Robotnik REST Implementation for Brain-Iot

Robotnik Automation, S.L.L.

Contents

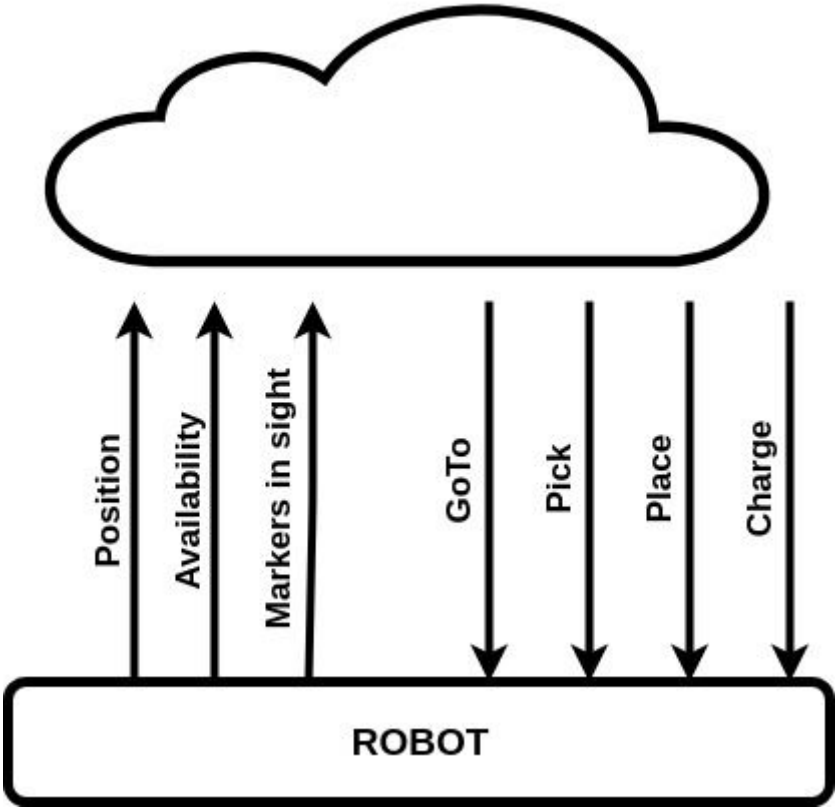
1. Introduction

The following document defines the REST protocol to interface with the Robotnik available services.This system is intended to control robots in order to perform several types of tasks related with autonomous transport of materials.

We recommend to use postman as a GUI.

2. Implementation

For the Brain-IoT application the structure will be the following:



The arrows in the cloud direction will be information available to **GET** from the robot and the arrows in the robot direction actions to **POST** to the robot.
Each action posted has also the option to cancel it and get the information on the current development. These will be explained later.

3. Communication protocol

All the communication with the robots is made by using a specific REST protocol. The format to access any service is the following:

http://**host**:**port**/***robot_model***/**ws_name**

> ***host****: hostname or ip address of the robot*
> ***port****: port used for the communication (default 8080)*
> ***robot_model:*** *model of the robot (rb1_base, rb2, etc.)*
> ***ws_name****: name of the web service*

3.1 Information Commands

These commands are used to obtain information from the robot situation:

- Position
- Availability
- Markers in sight

3.1.1 Get Position

Type: GET
Description: Gets Position from the robot with respect to the map
Command url:
host:port/rb1_base_a/robot_local_control/LocalizationComponent/status

The result obtained when sending the command will look like this:


```
{
   "node": "-1",
   "reliable": true,
   "pose": {
      "header": {
         "stamp": {
            "secs": 3373,
            "nsecs": 424000000
         },
         "frame_id": "rb1_base_a_map",
         "seq": 0
      },
      "pose": {
         "y": -0.011933975869164705,
         "x": 0.005695701796862853,
```

```
            "theta": -0.00103792269280593
        }
    },
    "type": "",
    "state": "READY"
}
```


Where the pose parameters x, y, and theta are the ones that define the position.

3.1.2 Availability

Type: GET
Description: Gets the robot availability to perform actions
Command url:
host:port/rb1_base_a/robot_local_control/state

The result obtained when sending the command will look like this


```
{
    "operation_state": "idle",
    "robot_state": "standby",
    "navigation_status": {
        "state": "",
        "type": "PickComponent,"
    },
    "localization_status": {
        "node": "-1",
        "reliable": true,
        "pose": {
            "header": {
                "stamp": {
                    "secs": 3398,
                    "nsecs": 334000000
                },
                "frame_id": "rb1_base_a_map",
                "seq": 0
            },
            "pose": {
                "y": -3.6808252039538503,
                "x": 0.23644128868579684,
                "theta": -3.117151068232479
            }
        },
        "type": "",
        "state": "READY"
    },
```

```
"robot_status": {
    "emergency_stop": false,
    "battery": {
        "time_remaining": 0,
        "time_charging": 0,
        "is_charging": false,
        "voltage": 0,
        "level": 0
    },
    "pose": {
        "header": {
            "stamp": {
                "secs": 3398,
                "nsecs": 334000000
            },
            "frame_id": "rb1_base_a_odom",
            "seq": 1373
        },
        "pose": {
            "y": -3.719545596838467,
            "x": 0.18625711057202374,
            "theta": -3.121946806174189
        }
    },
    "elevator": {
        "position": "unknown",
        "state": "idle"
    },
    "velocity": {
        "angular_z": -0.000014735845100020766,
        "linear_x": -0.00001552743234123355,
        "linear_y": 0
    },
    "sensors": [
        {
            "state": "READY",
            "type": "robot_local_control_components/LaserComponent",
            "name": "LaserComponent"
        }
    ]
},
"robot_id": "",
"mission_status": {
    "current": {
        "state": "",
        "id": 0,
        "description": ""
    },
```

```
      "last": []
   },
   "control_state": "auto"
}
```

Where:

- **operation_state**: Defines what the robot is doing
  - idle
  - calibrating
  - moving
  - raising_elevator
  - lowering_elevator
  - charging
- **robot_state**: Defines the robots global state
  - init: the robot is initializing
  - standby: the robot is ready to receive orders. Power saving behaviours can be applied
  - ready: the robot is ready to work or working
  - emergency: the robot cannot work/operate correctly temporary. It can imply a recovery procedure or an external action from the operator
  - failure: the robot is not working. It needs the operator intervention and probably restart/reboot the robot
- **navigation_status**: Defines the robot navigation status
  - type: Identifies the navigation procedure being used
  - state: Defines the navigation state
    - init
    - standby
    - ready
    - emergency
    - failure
- **localization_status**: Defines the robot localization status
  - node: closest node/tag (used with the Fleet Management System)
  - reliable: flag that indicates that the localization estimation is reliable
  - pose: current pose related to a global frame used by the localization component
  - type: identifies the localization procedure being used
  - state: defines the localization state
    - init
    - standby
    - ready
    - emergency
    - failure
    - unknown
- **robot_status**: Defines the current robot status in terms of internal hardware and odometry
  - emergency_stop: flag that indicates that the emergency_stop is active
  - battery:
    - time_remaining: in seconds

        - time_charging: in seconds
        - is_charging: flag to indicate if the robot is charging
        - voltage: battery voltage in volts
        - level: battery percentage
    - pose: current pose based on the robot's odometry
    - elevator:
        - position: "unknown"
        - state: "idle"
    - velocity: current robot velocity
    - sensors: list of robot sensors and their state
        - state:
            - init
            - standby
            - ready
            - emergency
            - failure
            - Unknown
        - type
        - name
- **robot_id**: string to identify the robot
- **mission_status**: state of the current mission or task
- **control_state**:
    - auto: the robot is moving autonomously
    - manual: the robot is being tele-operated by an operator
    - follow: the robot is following a person
    - unknown

3.1.3 Markers in sight

Type: GET
Description: Gets the information about the markers the robot is currently seeing.
Command url:
host:port/rb1_base_a/ar_pose_marker

The result obtained when sending the command will look like this


```
{
   "header": {
      "stamp": {
         "secs": 0,
         "nsecs": 0
      },
      "frame_id": "",
      "seq": 4324
   },
   "markers": [
```

```
{
    "header": {
        "stamp": {
            "secs": 3911,
            "nsecs": 658000000
        },
        "frame_id": "rb1_base_a_front_rgbd_camera_rgb_optical_frame",
        "seq": 0
    },
    "confidence": 0,
    "pose": {
        "header": {
            "stamp": {
                "secs": 0,
                "nsecs": 0
            },
            "frame_id": "",
            "seq": 0
        },
        "pose": {
            "position": {
                "y": -0.11792459036294335,
                "x": -1.099986603912028,
                "z": 2.7405836974247646
            },
            "orientation": {
                "y": 0.99827329586809,
                "x": -0.0020421436667357722,
                "z": 0.03078211091399566,
                "w": -0.0499871788920295
            }
        }
    },
    "id": 4
},
{
    "header": {
        "stamp": {
            "secs": 3911,
            "nsecs": 658000000
        },
        "frame_id": "rb1_base_a_front_rgbd_camera_rgb_optical_frame",
        "seq": 0
    },
    "confidence": 0,
    "pose": {
        "header": {
            "stamp": {
```

```
                "secs": 0,
                "nsecs": 0
            },
            "frame_id": "",
            "seq": 0
        },
        "pose": {
            "position": {
                "y": -0.1148982368617346,
                "x": 1.0202332485314194,
                "z": 2.5468427160904357
            },
            "orientation": {
                "y": 0.9984770068400043,
                "x": -0.0029178332670966403,
                "z": -0.008389671449998687,
                "w": -0.05444966918001045
            }
        }
    },
    "id": 3
  }
 ]
}
```


Where the id's of the markers are displayed (in this case the robot sees marker 3 and 4) and also their relative position with respect to the robot camera.

3.2 Action Commands

This actions are exclusive, non-preemptive and non-queueable, that means that the robot will only execute one action at a time, and will discard all subsequent actions until current action is finished. Actions can be canceled.
For each command it is possible to cancel it or check its state.

3.2.1 Goto

Component that performs the actions to navigate to a pose (position plus orientation) in a map.

3.2.1.1 Add

Type: POST

Description: Commands a robot to go to a goal. The mission will be finished when the robot arrive to the goal.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/GoToComponent/add


```
{
"procedure":
   {
   "goals":
         [
                  {
                  "header":
                           {
                           "seq": 0,
                           "stamp":
                                    {
                                    "secs": 0,
                           "nsecs": 0
                                    },
                           "frame_id": "rb1_base_a_map"
                           },
                  "pose":
                           {
                           "x":  0.413566538606,
                           "y":  -3.69224357086,
                           "theta": 3.14
                           }
                  }
         ],
   "max_velocities":
         [
                  {
                  "linear_x": 0.0,
                  "linear_y": 0.0,
                  "angular_z": 0.0
                  }
         ]
         }
}
```


Where:

- **goals**: List of goals.
- **frame_id**: Frame of reference for the pose.
- **pose**: Position and orientation of the goal.
- **max_velocities**: List of max velocities allowed.

Returns:


```
{
	"_format": "ros",
	"state": {
	"header": {
	"priority": 0,
	"stamp": {
		"secs": 0,
		"nsecs": 0
	},
	"id": 0,
	"name": ""
	},
	"current_state": "",
	"last_event": ""
	},
	"result": {
	"message": "Cannot add procedure because goal frame is empty",
	"result": "error"
	}
}
```


Where:

- **priority:** priority of the current procedure. used in case procedure preemption is allowed.
- **id**: identifies the procedure in the robot context.
- **name**: human readable identificator.
- **current_state**: allowed values are: queued, running, paused, finished and unknown
- **last_event**: allowed values are: added, start, stop, cancel, pause, resume, finish and abort.
- **message:** is the returned message description
- **result**: allowed values are: ok and error

3.2.1.2 Cancel

Type: POST
Description: Cancels the current goto mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/GoToComponent/cancel


```
{
```

```
"header":
   {
   "id": -1,
   "priority": 0,
   "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
   "name": ""
   }
}
```

Where:

- **id**: The id of the goto mission you want to cancel. -1 cancels last mission.

Returns:

Same type of return message that GoToComponent/Add

3.2.1.3 Query State

Type: POST
Description: Gets the state of a goto mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/GoToComponent/query_state

```
{
"header":
  {
  "id": -1,
  "priority": 0,
  "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
  "name": ""
  }
}
```

Where:

- **id**: The id of the goto mission you want to get the query state. -1 gets the query state of the last mission.

Returns:

Same type of return message that GoToComponent/Add

3.2.2 Pick Component

Component that performs the picking procedure of a cart. Only available when the robot is in front of a compatible cart.

3.2.2.1 Add

Type: POST
Description: Commands a robot to start picking procedure. The mission will be finished when the robot raise the cart.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/PickComponent/add

```
{
"procedure":
   {
   "pick_frame_id": "rb1_base_a_cart2_contact"
   }
}
```

Where:

- **pick_frame_id**: It can be different frames depending on the cart to pick (rb1_base_a_cart2_contact, rb1_base_a_cart3_contact or rb1_base_a_cart4_contact)

Returns:

Same type of return message that GoToComponent/Add

3.2.2.2 Cancel

Type: POST
Description: Cancels the current pick mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/PickComponent/cancel

```
{
"header":
```

```
  {
  "id": -1,
  "priority": 0,
  "stamp":
        {
        "secs": 0,
        "nsecs": 0
        },
  "name": ""
  }
}
```

Where:

- **id**: The id of the pick mission you want to cancel. -1 cancels last mission.

Note: This procedure currently is not available

3.2.2.3 Query State

Type: POST
Description: Gets the state of a pick mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/PickComponent/query_state

```
{
"header":
 {
 "id": -1,
 "priority": 0,
 "stamp":
        {
        "secs": 0,
        "nsecs": 0
        },
 "name": ""
 }
}
```

Where:

- **id**: The id of the pick mission you want to get the query state. -1 gets the query state of the last mission.

Returns:

Same type of return message that GoToComponent/Add

3.2.3 Place Component

Component to unload a cart from the robot. Includes the action to lower the elevator, so the cart is detached from the robot, and then moves away from the cart.

3.2.3.1 Add

Type: POST
Description: Commands a robot to start place procedure. The mission will be finished when the robot raise the cart.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/PlaceComponent/add

```
{
"procedure":
   {
   "pick_frame_id": ""
   }
}
```

Where:

- **pick_frame_id**: This is currently not used.

Returns:

Same type of return message that GoToComponent/Add

3.2.3.2 Cancel

Type: POST
Description: Cancels the current place mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/PlaceComponent/cancel

```
{
"header":
   {
   "id": -1,
   "priority": 0,
```

```
   "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
   "name": ""
   }
}
```

Where:

- **id**: The id of the place mission you want to cancel. -1 cancels current mission.

3.2.3.3 Query State

Type: POST
Description: Gets the state of a place mission.
Command url:
host:port/robot_model/robot_local_control/NavigationComponent/PlaceComponent/query_state

```
{
"header":
  {
  "id": -1,
  "priority": 0,
  "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
  "name": ""
  }
}
```

Where:

- **id**: The id of the place mission you want to get the query state. -1 gets the query state of the last mission.

Returns:

Same type of return message that GoToComponent/Add

3.2.4 Charge Battery Component

Component to dock into the Battery Docking Station and start the battery charge.

3.2.4.1 Add

Type: POST
Description: Commands a robot to dock and start battery charge.

Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/ChargeComponent/add

```
{
   "procedure":
   {
     “pick_frame_id": “docking_station_frame”
  }
}
```

Where:

- **docking_station_frame**: frame used to perform the docking. Note that this frame needs to be being published/accessible by the system.

Returns:

Same type of return message that GoToComponent/Add

3.2.4.2 Cancel

Type: POST
Description: Cancels the current charge mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/ChargeComponent/cancel

```
{
"header":
   {
   "id": -1,
   "priority": 0,
   "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
```

```
   "name": ""
   }
}
```

Where:

- **id**: The id of the mission you want to cancel. -1 cancels current mission.

3.2.4.3 Query State

Type: POST
Description: Gets the state of a charge mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/ChargeComponent/query_state

```
{
"header":
  {
  "id": -1,
  "priority": 0,
  "stamp":
          {
          "secs": 0,
          "nsecs": 0
          },
  "name": ""
  }
}
```

Where:

- **id**: The id of the mission you want to get the query state. -1 gets the query state of the last mission.

Returns:

Same type of return message that GoToComponent/Add

3.2.5 Uncharge Battery Component

Component to stop charging and perform the undocking actions from the docking station.

3.2.5.1 Add

Type: POST
Description: Commands a robot to stop charging and leave the docking station.

Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/UnChargeComponent/add

```
{
   "procedure":
   {

  }
}
```

Returns:

Same type of return message that GoToComponent/Add

3.2.5.2 Cancel

Type: POST
Description: Cancels the current uncharge mission.
Command url:
host:port/rb1_base_a/robot_local_control/NavigationComponent/UnChargeComponent/cancel

```
{
"header":
   {
   "id": -1,
   "priority": 0,
   "stamp":
         {
         "secs": 0,
         "nsecs": 0
         },
   "name": ""
```

```
  }
}
```

Where:

- **id**: The id of the mission you want to cancel. -1 cancels current mission.

3.2.5.3 Query State

Type: POST
Description: Gets the state of the mission.
Command url:
host:port/robot_model/robot_local_control/NavigationComponent/ChargeComponent/query_state

```
{
"header":
 {
 "id": -1,
 "priority": 0,
 "stamp":
        {
        "secs": 0,
        "nsecs": 0
        },
 "name": ""
 }
}
```

Where:

- **id**: The id of the mission you want to get the query state. Value -1 gets the query state of the last mission.

Returns:

Same type of return message that GoToComponent/Add