
RVL UR-Robotiq Integrated Driver Documentation

Release 0.0.1-alpha

Minh Tram

Mar 30, 2022

CONTENTS

- 1 Quickstart 3**
 - 1.1 Universal Robot Setup 3
 - 1.2 Docker Environment Setup 3
 - 1.3 Inside the container 4
 - 1.4 Debugging 6
- 2 RVL UR-Robotiq API 7**
 - 2.1 Imports 7
 - 2.2 Modules Quick Access 7
 - 2.3 Full Modules Documentation 8
- 3 References 19**
- 4 License 21**
 - 4.1 Apache 2.0 Boilerplate 21
- 5 Changelog 23**
 - 5.1 Releases 23
 - 5.2 Tasks 23
- Python Module Index 25**
- Index 27**

RVL UR-Robotiq Integrated Driver is a wrapper for the existing ROS services and topics provided by Universal Robots (UR) ROS Driver with additional support for Robotiq gripper (currently support 2F) directly attached to the UR tool port.

This driver also provides additional function for actuating the gripper in various modes (grasp on open/close, specific jaw gap, etc.) as well as integrating MoveIt! motion planning that tracks the entire setup, UR and Robotiq gripper.

At the time of writing, the driver is working on our setup on both Windows 10 (Version 10.0.19044 Build 19044) and Ubuntu 20.04.

QUICKSTART

1.1 Universal Robot Setup

1.1.1 Setup RS485 URCap and socat

Follow the [step by step instructions](#) provided by Universal Robots to allow tool communication via a socket on port 54321.

The control box file system can be accessed via `ssh root@[ROBOT-IP]` with the default password `easybot`. You should see the internal UR prompt, something along the line of

```
Universal Robots A/S Linux image
```

```
Production image
```

```
root@ur-[serial]:~#
```

`socat` can be installed from there from source or using a package manager. You may want to use the attached `socat-robotiq-gripper.service` so `socat` starts every time on system power on.

To setup the service file, just put it in `/etc/systemd/system/` and run

```
$ sudo systemctl enable socat-robotiq-gripper.service
```

which will start the `socat` service on boot.

1.2 Docker Environment Setup

1.2.1 Installing Docker

```
$ sudo apt-get install docker.io
```

or follow Docker on WSL 2 backend installation guide at [Get Docker](#).

Note: If you are on Linux, be sure to follow [Docker Post-Installation Steps](#).

1.2.2 Clone the Repository

```
$ git clone --recurse-submodules https://github.com/robotic-vision-lab/UR-Robotiq-Integrated-Driver
```

1.2.3 Build or Pull the Docker Image

You can build the image using the Dockerfile already included in the repository

```
$ cd UR-Robotiq-Integrated-Driver/docker
$ docker build -t rvl-ur-robotiq-driver -f Dockerfile.noetic .
```

or you can pull it directly from Docker Hub with `docker pull mqt0029/rvl-ur-robotiq-driver:latest`. If you pulled from Docker Hub, be sure to retag your image to `rvl-ur-robotiq-driver:latest`

```
$ docker tag mqt0029/rvl-ur-robotiq-driver:latest rvl-ur-robotiq-driver:latest
```

or update the image name in the launch script accordingly.

1.2.4 Creating the ROS Container

Note: Be sure to check your image name before running the script.

The included `[windows_]launch_docker_container.sh` is a useful script in creating and accessing docker containers created from the built/pulled image. Simply run

```
$ sh launch_docker_container.sh
```

First run will create the container if it is not already existed. Subsequent run will attached the terminal to the docker container via `bash`. At this point, you should see the following prompt:

```
[...]
root ~/catkin_ws
>
```

1.3 Inside the container

With our specific setup of running UR5e and Robotiq 2F-85 gripper, we have customized the accompanying `xacro` (hence, also includes generated `urdf`) to match our specific machine description. You may want to modify the launch file to use the correct files for your setup.

While the base path is commonly `[...]/catkin_ws/src`, this is not always the case. Paths will be relative to where you place your ROS packages.

1.3.1 Robot Xacro and URDF

The included ROS package `rvt_robot_description` contains an example of our robot setup. The file `[...]/rvt_robot_description/xacro/rvt_ur5e.xacro` can be a good place to start and components should be fairly straightforward to swap in and move around.

1.3.2 Universal Robots Driver Bringup

The `ur_robot_driver` package launch file has been modified to work with the tool port gripper and our specific setup. You can use the launch file `[...]/rvt_ur_remote_dashboard/launch/rvt_ur5e_bringup.launch` as a reference, and modify it to fit your setup. The important section is highlighted below.

```
<arg name="robot_ip" default="192.168.1.147" doc="IP address by which ..."/>
<arg name="robot_description_file" default="$(find rvt_ur_remote_dashboard)/launch/load_
↳ rvt_ur5e.launch" doc="Robot description launch file."/>
<arg name="kinematics_config" default="$(find rvt_ur_remote_dashboard)/configs/ur5e_
↳ calibration.yaml" doc="Kinematics config file used ..."/>

<!-- always start the UR side socat -->
<arg name="use_tool_communication" default="true" doc="On e-Series robots tool_
↳ communication can be enabled with this argument"/>
```

Warning: Double check your calibration file is correct or [retrieve it accordingly](#).

Warning: Make sure your robot description is correct! This is crucial to generating your own URDF and SRDF that MoveIt! uses to plan all subsequent motion!

1.3.3 Building and Running ROS

A few aliases has been provided to quickly setup ROS dependencies and paths. `bash` will also automatically source `devel/setup.bash`.

```
$ run_rosdep
[...] installing dependencies

$ rebuild_catkin
[...] building catkin workspace

$ roslaunch <your_stuff>
```

1.4 Debugging

Since this driver is built inside a Docker container, the most common issue is networking. A good place to start is if all the ports (or network interface) is exposed properly, the robot IP is correct, and there is nothing interfering with ROS communications both ways. Otherwise, the host OS should not be a problem.

However, should any issues arise, please [open an issue on our GitHub repository](#).

RVL UR-ROBOTIQ API

Due to how ROS structured Python source code and workspace setup, the module/package terminology usage might be confusing here.

2.1 Imports

```
from rvl_robotiq_controller.RobotiqController import Robotiq2FController
from rvl_ur_remote_dashboard.URRemoteDashboard import URRemoteDashboard
from rvl_ur_motion_planner.URMoveitCommander import URCommander
```

These are the primary modules and classes that covers most of the use case. The supporting functions might provide additional insights and is accessible as well.

Robotiq2FController is a wrapper to control Robotiq 2F grippers.

URRemoteDashboard is a wrapper for handling services and status monitoring as described on [ROS Interface page in UR Driver Repository](#).

URCommander is a wrapper for MoveIt! with some function built-in (pose goal, joint goal, etc.). Additional features are planned in future releases.

2.2 Modules Quick Access

These are links to quickly access modules mentioned under Import section. Otherwise, the full documentation of the entire driver is in the next section. Documentation will continue to be updated and covers more modules as the driver is being developed.

[rvl_robotiq_controller.RobotiqController.Robotiq2FController](#)

[rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard](#)

[rvl_ur_motion_planner.URMoveitCommander.URCommander](#)

2.3 Full Modules Documentation

2.3.1 Gripper Control

rvl_robotiq_controller ROS Package

rvl_robotiq_controller package

Submodules

rvl_robotiq_controller.Robotiq2FSupport module

rvl_robotiq_controller.Robotiq2FSupport.generate_2f_status_from_binary(binary)

rvl_robotiq_controller.Robotiq2FSupport.generate_binary_command_from_2f_msg(message)

rvl_robotiq_controller.Robotiq2FSupport.raw_to_rad_2f140(raw)

rvl_robotiq_controller.Robotiq2FSupport.raw_to_rad_2f85(raw)

rvl_robotiq_controller.Robotiq3FSupport module

rvl_robotiq_controller.RobotiqController module

```
class rvl_robotiq_controller.RobotiqController.Robotiq2FController(stroke: int, default_force: int = 100, default_speed: int = 100, initialize: bool = False, startup_reset: bool = False, calibrate: bool = False, bypass_power: bool = False)
```

Bases: object

```
__init__(stroke: int, default_force: int = 100, default_speed: int = 100, initialize: bool = False, startup_reset: bool = False, calibrate: bool = False, bypass_power: bool = False) → None
```

Controller for basic Robotiq 2F Gripper operation.

Parameters

- **stroke** (*int*) – Stroke of the gripper. 2F grippers must use either 85mm or 140mm.
- **default_force** (*int*, *optional*) – Default gripping force. Defaults to 100.
- **default_speed** (*int*, *optional*) – Default gripping speed. Defaults to 100.
- **initialize** (*bool*, *optional*) – Register ROS publisher and subscriber on object instantiation. Defaults to False.
- **startup_reset** (*bool*, *optional*) – Invoke internal reset on instantiation. Defaults to False.
- **calibrate** (*bool*, *optional*) – Adjust binary limits of gripper. Defaults to False.
- **bypass_power** (*bool*, *optional*) – Ignoring power state monitor. Useful when gripper not attached to UR tool port. Defaults to False.

Raises ValueError – Invalid stroke width, must be 85mm or 140mm

activate() → None
[summary]

auto_close(*alt_speed: Optional[int] = None, alt_force: Optional[int] = None, blocking: bool = True*) → None
Fully close the gripper or until obstructed.

Parameters

- **alt_speed** (*int, optional*) – Override internal speed settings. Defaults to None.
- **alt_force** (*int, optional*) – Override internal force settings. Defaults to None.
- **blocking** (*bool, optional*) – Wait until gripper motion is completed. Defaults to True.

auto_open(*alt_speed: Optional[int] = None, alt_force: Optional[int] = None, blocking: bool = True*) → None
Fully open the gripper or until obstructed.

Parameters

- **alt_speed** (*int, optional*) – Override internal speed settings. Defaults to None.
- **alt_force** (*int, optional*) – Override internal force settings. Defaults to None.
- **blocking** (*bool, optional*) – Wait until gripper motion is completed. Defaults to True.

block() → None
Convenient snippet for idle looping until gripper finishes motion.

calibrate() → Tuple[int, int]
Fully open and closes the gripper to record internal binary. Must be executed without any obstructions as this can dramatically influence gripper operations!

Returns lower and upper binary limit (default 0x00 to 0xFF or 0 to 255)

Return type (int, int)

compensated_publish(*command: rvl_robotiq_controller.msg._Robotiq2FCommand.Robotiq2FCommand, lag: float = 0.1*)

Publish with a small wait between message for ROS topics to cope with the refresh rate of the gripper.

deactivate() → None
Deactivate the gripper. Can also be used to clear the reset bit.

grasp_hard(*opening: bool = False, alt_speed: Optional[int] = None, blocking: bool = True*) → None
Hard grasp preset, grasping with force set to maximum or 255 (approximately 235 N).

Parameters

- **opening** (*bool, optional*) – Grasp in the opening direction i.e. internal grasp. Defaults to False.
- **alt_speed** (*int, optional*) – Override internal speed settings. Defaults to None.
- **blocking** (*bool, optional*) – Wait until gripper motion is completed. Defaults to True.

grasp_medium(*opening: bool = False, alt_speed: Optional[int] = None, blocking: bool = True*) → None
Medium grasp preset, grasping with force set to 128 (approximately 128 N).

Parameters

- **opening** (*bool, optional*) – Grasp in the opening direction i.e. internal grasp. Defaults to False.
- **alt_speed** (*int, optional*) – Override internal speed settings. Defaults to None.

- **blocking** (*bool*, *optional*) – Wait until gripper motion is completed. Defaults to True.

grasp_soft (*opening: bool = False, alt_speed: Optional[int] = None, blocking: bool = True*) → None
Soft grasp preset, grasping with force set to 1 (approximately 20 N).

Parameters

- **opening** (*bool*, *optional*) – Grasp in the opening direction i.e. internal grasp. Defaults to False.
- **alt_speed** (*int*, *optional*) – Override internal speed settings. Defaults to None.
- **blocking** (*bool*, *optional*) – Wait until gripper motion is completed. Defaults to True.

grasp_soft_regrasp (*opening: bool = False, alt_speed: Optional[int] = None, blocking: bool = True*) → None

Soft grasp preset, grasping with force set to 1 (approximately 20 N) with Re-Grasp enabled.

Parameters

- **opening** (*bool*, *optional*) – Grasp in the opening direction i.e. internal grasp. Defaults to False.
- **alt_speed** (*int*, *optional*) – Override internal speed settings. Defaults to None.
- **blocking** (*bool*, *optional*) – Wait until gripper motion is completed. Defaults to True.

inps_to_raw_speed (*inps: Union[int, float]*) → int
Convert inches/s to raw binary value [0-255]

is_holding() → bool
Returns true when the gripper is holding an object

is_moving() → bool
Returns true when the gripper is in motion

lbf_to_raw_force (*lbf: Union[int, float]*) → int
Convert lbf to raw binary value [0-255]

mmps_to_raw_speed (*mmps: Union[int, float]*) → int
Convert mm/s to raw binary value [0-255]

newton_to_raw_force (*newton: Union[int, float]*) → int
Convert N to raw binary value [0-255]

open_gripper (*value: Union[int, float], alt_speed: Optional[int] = None, alt_force: Optional[int] = None, unit: str = 'mm', blocking: bool = True*) → None
Open/Close the gripper to specified gap between the gripper pads.

Parameters

- **value** (*Union[int, float]*) – Width of the jaw opening.
- **alt_speed** (*int*, *optional*) – Override internal speed settings. Defaults to None.
- **alt_force** (*int*, *optional*) – Override internal force settings. Defaults to None.
- **unit** (*str*, *optional*) – Unit of measurement. Defaults to 'mm'.
- **blocking** (*bool*, *optional*) – Wait until gripper motion is completed. Defaults to True.

open_in_to_raw (*inches: Union[int, float]*) → int
Convert opening in inches to raw binary value [0-255]

open_mm_to_raw (*mm: Union[int, float]*) → int
Convert opening in mm to raw binary value [0-255]

power_monitor_callback(*msg*)

Callback for power subscriber

raw_force_to_lbf(*raw: int*) → float

Convert raw value [0-255] to lbf

raw_force_to_newton(*raw: int*) → float

Convert raw value [0-255] to N

raw_speed_to_inps(*raw: int*) → float

Convert raw value [0-255] to in/s

raw_speed_to_mmpps(*raw: int*) → float

Convert raw value [0-255] to mm/s

raw_to_open_in(*raw: int*) → float

Convert raw value [0-255] to in of opening

raw_to_open_mm(*raw: int*) → float

Convert raw value [0-255] to mm of opening

register(*timeout: int = 10*) → None

Register necessary ROS publishers and subscribers.

Parameters **timeout** (*int*, *optional*) – Wait time for topics to start publishing. Defaults to 10.

report_status(*verbose: bool = False*) → None

Output the current state of the gripper.

Parameters **verbose** (*bool*, *optional*) – Display additional raw status message. Defaults to False.

reset() → None

Reset the gripper to default state (may or may not be obstructed).

send_raw_position_command(*position: int, speed: int, force: int, blocking: bool = True*) → None

Send a position request command to the gripper ignoring internal settings.

Parameters

- **position** (*int*) – Raw position value [0-255].
- **speed** (*int*) – Raw speed value [0-255].
- **force** (*int*) – Raw force value [0-255].
- **blocking** (*bool*, *optional*) – Wait until gripper motion is completed. Defaults to True.

set_gripper_force(*value: Union[int, float], unit: str = 'N'*) → None

Set the internal (default) force setting of the gripper.

Parameters

- **value** (*Union[int, float]*) – New force value.
- **unit** (*str*, *optional*) – Unit of measurement. Defaults to 'N'.

set_gripper_speed(*value: Union[int, float], unit: str = 'mm/s'*) → None

Set the internal (default) speed setting of the gripper.

Parameters

- **value** (*Union[int, float]*) – New speed value.
- **unit** (*str*, *optional*) – Unit of measurement. Defaults to 'mm/s'.

```
status_monitor_callback(msg)
```

Callback for status subscriber

```
class rvl_robotiq_controller.RobotiqController.Robotiq3FController
```

Bases: object

```
__init__()
```

rvl_robotiq_modbus_server package

Submodules

rvl_robotiq_modbus_server.RobotiqModbusServer module

```
class rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient(unit_id=9,  
                                                                    input_addr=1000,  
                                                                    output_addr=2000)
```

Bases: object

```
__init__(unit_id=9, input_addr=1000, output_addr=2000)
```

```
connect(device_addr, delay=1)
```

```
disconnect()
```

```
parse_registers(recv_regs, nregs)
```

```
request_status(nbytes=6)
```

```
send_command(command)
```

```
class rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqTCPClient
```

Bases: object

```
__init__()
```

2.3.2 UR Dashboard Control

rvl_ur_remote_dashboard ROS Package

rvl_ur_motion_planner package

Submodules

rvl_ur_motion_planner.URMoveitCommander module

```
class rvl_ur_motion_planner.URMoveitCommander.URCommander(group_name='arm', speed=0.1,  
                                                            accel=0.1)
```

Bases: object

```
__init__(group_name='arm', speed=0.1, accel=0.1)
```

```
all_close(goal, actual, tolerance)
```

Convenience method for testing if the values in two lists are within a tolerance of each other. For Pose and PoseStamped inputs, the angle between the two quaternions is compared (the angle between the identical

orientations q and $-q$ is calculated correctly). @param: goal A list of floats, a Pose or a PoseStamped
@param: actual A list of floats, a Pose or a PoseStamped @param: tolerance A float @returns: bool

```
define_preset_locations()
go_to_preset_location(name)
home()
report()
```

rvl_ur_remote_dashboard package

Submodules

rvl_ur_remote_dashboard.URInterfaceMapping module

```
class rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping(value)
```

Bases: enum.Enum

An enumeration.

```
BACKDRIVE = 6
BOOTING = 2
CONFIRM_SAFETY = 1
DISCONNECTED = 0
IDLE = 5
NO_CONTROLLER = -1
POWER_OFF = 3
POWER_ON = 4
RUNNING = 7
UPDATING_FIRMWARE = 8
```

```
class rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping(value)
```

Bases: enum.Enum

An enumeration.

```
AUTOMATIC_MODE_SAFEGUARD_STOP = 12
FAULT = 9
NORMAL = 1
PROTECTIVE_STOP = 3
RECOVERY = 4
REDUCED = 2
ROBOT_EMERGENCY_STOP = 7
SAFEGUARD_STOP = 5
SYSTEM_EMERGENCY_STOP = 6
SYSTEM_THREE_POSITION_ENABLING_STOP = 13
```

UNDEFINED_SAFETY_MODE = 11

VALIDATE_JOINT_ID = 10

VIOLATION = 8

class rvl_ur_remote_dashboard.URInterfaceMapping.**SetIOFunctionMapping**(*value*)

Bases: enum.Enum

An enumeration.

SET_ANALOG_OUT = 3

SET_DIGITAL_OUT = 1

SET_FLAG = 2

SET_TOOL_VOLTAGE = 4

class rvl_ur_remote_dashboard.URInterfaceMapping.**SetIOPinMapping**(*value*)

Bases: enum.Enum

An enumeration.

class rvl_ur_remote_dashboard.URInterfaceMapping.**SetIOPinState**(*value*)

Bases: enum.Enum

An enumeration.

OFF = 0

ON = 1

class rvl_ur_remote_dashboard.URInterfaceMapping.**SetIOToolState**(*value*)

Bases: enum.Enum

An enumeration.

TOOL_VOLTAGE_0V = 0

TOOL_VOLTAGE_12V = 12

TOOL_VOLTAGE_24V = 24

rvl_ur_remote_dashboard.URRemoteDashboard module

class rvl_ur_remote_dashboard.URRemoteDashboard.**URRemoteDashboard**(*name: str = 'UR5e',*
using_gripper: bool = False,
using_urscript: bool = False,
service_timeout: int = 5)

Bases: object

__init__(*name: str = 'UR5e', using_gripper: bool = False, using_urscript: bool = False, service_timeout: int = 5*) → None

The UR Remote Dashboard class. This is the primary extension overlaying the existing Universal Robot Driver code base to access mapped services.

Parameters

- **name** (*str, optional*) – Readable name to identify controller. Defaults to ‘UR5e’.
- **using_gripper** (*bool, optional*) – Initialized the attached Robotiq gripper. Defaults to False.

- **using_urscript** (*bool*, *optional*) – Register appropriate publisher to send UR Script. Defaults to False.
- **service_timeout** (*int*, *optional*) – Wait time for services to come on. Defaults to 5.

clear_operational_mode() → bool

Allow PolyScope to change operational mode. User password will be enabled.

clear_protective_stop(*timeout: int = 30*) → bool

Clear a protective stop.

close_popup(*safety: bool = False*) → bool

Close a popup on the Teach Pendant or PolyScope.

Parameters **safety** (*bool*, *optional*) – Set to True if popup is a safety popup. Defaults to False.

Returns True if the targeted popup is closed.

Return type bool

cold_boot() → bool

Go directly to operational state (power on, brakes released). See `release_brakes()`.

connect_dashboard(*quiet: bool = False*) → bool

Connect to the dashboard server. Need to be done before calling other services.

define_services()

disconnect_dashboard() → bool

Disconnect from the dashboard server.

get_loaded_program() → Optional[str]

Returns the name of the loaded program.

get_robot_mode() → Optional[int]

Returns the current robot mode.

get_safety_mode() → Optional[int]

Returns the current safety mode.

is_program_running() → bool

Returns true if the default or loaded program is running.

is_program_saved() → bool

Returns true if the default or loaded program is saved.

load_program(*filename: str, ptype: str, wait: int = 10, attempts: int = 10*) → None

Load a program or installation file.

Parameters

- **filename** (*str*) – Name of file with extension e.g., program.urp
- **ptype** (*str*) – Type of program. Accepting ['prog', 'p', 'program', 'urp'] or ['inst', 'i', 'installation'].
- **wait** (*int*, *optional*) – Wait time to handle known disconnection issue. Defaults to 10.
- **attempts** (*int*, *optional*) – Number of reconnection attempts. Defaults to 10.

log_to_pendant(*message: str*) → None

Log a message to PolyScope logs.

pause_loaded_program() → bool

Pause PolyScope program execution.

power_off_arm(*timeout: int = 30*) → bool

Power off the arm.

power_on_arm(*timeout: int = 30*) → bool

Power on the arm to idle state (brakes engaged).

query_program_state() → None

Display the name and execution state of the current PolyScope program.

raw_request(*query*)

Send any arbitrary message or request to the dashboard server.

register_robot_status() → None

Register necessary subscribers and callbacks to monitor robot operational status.

release_brakes(*timeout: int = 30*) → bool

Fully power on the robot with brakes released.

restart_safety() → bool

Clear a safety fault or violation. Arm will be powered off.

robot_iostate_callback(*msg*)

robot_safety_callback(*msg*)

robot_status_callback(*msg*)

send_popup(*message: str*) → bool

Send a message as a popup to Teach Pendant or PolyScope.

set_io(*function: int, pin: int, state: float*) → None

Set specific IO port on the robot. Currently not supporting specific domains (current/voltage).

Parameters

- **function** (*int*) – See SetIOFunctionMapping.
- **pin** (*int*) – Which pin to execute the function on.
- **state** (*float*) – 0/1 for digital IOs and value for analog IO.

set_payload(*mass: float, cx: float, cy: float, cz: float*) → None

Set the payload mass and center of gravity.

Parameters

- **mass** (*float*) – Mass of the payload in kg.
- **cx** (*float*) – Center of gravity of the payload.
- **cy** (*float*) – Center of gravity of the payload.
- **cz** (*float*) – Center of gravity of the payload.

set_speed_slider(*fraction: float*) → None

Set robot execution speed as a fraction. Only set less than 1 on scaled controllers.

Parameters **fraction** (*float*) – 0 to 1 if using scaled (default) controllers.

spam_connect(*attempts: int = 10*) → bool

Repeatedly calling connect() due to error prone and asynchronous status of the server.

Parameters **attempts** (*int, optional*) – Number of times connect() is called internally. Defaults to 10.

start_loaded_program() → bool

Start execution of default or loaded program.

stop_loaded_program() → bool
Stop PolyScope program execution.

system_shutdown() → None
Fully power down the robot (including control box).

terminate_external_control() → bool
Make the external_control node on PolyScope returns.

trigger_service(serv_alias)
Internal trigger service handling with exceptions

verify_services()

zero_force_torque_sensor() → bool
Zero the ft-sensor. Only work on e-Series in remote-control mode.

2.3.3 Utilities

rvl_utilities ROS Package

rvl_utilities package

Submodules

rvl_utilities.CustomLogger module

```
class rvl_utilities.CustomLogger.ColorLogger(label=None)
    Bases: object
    __init__(label=None)
    define_sequences()
    gen_output_str(msg, esc, indent=0)
    log_error(msg, indent=0)
    log_info(msg, indent=0)
    log_success(msg, indent=0)
    log_warn(msg, indent=0)
rvl_utilities.CustomLogger.gen_ansi_rgb_esc(code)
rvl_utilities.CustomLogger.gen_esc(code)
rvl_utilities.CustomLogger.hex_to_rgb(hex, ansi=True)
rvl_utilities.CustomLogger.rgb_to_hex(rgb)
```


REFERENCES

The bulk of this driver is written based on old or current work, with some additional tweaks and tricks collected from browsing Universal Robots and Robotiq resources. Below is a tentative list of references, which will grow as we collect more.

- [UniversalRobots/Universal_Robots_ROS_Driver](#)
- [ros-industrial/robotiq](#)
- [IntelRealSense/realsense-ros](#)
- [MoveIt! Motion Planning Framework](#)
- [Blog post by Jean-Philippe Roberge](#) (last maintainer of the ROS Industrial Robotiq repository)
- [ROS Tutorials](#)

LICENSE

This repository is distributed under the **Apache License, Version 2.0**. Be sure to read (or briefly tl;dr scan) what this means.

A nice tl;dr is provided [here](#).

Warning: As always when working with any robot, safety is only as safe as the human can make it. So please, **proceed with caution**.

4.1 Apache 2.0 Boilerplate

Copyright 2022 Minh Tram

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

CHANGELOG

5.1 Releases

5.1.1 0.0.1-alpha - 2022/FEB/01

- Initial release of driver components: UR Remote Dashboard and Robotiq Controller
- Support for rudimentary forward and inverse kinematics
- Added pre-built Docker Image

5.2 Tasks

Don't worry if there's nothing here. I'm sure there's more...

PYTHON MODULE INDEX

r

- `rvl_robotiq_controller.Robotiq2FSupport`, [8](#)
- `rvl_robotiq_controller.Robotiq3FSupport`, [8](#)
- `rvl_robotiq_controller.RobotiqController`, [8](#)
- `rvl_robotiq_modbus_server.RobotiqModbusServer`, [12](#)
- `rvl_ur_motion_planner.URMoveitCommander`, [12](#)
- `rvl_ur_remote_dashboard.URInterfaceMapping`, [13](#)
- `rvl_ur_remote_dashboard.URRemoteDashboard`, [14](#)
- `rvl_utilities.CustomLogger`, [17](#)

Symbols

[__init__\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 8](#)
[__init__\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq3FController method\), 12](#)
[__init__\(\) \(rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient method\), 12](#)
[__init__\(\) \(rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqTCPClient method\), 12](#)
[__init__\(\) \(rvl_ur_motion_planner.URMoveitCommander.URCommander method\), 12](#)
[__init__\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 14](#)
[__init__\(\) \(rvl_utilities.CustomLogger.ColorLogger method\), 17](#)

A

[activate\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 8](#)
[all_close\(\) \(rvl_ur_motion_planner.URMoveitCommander.URCommander method\), 12](#)
[auto_close\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[auto_open\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[AUTOMATIC_MODE_SAFEGUARD_STOP \(rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute\), 13](#)

B

[BACKDRIVE \(rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute\), 13](#)
[block\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[BOOTING \(rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute\), 13](#)

C

[calibrate\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[clear_operational_mode\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)
[clear_protective_stop\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)
[close_popup\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)
[cold_boot\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)
[ColorLogger \(class in rvl_utilities.CustomLogger\), 17](#)
[compensated_publish\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[CONFIRM_SAFETY \(rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute\), 13](#)
[connect\(\) \(rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient method\), 12](#)
[connect_dashboard\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)

D

[deactivate\(\) \(rvl_robotiq_controller.RobotiqController.Robotiq2FController method\), 9](#)
[define_preset_locations\(\) \(rvl_ur_motion_planner.URMoveitCommander.URCommander method\), 13](#)
[define_sequences\(\) \(rvl_utilities.CustomLogger.ColorLogger method\), 17](#)
[define_services\(\) \(rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method\), 15](#)

disconnect() (*rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient method*), 12
disconnect_dashboard() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
DISCONNECTED (*rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute*), 13

F

FAULT (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), 13

G

gen_ansi_rgb_esc() (*in module rvl_utilities.CustomLogger*), 17
gen_esc() (*in module rvl_utilities.CustomLogger*), 17
gen_output_str() (*rvl_utilities.CustomLogger.ColorLogger method*), 17
generate_2f_status_from_binary() (*in module rvl_robotiq_controller.Robotiq2FSupport*), 8
generate_binary_command_from_2f_msg() (*in module rvl_robotiq_controller.Robotiq2FSupport*), 8
get_loaded_program() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
get_robot_mode() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
get_safety_mode() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
go_to_preset_location() (*rvl_ur_motion_planner.URMoveitCommander.URCommander method*), 13
grasp_hard() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 9
grasp_medium() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 9
grasp_soft() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
grasp_soft_regrasp() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10

H

hex_to_rgb() (*in module rvl_utilities.CustomLogger*), 17
home() (*rvl_ur_motion_planner.URMoveitCommander.URCommander method*), 13

I

IDLE (*rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute*), 13
inps_to_raw_speed() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
is_holding() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
is_moving() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
is_program_running() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
is_program_saved() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15

L

lbf_to_raw_force() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
load_program() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
log_error() (*rvl_utilities.CustomLogger.ColorLogger method*), 17
log_info() (*rvl_utilities.CustomLogger.ColorLogger method*), 17
log_success() (*rvl_utilities.CustomLogger.ColorLogger method*), 17
log_to_pendant() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 15
log_warn() (*rvl_utilities.CustomLogger.ColorLogger method*), 17

M

mmmps_to_raw_speed() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 10
module
 rvl_robotiq_controller.Robotiq2FSupport, 8
 rvl_robotiq_controller.Robotiq3FSupport, 8
 rvl_robotiq_controller.RobotiqController, 8
 rvl_robotiq_modbus_server.RobotiqModbusServer, 12
 rvl_ur_motion_planner.URMoveitCommander, 12
 rvl_ur_remote_dashboard.URInterfaceMapping, 13
 rvl_ur_remote_dashboard.URRemoteDashboard, 14

`rvl_utilities.CustomLogger`, 17

N

`newton_to_raw_force()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 10
`NO_CONTROLLER` (`rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping` attribute), 13
`NORMAL` (`rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping` attribute), 13

O

`OFF` (`rvl_ur_remote_dashboard.URInterfaceMapping.SetIOPinState` attribute), 14
`ON` (`rvl_ur_remote_dashboard.URInterfaceMapping.SetIOPinState` attribute), 14
`open_gripper()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 10
`open_in_to_raw()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 10
`open_mm_to_raw()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 10

P

`parse_registers()` (`rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient` method), 12
`pause_loaded_program()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 15
`power_monitor_callback()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 10
`POWER_OFF` (`rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping` attribute), 13
`power_off_arm()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 15
`POWER_ON` (`rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping` attribute), 13
`power_on_arm()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`PROTECTIVE_STOP` (`rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping` attribute), 13

Q

`query_program_state()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16

R

`raw_force_to_lbf()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_force_to_newton()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_request()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`raw_speed_to_inps()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_speed_to_mmmps()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_to_open_in()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_to_open_mm()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`raw_to_rad_2f140()` (in module `rvl_robotiq_controller.Robotiq2FSupport`), 8
`raw_to_rad_2f85()` (in module `rvl_robotiq_controller.Robotiq2FSupport`), 8
`RECOVERY` (`rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping` attribute), 13
`REDUCED` (`rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping` attribute), 13
`register()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`register_robot_status()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`release_brakes()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`report()` (`rvl_ur_motion_planner.URMoveitCommander.URCommander` method), 13
`report_status()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`request_status()` (`rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient` method), 12
`reset()` (`rvl_robotiq_controller.RobotiqController.Robotiq2FController` method), 11
`restart_safety()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`rgb_to_hex()` (in module `rvl_utilities.CustomLogger`), 17
`ROBOT_EMERGENCY_STOP` (`rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping` attribute), 13
`robot_iostate_callback()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`robot_safety_callback()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`robot_status_callback()` (`rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard` method), 16
`Robotiq2FController` (class in `rvl_robotiq_controller.RobotiqController`), 8

Robotiq3FController (*class in rvl_robotiq_controller.RobotiqController*), 12
 RobotiqRTUClient (*class in rvl_robotiq_modbus_server.RobotiqModbusServer*), 12
 RobotiqTCPClient (*class in rvl_robotiq_modbus_server.RobotiqModbusServer*), 12
 RobotModeMapping (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 13
 RUNNING (*rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute*), 13
 rvl_robotiq_controller.Robotiq2FSupport
 module, 8
 rvl_robotiq_controller.Robotiq3FSupport
 module, 8
 rvl_robotiq_controller.RobotiqController
 module, 8
 rvl_robotiq_modbus_server.RobotiqModbusServer
 module, 12
 rvl_ur_motion_planner.URMoveitCommander
 module, 12
 rvl_ur_remote_dashboard.URInterfaceMapping
 module, 13
 rvl_ur_remote_dashboard.URRemoteDashboard
 module, 14
 rvl_utilities.CustomLogger
 module, 17

S

SAFEGUARD_STOP (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), 13
 SafetyModeMapping (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 13
 send_command() (*rvl_robotiq_modbus_server.RobotiqModbusServer.RobotiqRTUClient method*), 12
 send_popup() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 send_raw_position_command() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 11
 SET_ANALOG_OUT (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOFunctionMapping attribute*), 14
 SET_DIGITAL_OUT (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOFunctionMapping attribute*), 14
 SET_FLAG (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOFunctionMapping attribute*), 14
 set_gripper_force() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 11
 set_gripper_speed() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 11
 set_io() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 set_payload() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 set_speed_slider() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 SET_TOOL_VOLTAGE (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOFunctionMapping attribute*), 14
 SetIOFunctionMapping (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 14
 SetIOPinMapping (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 14
 SetIOPinState (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 14
 SetIOToolState (*class in rvl_ur_remote_dashboard.URInterfaceMapping*), 14
 spam_connect() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 start_loaded_program() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 status_monitor_callback() (*rvl_robotiq_controller.RobotiqController.Robotiq2FController method*), 11
 stop_loaded_program() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 16
 SYSTEM_EMERGENCY_STOP (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), 13
 system_shutdown() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 17
 SYSTEM_THREE_POSITION_ENABLING_STOP (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), 13

T

terminate_external_control() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), 17
 TOOL_VOLTAGE_0V (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOToolState attribute*), 14

TOOL_VOLTAGE_12V (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOToolState attribute*), [14](#)
TOOL_VOLTAGE_24V (*rvl_ur_remote_dashboard.URInterfaceMapping.SetIOToolState attribute*), [14](#)
trigger_service() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), [17](#)

U

UNDEFINED_SAFETY_MODE (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), [14](#)
UPDATING_FIRMWARE (*rvl_ur_remote_dashboard.URInterfaceMapping.RobotModeMapping attribute*), [13](#)
URCommander (*class in rvl_ur_motion_planner.URMoveitCommander*), [12](#)
URRemoteDashboard (*class in rvl_ur_remote_dashboard.URRemoteDashboard*), [14](#)

V

VALIDATE_JOINT_ID (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), [14](#)
verify_services() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), [17](#)
VIOLATION (*rvl_ur_remote_dashboard.URInterfaceMapping.SafetyModeMapping attribute*), [14](#)

Z

zero_force_torque_sensor() (*rvl_ur_remote_dashboard.URRemoteDashboard.URRemoteDashboard method*), [17](#)