, Norway. The project is supported by two further industrial partners from Norway: Mechatronics Innovation Lab AS (http://www.mil-as.no) and Rainpower Norge AS (www.rainpower.eu).

Contact

Prof PhD Trygve Thomessen

Managing Director and ROSWELD Coordinator

PPM Robotics AS

Leirfossveien 27

7038 Trondheim

Norway

+47 92 23 74 35

info@ppm.no

7 Appendix

7.1 ROS Hyundai driver and simulator installation manual

7.1.1 Installing the Hyundai Simulator

The Hyundai simulator is called HRSpace. You can download it by using the link below:

https://www.dropbox.com/s/xyn3ys1t3vy8q1t/HRSpace_v3_83b1.zip?dl=1

Note, that Windows is required to install this software. Please start the installation. Please make sure, that you also install the Visual C++ 2008 Redistributable package. If this package is missing, the simulator won't start. The zip file downloaded above contains the Korean version of this package, you might want to get the package that fits your operating system language.

7.1.2 Installing the sample config

We have prepared a sample config, so it's easy to test the initial settings. Please download using the link below:

http://git.ppm.no/Repository/2eb91bcd-2def-4672-8b4e-22f73840972e/master/Download/hyundai

Extract the downloaded files anywhere on your computer. Then open the config.hrs file found in this folder using a text editor, like notepad or notepad++ and find the following line:

```
IpAddrExt=192.168.56.1
```

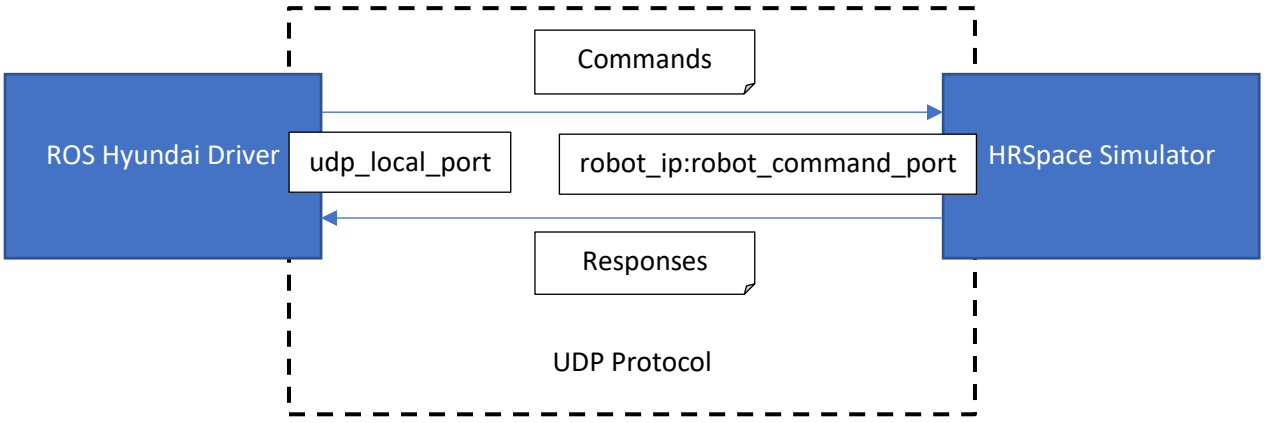
Please change the IP address to the IP of your own PC, where you are running this simulator. Save the file and close it.

Note: you can find your IP using the command:

```
ipconfig
```

In your command prompt.

7.1.3 Main program for command processing



The robot programs are stored inside the "robot" folder, next to the config file. Open the folder, and the 0001.JOB file in it. Look for the following lines:

```
REM This IP is the controller IP
ENET1.IP="192.168.56.1"
ENET1.RPORT=12349
ENET1.LPORT=12340
```

Modify the IP address to the IP of your PC, where the simulator is running. Also modify the ports, if you don't want to use the default configuration. RPORT means remote port, the port, where the ROS Hyundai Driver is listening (udp_local_port in rosparam). LPORT is the local port, where the simulator will listen (robot_command_port in rosparam).

Now look for the following lines:

```
REM This IP is the other party's IP
ENET1.IP="192.168.56.100"
```

And modify the IP to the address, where the ROS Hyundai Driver is running.

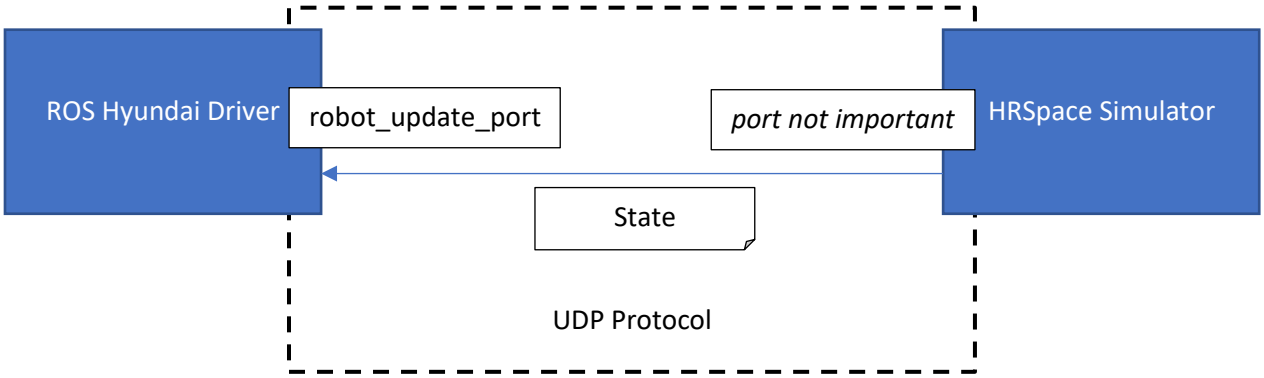
Save the file and close it.

Please note, that the default IP, where the ROS Hyundai Driver sends the data is " 192.168.1.143". If you want to use another address, use the following command to set it:

```
rosparam set robot_ip "x.x.x.x"
```

Please replace the x.x.x.x to your robot's IP address. This value is not saved, when ROS is restarted, please make sure, you save and load rosparams or include this value into your launch file.

7.1.4 State update program



Open now the 0002.JOB, and find the following lines:

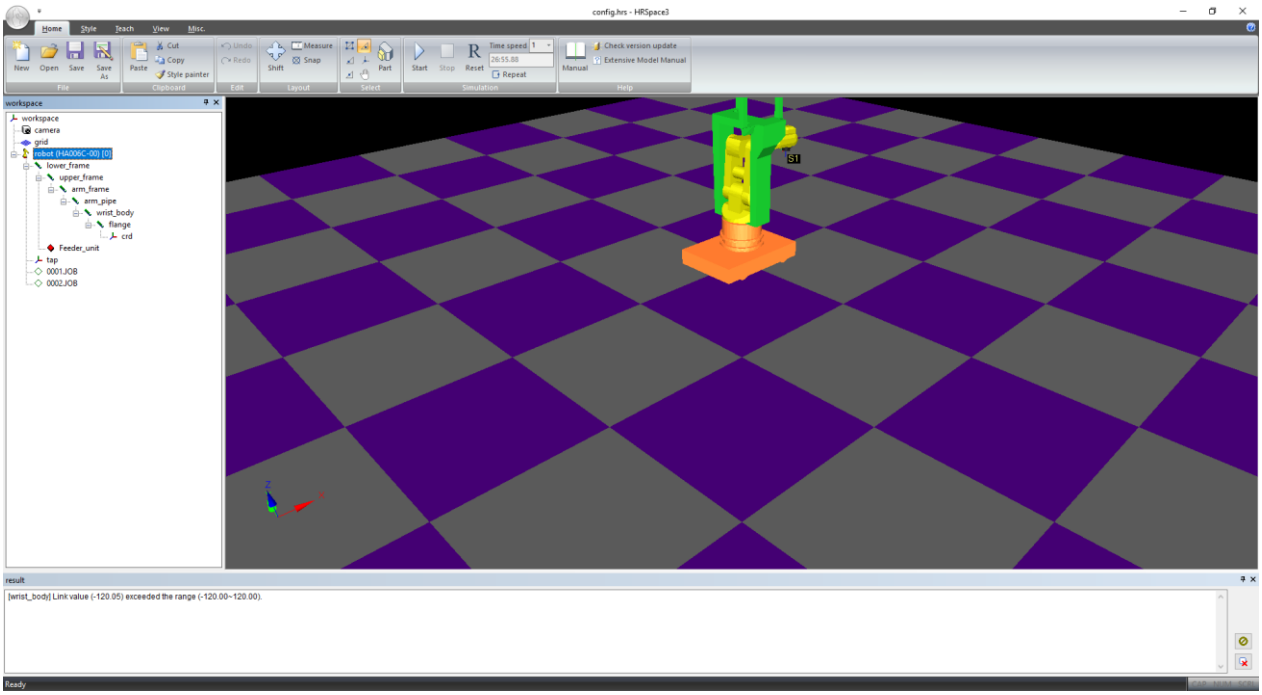
```
REM This IP is the other party's IP
ENET2.IP="192.168.56.100"
ENET2.RPORT=12350
ENET2.LPORT=12341
```

Modify the IP to the address, where the ROS Hyundai Driver is running. Also, you can modify the ports, if you don't want to use the default configuration. RPORT means remote port, the port, where the ROS Hyundai Driver is listening (robot_update_port in rosparam). LPORT is the local port, where the simulator will listen (the value is not important, because there is no message coming).

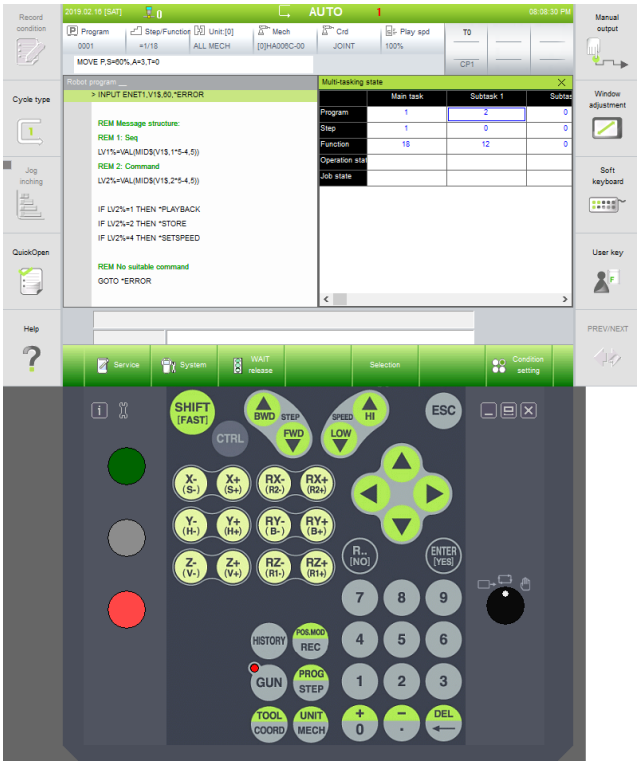
Save the file and close it.

7.1.5 Starting the simulation

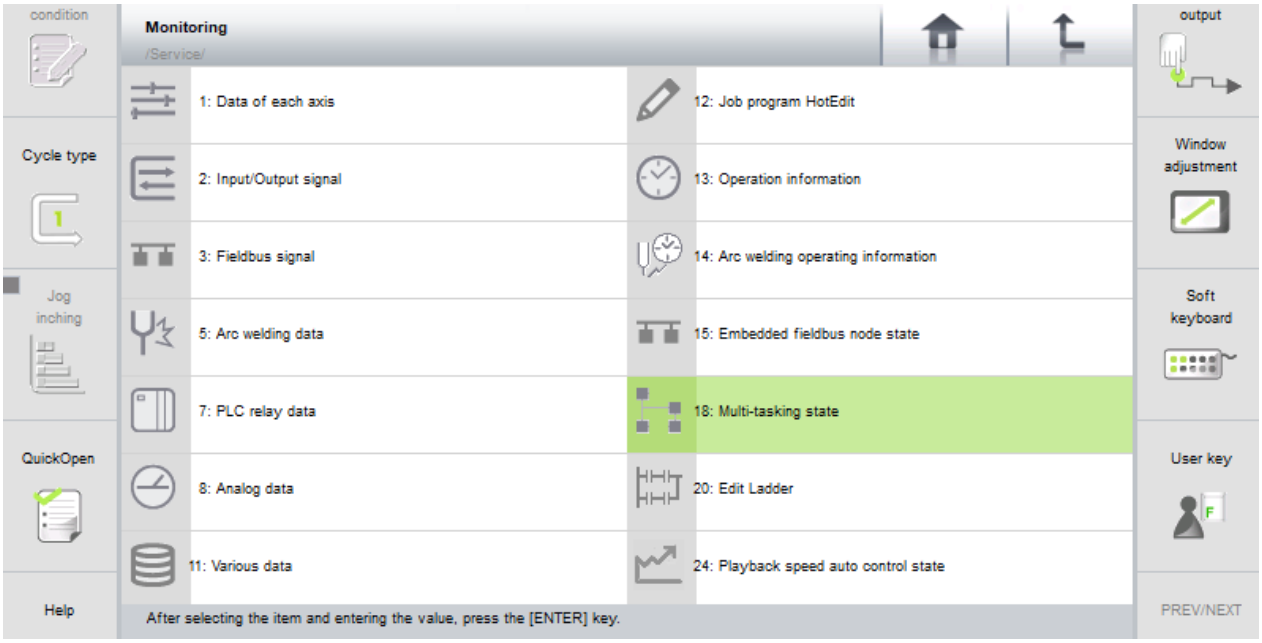
Now start the HRSpace, and open the config.hrs in it. You should see the following:



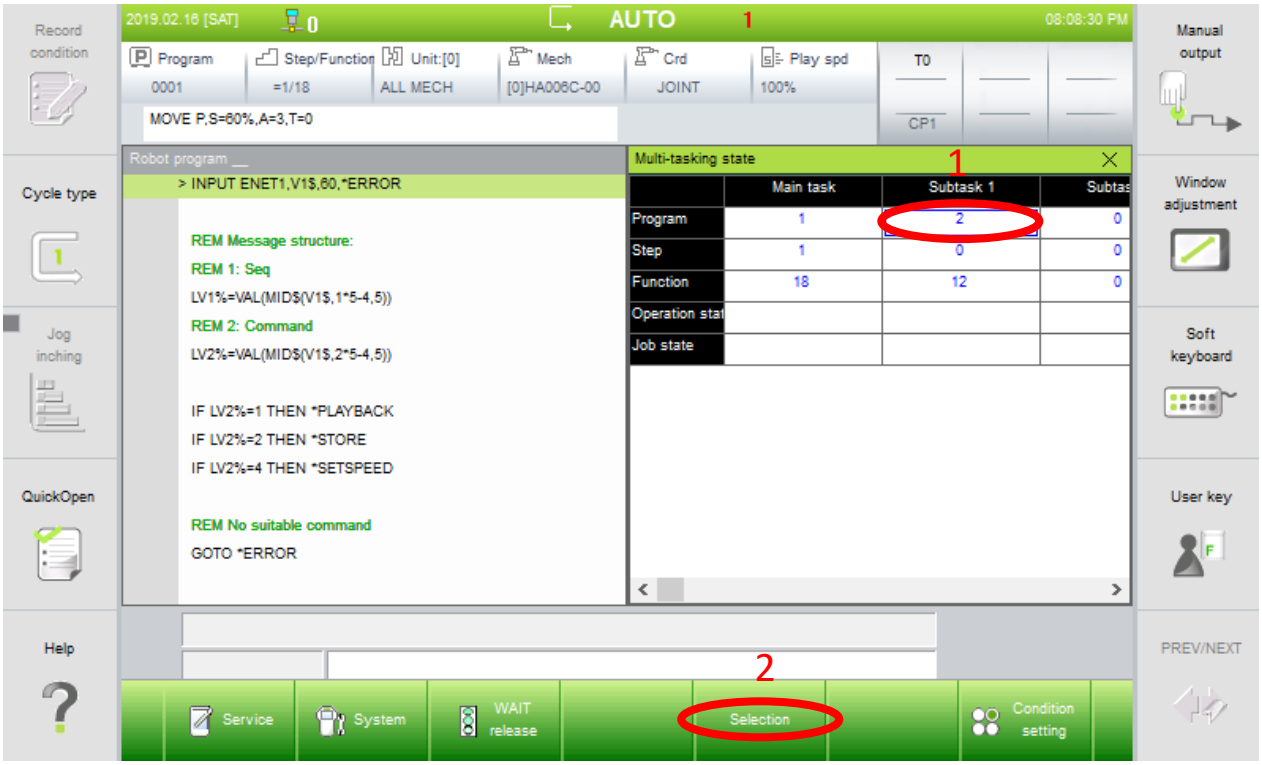
Start also the Hi5a virtual teach pendant, which will be used to control the robot. The teach pendant looks like this:



If you don't have the multi-tasking state panel on the right side, you need to enable it first by selecting the Service/Monitoring/Multi-tasking state:

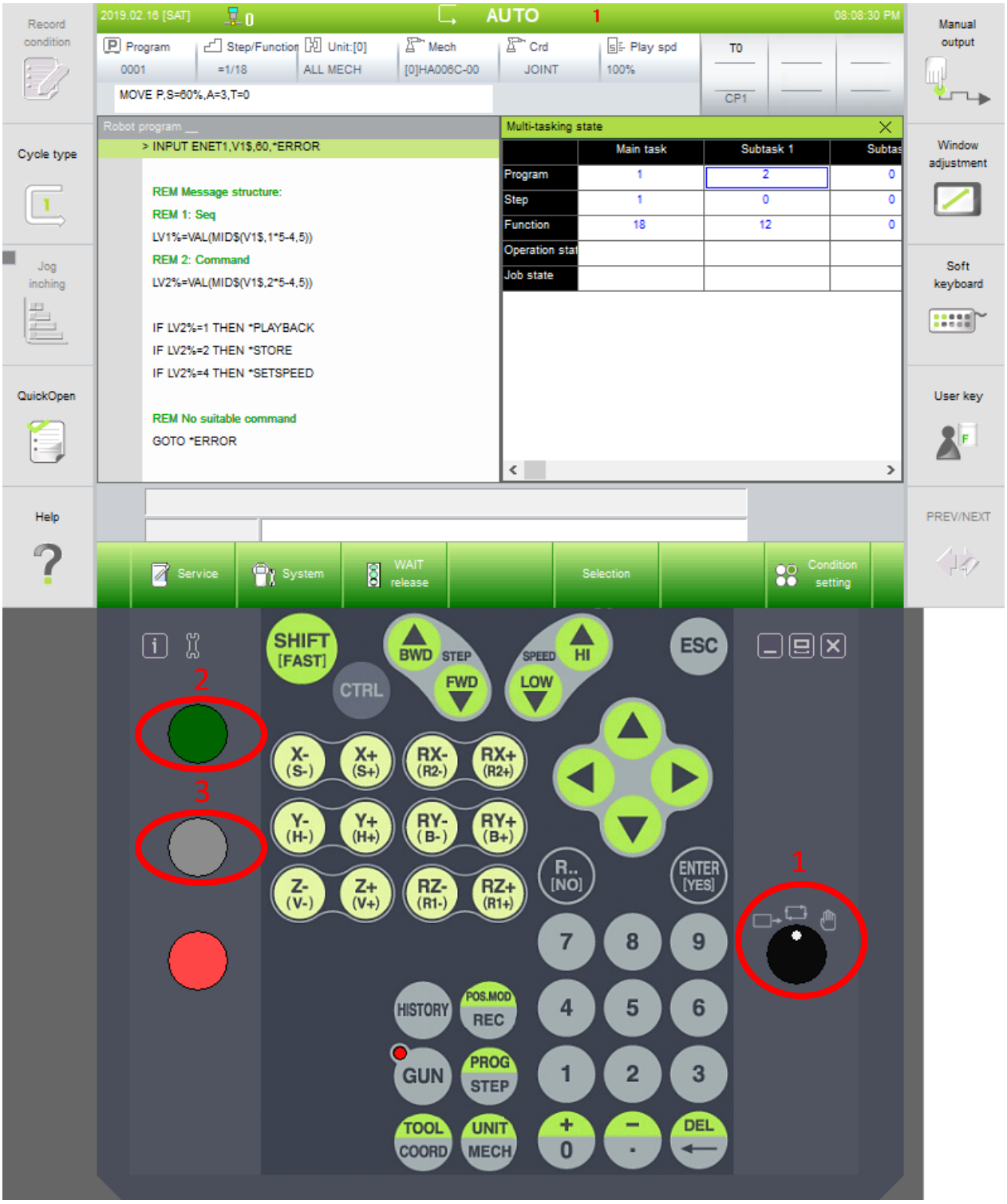


To select, which programs to run, simply click on the program of subtask 2, then the selection softkey:



Selecting the Main task will reset the subtasks, so select the main task first. Alternatively, you can use the main program selector on the soft keyboard, to select the main task.

To start both programs, switch to playback mode, start the motors, then start the execution:



If the program execution is on the line visible on the picture above (green), that means, it is waiting for a command.

7.1.6 Debugging

We collected a couple of examples, when the user might find difficulties, and possible steps to find the problem.

7.1.6.1 Ethernet device BIND failure

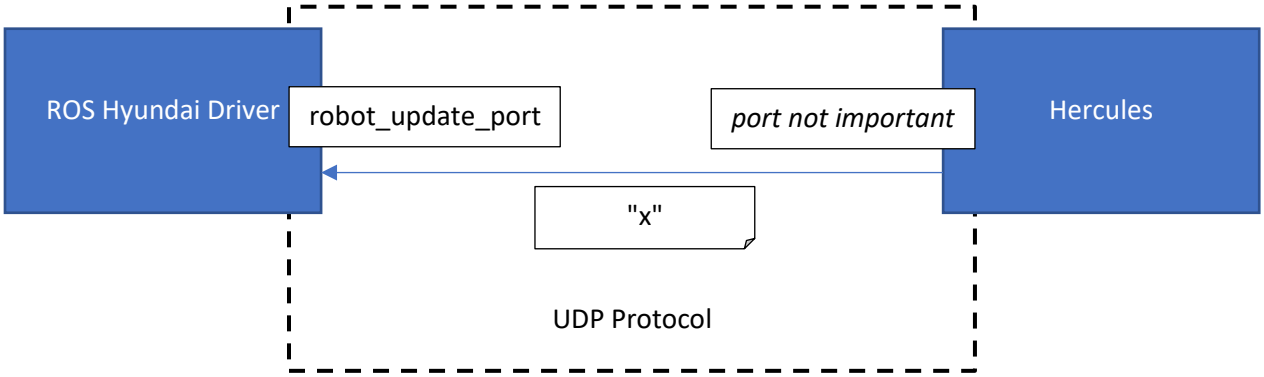
If the teach pendant shows the error message "Ethernet device BIND failure", that is printed by the update job, it means, that the simulator was unable to register the UDP port in the system. Please try to restart the simulator, then the system itself.

If this doesn't help, you can use the command:

```
netstat -a -p udp
```

To list all listening UDP ports, and find a possible collision.

7.1.6.2 Updates are not received by the ROS Hyundai Driver



You can start the ROS Hyundai Driver separately from the rest of the system, by navigating to the containing folder, then entering the command:

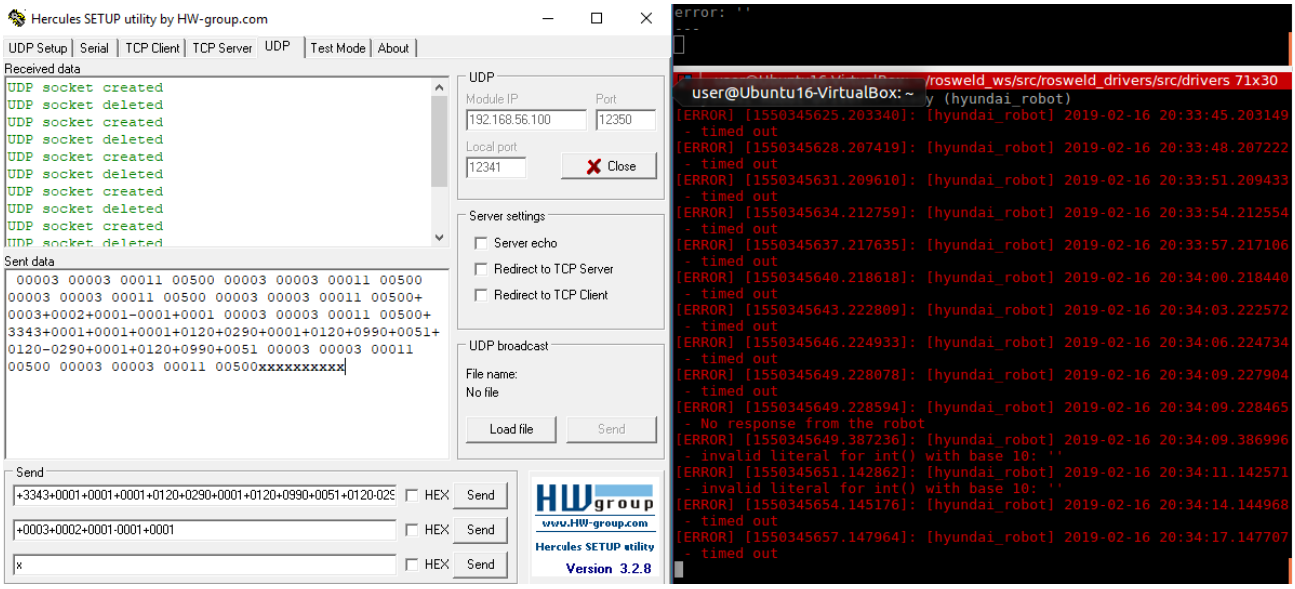
```
python hyundai_robot.py
```

This way, you will see all error messages. Now start the Hercules debugging utility, that you can download from the link below:

https://www.dropbox.com/s/wyztyxf50gu0j7l/hercules_3-2-8.exe?dl=1

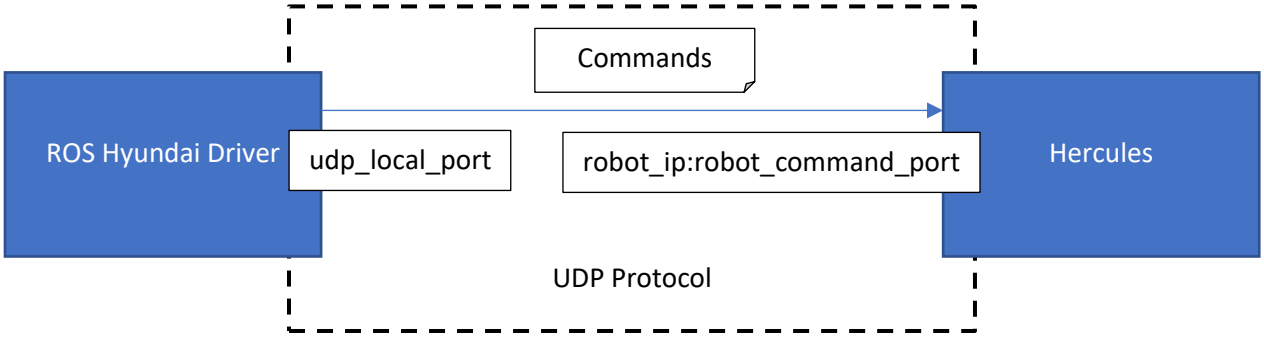
Now, select the UDP tab, enter the IP address of the ROS server into the Module IP field, the robot_update_port into the Port field. This is the target address. Also enter any port into the local port field.

Now use any of the Send buttons below, to send one character: "x". If the network is set up well, the ROS Hyundai Driver should output an error message: "Invalid literal…" on all presses.



If this happens, your Windows PC can reach the ROS PC. If not, please double check the state of your network: IPs, ports, firewalls.

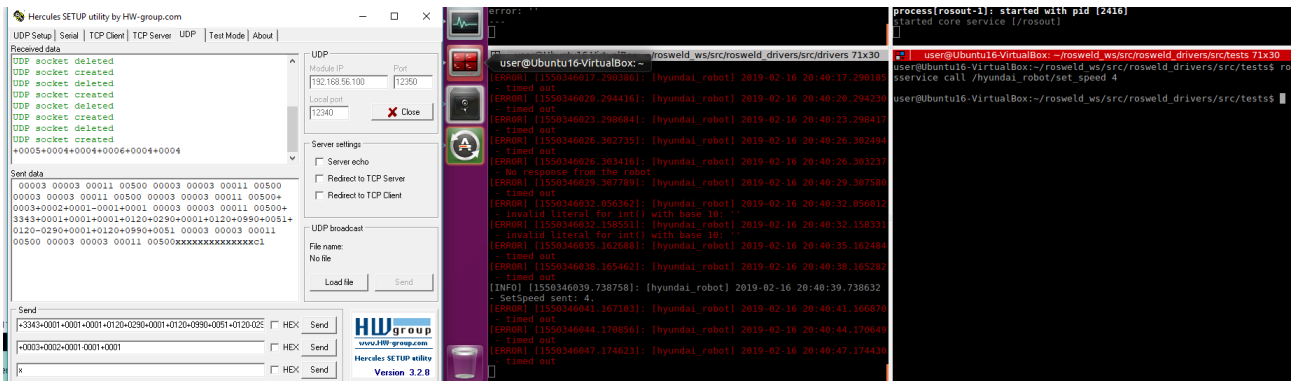
7.1.6.3 Commands are not received by the robot



Make sure, that the Hyundai simulator is not running, to avoid port binding collisions. Change the local port of Hercules to the robot_command_port, and press listen. Now, if you send a command from the ROS Hyundai Driver, using the following command:

rosservice call /hyundai_robot/set_speed 4

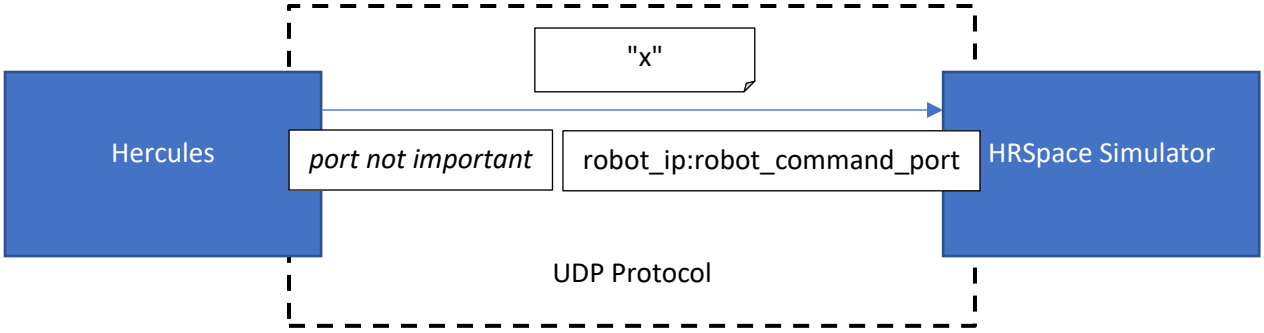
You should see the data being received in Hercules:



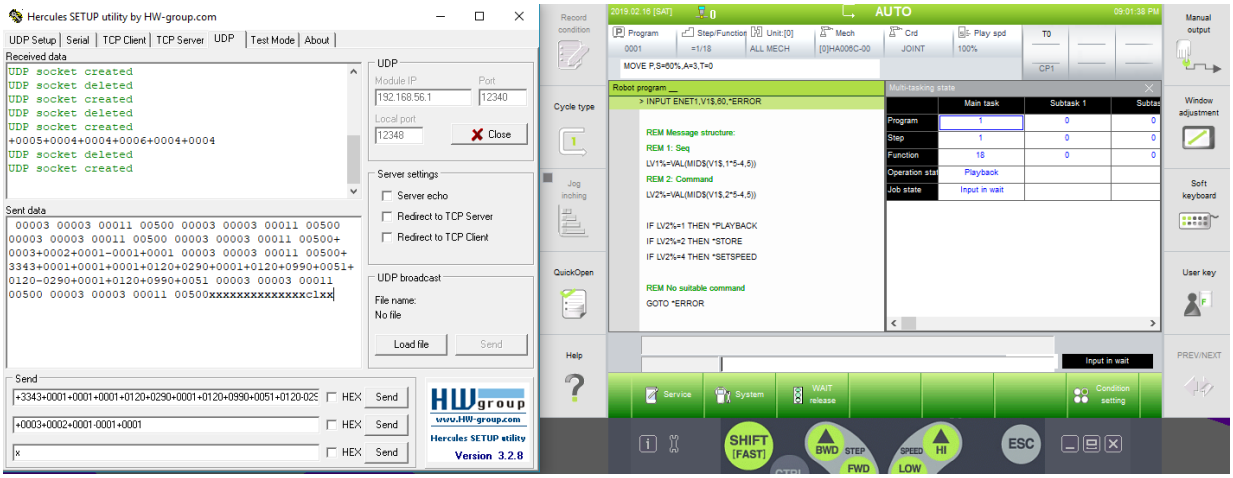
The data received means: [seq number = 5][command = 4 (setspeed)][speed = 4]

If you can get this far, the network connection from the ROS PC to the Windows PC works. If not, please double check the state of your network: IPs, ports, firewalls.

7.1.6.4 Hercules works with the ROS Hyundai Driver, but the robot doesn't



Now we can connect Hercules directly to the robot, to send commands.



Make sure, no Hercules is using the Local Port of the Robot Simulator. Start the simulator, and the virtual teach pendant. Start the execution of the Main JOB 0001. If the executed line in the simulator is the INPUT command, that means, that the robot is waiting for an incoming message.

In Hercules, set the Module IP to the address of the robot (in the simulator, the address of the PC), and the port to the robot_command_port. The local port doesn't matter, as long as it doesn't collide with anything else.

Now send an "x" character using one of the Send buttons in Hercules to the robot. If the execution moves from the INPUT command every time, you press the Send button, then the robot can receive the network messages.

If you get this far, it means, that the robot can receive local network messages. If not, please try to restart the simulator, check your firewall and network settings.