Funded by the European Union

Robot Competitions Kick Innovation
in Cognitive Systems and Robotics
FP7-ICT-601012

RoCKIn@Work

– Innovation in Industrial Mobile Manipulation –

II: RoCKIn@Work Rule Book

Revision: 2.0
Date: November 11, 2015

Editors: Jakob Berghofer, Rhama Dwiputra, Tim Friedrich, Gerhard Kraetzschmar
Aamir Ahmad, Francesco Amigoni, Iman Awaad,
Contributors: Jakob Berghofer, Rainer Bischoff, Andrea Bonarini,
Rhama Dwiputra, Giulio Fontana, Frederik Hegger,
Nico Hochgeschwender, Luca Iocchi, Gerhard Kraetzschmar,
Pedro Lima, Matteo Matteucci, Daniele Nardi,
Viola Schiaffonati, Sven Schneider

Contents

1 Introduction to RoCKIn@Work

RoCKIn@Work is a competition that aims at bringing together the benefits of scientific benchmarking with the economic potential of innovative robot applications for industry, which call for robots capable of working interactively with humans and requiring reduced initial programming. The following ***user story*** is the basis upon which the RoCKIn@Work competition is built:

RoCKIn@Work is set in the RoCKIn'N'RoLLIn factory - a medium-sized factory that is trying to optimize its production process to meet the increasing number of unique demands from its customers. RoCKIn'N'RoLLIn specializes in the production of small to medium sized lots of mechanical parts and assembled mechatronic products. Furthermore, the RoCKIn'N'-RoLLIn production line integrates incoming shipments of damaged or unwanted products and raw materials.

Greater automation in broader application domains than today is essential for ensuring European industry remains competitive, production processes are flexible to custom demands and factories can operate safely in harsh or dangerous environments. In RoCKIn@Work, robots will assist with the assembly of a drive axle - a key component of the robot itself and therefore a step towards self-replicating robots. Tasks include locating, transporting and assembling necessary parts, checking their quality and preparing them for other machines and workers. By combining the versatility of human workers and the accuracy, reliability and robustness of mobile robot assistants, the entire production process is able to be optimized.

RoCKIn@Work is looking to make these innovative and flexible manufacturing systems, such as that required by the RoCKIn'N'RoLLIn factory, a reality. This is the inspiration behind the Challenge and the following scenario description.

A more detailed account of RoCKIn@Work, but still targeted towards a general audience, is given in the RoCKIn@Work in a Nutshell document (see [1]), which gives a brief introduction to the very idea of RoCKIn and RoCKIn@Work, the underlying user story, and surveys the scenario, including the environment for user story, the tasks to be performed, and the robots targeted. Furthermore, this document gives general descriptions of the task benchmarks and the functional benchmarks that make up RoCKIn@Work.

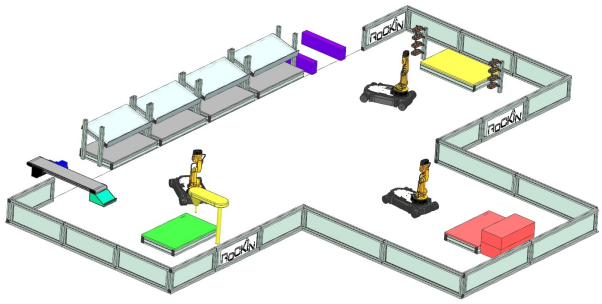
The document on hand is the rule book for RoCKIn@Work, and it is assumed that the reader has already read the nutshell document. The audience for the current document are teams who want to participate in the competition, the organizers of events where the RoCKIn-@Work competition is supposed to be executed, and the developers of simulation software, who want to provide their customers and users with ready-to-use models of the environment. They all need to know more details on the competition than the nutshell document provides.

The remainder of this document is structured as follows: The ***test bed*** for RoCKIn@Work competitions is described in some detail in the next section (Section 2). Subsections are devoted to the specification of the structure of the environment and its properties (2.1), to the mechanical parts and objects in the environment which can be manipulated (2.2.1, 2.2.2), to objects in the environment that need to be recognized for completing the task (2.2.3), to the networked devices embedded in the environment and accessible to the robot (2.4), and to the benchmarking equipment which we plan to install in the environment and which may impose additional constraints to the robot's behavior (equipment presenting obstacles to avoid) or add further perceptual noise (visible equipment) (2.6). Next (Section 3), we provide some specifications and constraints applying to the ***robots and teams*** permitted to participate in RoCKIn@Work. The RoCKIn consortium is striving to minimize such constraints, but for reasons of safety and practicality such constraints are required. After that, the next two sections describe in detail the ***task benchmarks*** (Section 4) and the ***functionality benchmarks*** (Section 5) comprising the RoCKIn@Work competition. While information on scoring and ranking the performance of participating teams on each benchmark is already provided in the benchmark descriptions, Section 6, ***award categories*** surveys the number and kind of awards that will be awarded and how the ranking of the award categories is determined based on individual benchmark results.

Last but not least, Section 7 provides details on ***organizational issues***, like the committees involved, the media to communicate with teams, qualification and setup procedures, competition schedules, and post-competition activities.

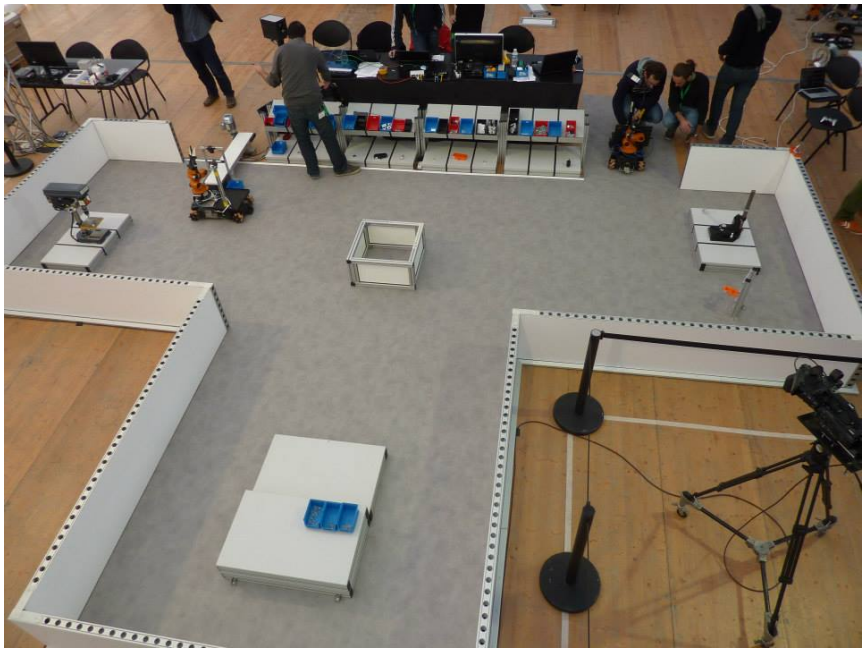
2 The RoCKIn@Work Test Bed

The test bed for RoCKIn@Work consists of the environment in which the competition will happen, including all the objects and artifacts in the environment, and the equipment brought into the environment for benchmarking purposes. An aspect that is comparatively new in robot competitions is that RoCKIn@Work is, to the best of our knowledge, the first industry-oriented robot competition targeting an *environment with ambient intelligence*, i.e. the environment is equipped with networked electronic devices the robot can communicate and interact with, and which allow the robot to exert control on certain environment artifacts like conveyor belts or machines. Figure 1 shows the evolution of the RoCKIn@Work environment from its early concept to its implementation in the RoCKIn@Work event. Participating teams should assume the competition environment to be similar to those shown in Figure 1; deviations should only occur if on-site constraints (space available, safety regulations) enforce them.



(a) Early concept

(b) Laboratory installation



(c) RoCKIn@Work 2014

Figure 1: The evolution of the RoCKIn@Work environment

2.1 Environment Structure and Properties

The following set of scenario specifications must be met by the RoCKIn@Work environment.

Environment Specification 2.1 (*Structured Environment*)

The environment consists of five spatial areas:

1. *a row of shelves*
2. *force fitting workstation*
3. *drilling workstation*
4. *a conveyor belt*
5. *assembly workstation*

Figure 2 shows an example of the these areas in the RoCKIn@Work environment. The spatial areas extend beyond the space occupied by the respective workstations or objects and include the surrounding area as well.

Environment Specification 2.2 (*Flat Environment*)

All spatial areas are located on the same level, except where specified otherwise. There are no stairs in the environment.

Environment Specification 2.3 (*Spatial Areas and Rooms*)

The factory is a single, large open space; there are no rooms separated by walls in the environment. Spatial areas can be partially separated by dividing or protective walls or other objects present in the factory (e.g. shelves, workstations, platforms, tables).

Figure 2: Example of the spatial areas in RoCKIn@Work environment. The spatial areas are shelves (red), force fitting workstation (blue), conveyor belt (green), drilling workstation (orange) and assembly workstation (yellow).

Environment Specification 2.4 (*Dimensions*)

The precise dimensions and the arrangement of the spatial areas are not predefined, but estimated sizes are given. The estimated sizes of the spatial areas are as follows: workstations (assembly, drilling and force fitting) $2m \times 2m$*, conveyor belt* $1,5m \times 0,5m$ *and shelves* $5m \times 0,5m$*. The bounding box of the environment has a minimum area of* $16m^2$ *and a maximum area of* $100m^2$*. More space is used, when areas and workstations are doubled for teams working parallel.*

Environment Specification 2.5 (*Set of Shelves*)

The shelves-area is a set of connected shelves and each shelves has two level (upper level and lower level). The robot can take and/or deliver objects from the shelves (through the containers or directly onto shelves). Figure 2 shows an example of the shelves-area made from two set of shelves.

Environment Specification 2.6 (*Force Fitting Workstation*)

Force fitting workstation has a table for temporarily storing handled parts. The table is part of the force fitting machine which is operated by a robot or human worker. On one side is the assembly aid tray rack to attach filled or unfilled aid trays.

Environment Specification 2.7 (*Drilling Workstation*)

Drilling workstation consists of a storing area to store "file card" boxes and the drilling machine.

Environment Specification 2.8 (*Conveyor Belt*)

The conveyor belt transports parts from outside of the arena into the area. At the end of the conveyor belt, parts fall down on an exit ramp in a predefined position through guiders where they can be taken by the robot.

Environment Specification 2.9 (*Assembly workstation*)

Assembly workstation consists of a table, where a human worker performs assembly of parts. The table features predefined areas where the robot can put boxes with supplies and pick up boxes with finished parts, that have already been processed by the worker and need to be delivered elsewhere.

2.2 Objects in the Environment

The following lists describe the objects found in the environment. Three classes of objects are defined:

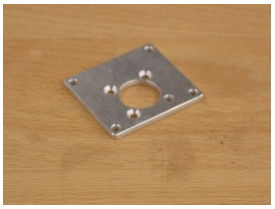
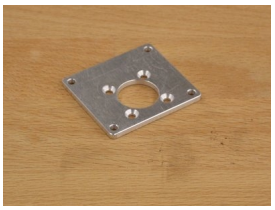
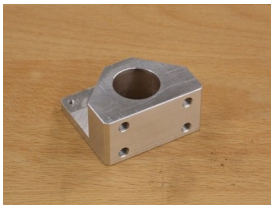
- mechanical parts that have to be recognized and manipulated.
- objects in the environment that have to be recognized and manipulated.
- objects in the environment that have only to be recognized (because they are fixed to the environment, to heavy to lift and it is not necessary).

These objects are listed in tables 1, 2 and 3.

2.2.1 Parts to Manipulate

Table 1: List of parts for assembling the drive axle

Part ID	Part name	Picture
AX-01	Bearing box type A	
AX-02	Bearing	
AX-03	Axis	

AX-04	Shaft nut	
AX-05	Distance tube	
AX-06	Faulty cover plate	
AX-07	Perfect cover plate	
AX-08	Unusable cover plate	
AX-09	Assembled motor with bearing box type A	
AX-16	Bearing box type B	

2.2.2 Objects in the Environment to Manipulate

The abbreviation "EM" stands for *environment manipulate*.

Table 2: List of manipulated objects in the environment

Object ID	Object name	Picture
EM-01-XX	Aid tray (identifier code can be placed on any side)	
EM-02-XX	File card box	

2.2.3 Objects to Recognize

In the following list only "smaller" objects are described. For a detailed description of the environment's furniture see Section 2.1. The abbreviation "ER" stands for *environment recognize*.

Table 3: List of objects in the environment

Object ID	Object name	Picture
ER-01-XX	Assembly aid tray rack	
ER-02-X	Container box	

2.3 Identifier

The identifier consists of the information about the object class, the object type and the unique identifier. Each information in the identifier is separated by a single dash. For example EM-01-01 means that the object is of class EM (objects in the environment to manipulate), it is of type 01 (assembly aid tray) and it is the object "01". The identifier is used to monitor the production

process and as a reference number to track a customer specific assemblies. RoCKIn@Work uses ArUco marker (size of approximately $2.5cm \times 2.5cm$) as an identifier for areas and objects in its environment. For example ArUco marker id "1" is mapped to EM-01-01 which is the identifier for one of the assembly aid trays available in the environment. Table 4 and 5 show the identifier of each object and area in the RoCKIn@Work environment.

Table 4: Objects ID

Object	**Object ID**	**Marker ID**
Assembly aid tray	EM-01-01	1
	EM-01-02	2
	EM-01-02	3
File card box	EM-02-01	21
	EM-02-02	22
	EM-02-03	23
Container Box (ER-02-01 -> ER-02-10)	ER-02-01	61
	ER-02-02	62
	ER-02-03	63
	ER-02-04	64
	...	...
	ER-02-08	68
	ER-02-09	69
	ER-02-10	70

Table 5: Area ID

Object	Object ID	Marker ID
Shelves (SH-01 -> SH-24)	SH-01	81
	SH-02	82
	SH-03	83
	...	...
	SH-22	102
	SH-23	103
	SH-24	104
Drilling workstation	WS-01	131
	WS-02	132
Force fitting workstation	WS-03	133
	WS-04	134
Assembly workstation	WS-05	135
	WS-06	136
	WS-07	137
Conveyor belt	CB-01	161
Drilling machine	DM-01	162

2.4 Networked Devices in the Environment

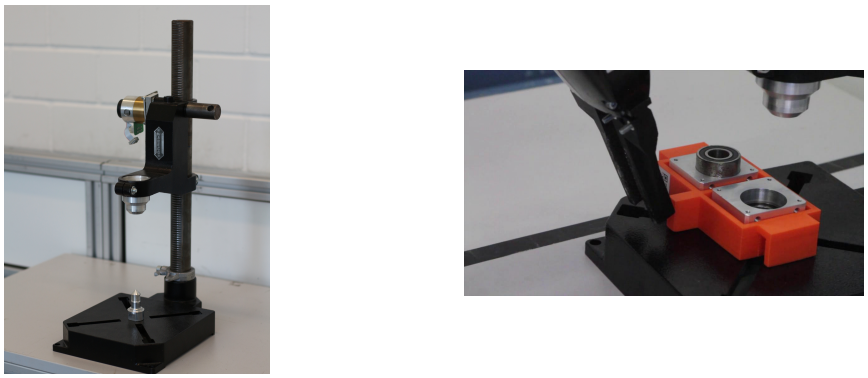
There are four networked devices available in the RoCKIn@Work environment. The four networked devices are: 1. force fitting machine, 2. drilling machine, 3. conveyor belt and 4. quality control camera. The following description provides an overview on the capability of each networked devices and their role in the related task. The interface description for each networked device is provided in detail in the task benchmark section (Section 4).

- *Force fitting machine.* The force fitting machine is used for the insertion of a bearing into a bearing box. The force fitting process is performed by first inserting a bearing box with bearing on top of the bearing box. The placement process is executed with the help of an assembly aid tray. After the bearing box and bearing is properly placed, the force fitting machine is instructed to move down. Finally, the force fitting machine is instructed to move up again and the processed bearing box and bearing can be picked up (Figure 3). The force fitting machine is used in the *prepare assembly aid tray for force fitting* task.

- *Drilling machine.* The drilling machine is used for drilling a cone sink in a cover plate. The drilling machine is equipped with a customized fixture for the plates. Similar to the force fitting machine, the drilling machine is operated by first inserting the cover plate into the fixture of the drilling machine. The cover plate placement is followed by moving the drill head down. Finally, the drill is moved up again and the drilled cover plate can be picked up. The drilling machine is used in the plate drilling task specifically in the correction of a faulty cover plate.

- *Conveyor belt.* The conveyor belt is used for delivering parts to the RoCKIn@Work arena. The conveyor has a quality control camera to detect defects on the parts which are being delivered. The conveyor belt can be started and stopped. The conveyor belt is operated by the quality control camera and both are used for the plate drilling task.

- *Quality control camera.* The quality control camera or QCC is placed on top of the conveyor belt and it is used to acquire information on the quality of incoming cover plates delivery through the conveyor belt. The QCC also has the responsibility to deliver one cover plate through the conveyor belt (until the cover plate reaches the exit ramp of the conveyor belt)



(a) (b)

Figure 3: The RoCKIn@Work force fitting machine (a) and bearing boxes in the middle of force fitting process (b). The first bearing (right side) has been successfully force fitted into a bearing box.

(a) (b)

Figure 4: The RoCKIn@Work drilling machine (a) and a cover plate which is placed in the fixture of the drilling machine (b).

for each received command. After receiving a command, the QCC operates the conveyor belt until a cover plate is within the QCC range of view where the QCC will detect any defects on the cover plate. The conveyor belt will keep moving until it is stopped by QCC when the cover plate is at the exit ramp of the conveyor belt. The QCC it is used for the plate drilling task.

2.5 Central Factory Hub

The main idea of the RoCKIn@Work testbed software infrastructure is to have a central server-like hub (the RoCKIn@Work Central Factory Hub) that serves all the services that are needed for executing and scoring tasks and successfully realize the competition. This hub is derived from software systems well known in industrial business (e.g. SAP). It provides the robots with information regarding the specific tasks and tracks the production process as well as stock and logistics information of the RoCKIn'N'RoLLIn factory. It is a plug-in driven software system. Each plug-in is responsible for a specific task, benchmarking or other functionality.
The following set of specifications must be met by the RoCKIn@Work Central Factory Hub(CFH).

CFH Specification 2.1 (*Central Factory Hub*)

Central managing of all services needed for controlling data, devices and robots. In the following paragraphs, several plug-in will be described. In future, a web-based user interface would be useful.

CFH Specification 2.2 (*Time Table and Scoring Plug-in*)

A plug-in that is able to present actual scoring and time table of upcoming task executions by teams. This is used to keep the audience informed via a big screen or in the internet.

CFH Specification 2.3 (*Benchmarking Plug-in*)

A plug-in to serve general benchmark functionalities or interact with a separated BM software.

CFH Specification 2.4 (*Production Tracking Database Plug-in*)

An automatic plug-in with the central database tracking all identifiers, status of finished products, sub-assemblies, stock level etc.

2.6 Benchmarking Equipment in the Environment

RoCKIn benchmarking is based on the processing of data collected in two ways:

- **internal benchmarking data**, collected by the robot system under test (see Section 3);
- **external benchmarking data**, collected by the equipment embedded into the test bed.

External benchmarking data is generated by the RoCKIn test bed with a multitude of methods, depending on their nature.

One of the types of external benchmarking data used by RoCKIn are pose data about robots and/or their constituent parts. To acquire these, RoCKIn uses a camera-based commercial motion capture system (NaturalPoint OptiTrack), composed of dedicated hardware and software. Benchmarking data has the form of a time series of poses of rigid elements of the robot (such as the base or the wrist). Once generated by the OptiTrack system, pose data are acquired and logged by a customized external software system based on ROS (Robot Operating System): more precisely, logged data is saved as *bagfiles* created with the *rosbag* utility provided by ROS.

Pose data is especially significant because it is used for multiple benchmarks. There are other types of external benchmarking data that RoCKIn acquires: however, these are usually collected using devices that are specific to the benchmark. For this reason, such devices are described in the context of the associated benchmark, rather than here.

Finally, equipment to collect external benchmarking data includes any *server* which is part of the test bed and that the robot subjected to a benchmark has to access as part of the benchmark. Communication between servers and robot is performed via the test bed's own wireless network (see Section 3.2).

3 Robots and Teams

The purpose of this section is twofold:

1. It specifies information about various robot features that can be derived from the environment and the targeted tasks. These features are to be considered at least as desirable, if not required for a proper solution of the task. Nevertheless, we will try to leave the design space for solutions as large as possible and to avoid premature and unjustified constraints.

2. The robot features specified here should be supplied in detail for any robot participating in the competition. This is necessary in order to allow better assessment of competition and benchmark results later on.

The description of the robot should be included in the team description paper.

3.1 General Specifications and Constraints on Robots and Teams

Robot Specification 3.1 (*System*)

A competing team may use a single robot or multiple robots acting as a team. It is not required that the robots are certified for industrial use. At least one of the robots entered by a team is capable of:

- *mobility and autonomous navigation.*
- *manipulate and grasp at least several different task-relevant objects. The specific kind of manipulation and grasping activity required is to be derived from the task specifications.*

The robot subsystems (mobility, manipulation and grasping) should work with the environment and objects specified in this rulebook.

Robot Specification 3.2 (*Sensor Subsystems*)

Any robot used by a team may use any kind of ***onboard*** *sensor subsystem, provided that the sensor system is admitted for use in the general public, its operation is safe at all times, and it does not interfere with other teams or the environment infrastructure. A team may use the sensor system in the environment provided by the organizer by using a wireless communication protocol specified for such purpose. Sensor systems used for benchmarking and any other systems intended for exclusive use of the organisers are not accessible by the robot system. Teams are not allowed to modify the environment or to install their own embedded devices in the environment, e.g., additional sensors or actuators.*

Robot Specification 3.3 (*Communication Subsystems*)

Any robot used by a team may ***internally*** *use any kind of communication subsystem, provided that the communication system is admitted for use in the general public, its operation is safe at all times, and it does not interfere with other teams or the environment infrastructure. A robot team must be able to use the communication system provided* ***as part of the environment*** *by correctly using a protocol specified for such purpose and provided as part of the scenario.*

Robot Specification 3.4 (*Power Supply*)

Any mobile device (esp. robots) must be designed to be usable with an onboard power supply (e.g. a battery). The power supply should be sufficient to guarantee electrical autonomy for a duration exceeding the periods foreseen in the various benchmarks, before recharging of batteries is necessary.

Charging of robot batteries must be done outside of the competition environment. The team members are responsible for safe recharging of batteries. If a team plans to use inductive power transmission devices for charging the robots, they need to request permission from the event organizers in advance and at least three months before the competition. Detailed specifications about the inductive device need to be supplied with the request for permission.

Robot Constraint 3.1 (*Computational Subsystems*)

Any robot or device used by a team as part of their solution approach must be suitably equipped with computational devices (such as onboard PCs, microcontrollers, or similar)

with sufficient computational power to ensure safe autonomous operation. Robots and other devices may use external computational facilities, including Internet services and cloud computing to provide richer functionalities, but the safe operation of robots and devices may not depend on the availability of communication bandwidth and the status of external services.

Robot Constraint 3.2 (*Safety and Security Aspects*)

For any device a team brings into the environment and/or the team area, and which features at least one actuator of any kind (mobility subsystems, robot manipulators, grasping devices, actuated sensors, signal-emitting devices, etc.), a mechanisms must be provided to immediately stop its operation in case of an emergency (emergency stop). For any device a team brings into the environment and/or the team area, it must guarantee safe and secure operation at all times. Event officials must be instructed about the means to stop such devices operating and how to switch them off in case of emergency situations.

Robot Constraint 3.3 (*Operation*)

In the competition, the robot should perform the tasks autonomously. An external device is allowed for additional computational power. It must be clear at all times that no manual or remote control is exerted to influence the behavior of the robots during the execution of tasks.

Robot Constraint 3.4 (*Environmental Aspects*)

Robots, devices, and apparatus causing pollution of air, such as combustion engines, or other mechanisms using chemical processes impacting the air, are not allowed. Robots, devices, and any apparatus used should minimize noise pollution. In particular, very loud noise as well as well-audible constant noises (humming, etc.) should be avoided. The regulations of the country in which a competition or benchmark is taking place must be obeyed at all times. The event organizers will provide specific information in advance, if applicable. Robots, devices, and any apparatus used should not be the cause of effects that are perceived as a nuisance to the humans in the environment. Examples of such effects include causing wind and drafts, strong heat sources or sinks, stenches, or sources for allergic reactions.

3.2 Benchmarking Equipment in the Robots

Preliminary Remark: Whenever teams are required to install some element provided by RoCKIn on (or in) their robots, such element will be carefully chosen in order to minimize the work required from teams and the impact on robot performance.

Hardware As a general rule, RoCKIn does not require that teams install additional robotic hardware on their robots. Moreover, permanent change to the robot's hardware is never required. However, RoCKIn may require that additional standard PC hardware (such as an external, USB-connected hard disk for logging) is temporarily added to the robot in order to collect internal benchmarking data. When this is the case, the additional hardware is provided by RoCKIn during the Competition, and its configuration for use is either automatically performed by the operating system, or very simple.

To allow the acquisition of external benchmarking data about their pose, robots need to be fitted with special reflective markers, mounted in known positions. The teams will be required to prepare their robots so to ease the mounting of the markers. Teams will also be required to provide the geometric transformation from the position the marker to the odometric center of the robot[1].

[1] Benchmarking data related to poses will refer to the marker position: this is why additional information is required to know the position of the base.

Software RoCKIn may require that robots run RoCKIn-provided (or publicly available) software during benchmarks. A typical example of such software is a package that logs data provided by the robot, or a client that interfaces with a RoCKIn server via the wireless network of the test bed. Whenever a team is required to install and run such a package, it will be provided as source code, its usage will be most simple, and complete instruction for installation and use will be provided along with it. All RoCKIn software is written to have a minimal impact on the performance of a robot, both in terms of required processing power and in terms of (lack of) interaction with other modules. When required by a benchmark, the relevant RoCKIn software to be run by participating robots is provided well in advance with respect to the Competition.

RoCKIn will make any effort to avoid imposing constraints on the teams participating to the Competition in terms of software architecture of their robots. This means that any provided piece of software will be designed to have the widest generality of application. However, this does not mean that the difficulty of incorporating such software into the software architecture of a robot will be independent from such architecture: for technical reasons, differences may emerge. A significant example is that of software for data logging. At the moment, it appears likely that any such software by RoCKIn will be based on the established *rosbag* software tool, library and file format. As rosbag is part of ROS (Robot Operating System), robots based on ROS can use it to log data without any modification; on the contrary, robots not using ROS will be required to employ the rosbag library to create rosbag files (*bagfiles*) or to develop ad-hoc code to convert their well established logging format into the rosbag one by using the rosbag API. If this will be the case, RoCKIn will provide tools to ease the introduction of software modules for creation of bagfiles into any software architecture; yet, teams not using ROS will probably have to perform some additional work to use such tools.

3.3 Robot Communication with Benchmarking Equipment

For some types of internal benchmarking data (i.e. provided by the robot), logging is done on board the robot, and data are collected after the benchmark (for instance, via USB stick). Other types of internal benchmarking data, instead, are communicated by the robot to the test bed during the benchmark. In such cases, communication is done by interfacing the robot with standard wireless network devices (IEEE 802.11n) that are part of the testbed, and which therefore become a part of the benchmarking equipment of the test bed. However, it must be noted that network equipment is not strictly dedicated to benchmarking: for some benchmarks, in fact, the WLAN may be also (or exclusively) used to perform interaction between the robot and the test bed.

Due to the need to communicate with the test bed via the WLAN, all robots participating to the RoCKIn Competition are required to:

1. possess a fully functional IEEE 802.11n network interface[2];
2. be able to keep the wireless network interface permanently connected to the test bed WLAN for the whole duration of the benchmarks

3.4 YAML Data File Specification

The subsequent paragraphs specify the YAML file format that can be converted to ROS bag files. This closely follows the data items described in D-2.1.7 [2]. The YAML format was chosen because it is a simple format, easy to produce without using any special library. Furthermore, the ROS messages format is already defined: as produced by the `rostopic echo` command.

[2]It must be stressed that full functionality also requires that the network interface must not be hampered by electromagnetic obstacles, for instance by mounting it within a metal structure and/or by employing inadequate antenna arrangements. Network spectrum in the Competition area is typically very crowded, and network equipment with impaired radio capabilities may not be capable of accessing the test bed WLAN, even if correctly working in less critical conditions.

3.4.1 File Format

The YAML file should be composed of a single list of messages. Each message should have four items:

- `topic` - The topic name.
- `secs` - Timestamp of the message, in number of seconds since 1970.
- `nsecs` - Nanoseconds component of the timestamp.
- `message` - The message, according to the topic type.

The message should be formatted in YAML, according to its structure. This is the same as the output of `rostopic echo`. However, binary fields may be specified in base 64 encoding for much smaller files. You can copy the file `src/base64.hpp` to your project, it depends only on boost to encode base 64.

And example for a file generated according to above specification could look as follows:

```
- topic: pose2d
  secs: 1397024209
  nsecs: 156423000
  message:
   x: 5.5
   y: 6
   theta: 6.4
- topic: image
  secs: 1397024210
  nsecs: 53585000
  message:
   header:
     seq: 306
     stamp:
       secs: 1397024210
       nsecs: 53585000
     frame_id: ''
   height: 4
   width: 4
   encoding: bgr8
   is_bigendian: 0
   step: 12
   data:
     !!binary JaU8JYOkGXUIAZOUDWzgAXjgAbOkIglwbkGsnkWwoiWUfiGUhi2olhmUgc1YRaUw
```


4 Task Benchmarks

Details concerning rules, procedures, as well as scoring and benchmarking methods, are common to all task benchmarks.

Rules and Procedures Every run of each of the task benchmark will be preceded by a safety-check, outlined as follows:

1. The team members must ensure and inform at least one of the organizing committee (OC) member, present during the execution of the task, that they have an emergency stop button on the robot which is fully functional. Any member of the OC can ask the team to stop their robot at any time which must be done immediately.
2. A member of the OC present during the execution of the task will make sure that the robot complies with the other safety-related rules and robot specifications presented in Section 3.

All teams are required to perform each task according to the steps mentioned in the rules and procedures sub-subsections for the tasks. During the competition, all teams are required to repeat the task benchmarks several times. On the last day, only a selected number of top teams will be allowed to perform the task benchmarks again. Maximum time allowed for one task benchmark is 10 minutes.

Acquisition of Benchmarking Data Following some general notes on the acquisition of benchmarking data are described. They are valid for all task benchmarks, as well as for the functional benchmarks.

- **Calibration parameters** Important! Calibration parameters for cameras must be saved. This must be done also for other sensors (e.g., Kinect) that require calibration, if a calibration procedure has been applied instead of using the default values (e.g., those provided by OpenNI).
- **Notes on data saving** The specific data that the robot must save is described in the benchmark section. In general some data streams (those with the highest bitrate) must be logged only in the time intervals when they are actually used by the robot to perform the activities required by the benchmark. In this way, system load and data bulk are minimized. For instance, whenever a benchmark includes object recognition activities, video and point cloud data must be logged by the robot only in the time intervals when it is actually performing object recognition.
- **Use of data** The logged data is not used during the competition. In particular, it is not used for scoring. The data is processed by RoCKIn after the end of the competition. It is used for in-depth analysis and/or to produce datasets to be published for the benefit of the robotics community.
- **Where and when to save data** Robots must save the data as specified in the section "Acquisition of Benchmarking Data" of their respective TBM/FBM on a USB stick provided by RoCKIn. The USB stick is given to the team immediately before the start of the benchmark, and must be returned (with the required data on it) at the end of the benchmark. Each time a teams robot executes a benchmark, the team must:
  1. Create, in the root directory of the USB stick, a new directory named
     - NameOfTheTeam_FBMx_DD_HH-MM (for FBM) or
     - NameOfTheTeam_TBMx_DD_HH-MM (for TBM)
  2. Configure the robot to save the data files in these directories.

In the filenames above x denotes the number of the benchmark, DD is the day of the month and HH, MM represent the time of the day (hours and minutes).

All files produced by the robot that are associated with the execution of the benchmark must be written in this directory. Please note that a new directory must be created for each benchmark executed by the robot. This holds true even when the benchmark is a new run of one that the robot already executed.

During the execution of the benchmark, the following data will be collected[3]. In brackets the expected ROS topics are named. Corresponding data types can be stored in a YAML file 3.4 or rosbag. Following the list of **offline data** to be logged:

Topic	Type	Frame Id	Notes
/rockin/robot_pose[4]	geometry_msgs/PoseStamped	/map	10 Hz
/rockin/marker_pose[5]	geometry_msgsPoseStamped	/map	10 Hz
/rockin/trajectory[6]	nav_msgs/Path	/map	Each (re)plan
/rockin/<device>/image[7]	sensor_msgs/Image	/<device>_frame	–
/rockin/<device>/camera_info[8]	sensor_msgs/CameraInfo	–	–
/rockin/depth_<id>/pointcloud[9]	sensor_msgs/PointCloud2	/depth_<id>_frame	–
/rockin/scan_<id>[10]	sensor_msgs/LaserScan	/laser_<id>_frame	10-40Hz
tf[11]	tf	–	–

Some robots might not have some of the sensors or they might have multiple instances of the previous data (e.g., multiple rgb cameras or multiple laser scanner), in this case you append the number of the device to the topic and the frame (e.g., /rockin/scan_0 in /laser_frame_0). It is possible not to log some of the data, if the task does not require it.

The **online** data part can be found in the description of the respective benchmark.

Communication with CFH The following steps describe the part of the CFH communication that is applicable for all TBMs.

1. The robot sends a **BeaconSignal** message at least every second.
2. The robot waits for a **BenchmarkState** messages. ~~The robot is supposed to start the task benchmark as soon as the state is equal to RUNNING.~~It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.
3. The robot waits for an **Inventory** message from the CFH (which is continuously sent out by the CFH) in order to receive the initial distribution of objects and their locations in the environment.

[3] In the following, 'offline' identifies data produced by the robot, and stored locally on the robot, that will be collected by the referees when the execution of the benchmark ends (e.g., as files on a USB stick), while 'online' identifies data that the robot has to transmit to the CFH during the execution of the benchmark. Data marked neither with 'offline' nor with 'online' is generated outside the robot.

[4] The 2D robot pose at the floor level, i.e., $z = 0$ and only yaw rotation.

[5] The 3D pose of the marker in 6 degrees of freedom.

[6] Trajectories planned by the robot including when replanning.

[7] Image processed for object perception; <device> must be any of stereo_left, stereo_right, rgb; if multiple devices of type <device> are available on your robot, you can append "_0", "_1", and so on to the device name: e.g., "rgb_0", "stereo_left_2", and so on.

[8] Calibration info for /rockin/<device>/image.

[9] Point cloud processed for object perception; <id> is a counter starting from 0 to take into account the fact that multiple depth camera could be present on the robot: e.g., "depth_0", "depth_1", and so on.

[10] Laser scans, <id> is a counter starting from 0 to take into account the fact that multiple laser range finders could be present on the robot: e.g., "scan_0", "scan_1", and so on.

[11] The tf topic on the robot; the tf tree needs to contain the frames described in this table properly connected through the /base_frame which is the odometric center of the robot.

4. The robot waits for an **Order** message from the CFH (which is sent out continuously by the CFH) in order to receive the actual task, i.e., where the objects should be at the end.
5. The task benchmark ends when all objects are at their final location as specified in the **Order** message. After that the robot ~~should send out an empty **BenchmarkFeedback** message~~sends a message of type **BenchmarkFeedback** to the CFH with the *phase_ to_ terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *state* field has changed.

The messages to be sent and to be received can be seen on the Github repository located at [3].

Scoring and Ranking Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes and they are related to the fact that the robot has done the required task or not.

The criterion defining the performance equivalence class of robots is based on the concept of *tasks required achievements*. While the ranking of the robot within each equivalence class is obtained by looking at the performance criteria. In particular:

- The performance of any robot belonging to performance class N is considered as better than the performance of any robot belonging to performance class M whenever $M < N$
- Considering two robots belonging to the same class, then a penalization criterion (penalties are defined according to task performance criteria) is used and the performance of the one which received less penalizations is considered as better
- If the two robots received the same amount of penalizations, the performance of the one which finished the task more quickly is considered as better (unless not being able to reach a given achievement within a given time is explicitly considered as a penalty).

Performance equivalence classes and in-class ranking of the robots are determined according to three sets:

- A set A of **achievements**, i.e., things that should happen (what the robot is expected to do).
- A set PB of **penalized behaviors**, i.e., robot behaviors that are penalized, if they happen, (e.g., hitting furniture).
- A set DB of **disqualifying behaviors**, i.e., robot behaviors that absolutely must not happen (e.g., hitting people).

Scoring is implemented with the following 3-step sorting algorithm:

1. If one or more of the elements of set DB occur during task execution, the robot gets disqualified (i.e. assigned to the lowest possible performance class, called class 0), and no further scoring procedures are performed.
2. Performance equivalence class X is assigned to the robot, where X corresponds to the number of achievements in set A that have been accomplished.
3. Whenever an element of set PB occurs, a penalization is assigned to the robot (without changing its performance class).

One key property of this scoring system is that a robot that executes the required task completely will always be placed into a higher performance class than a robot that executes the task partially. Moreover the penalties do not make a robot change class (also in the case of incomplete task).

4.1 Task *Prepare Assembly Aid Tray for Force Fitting*

This task serves as an example for collecting and assembling parts from different locations. Additionally, the teams can show their robots capability in loading and unloading machines (a well known industrial task). Figure 5 shows the aid tray rack used in the benchmark. On the side of the aid tray unique identifiers are visible to identify the object. The aid tray is a container that can store up to two bearing boxes.

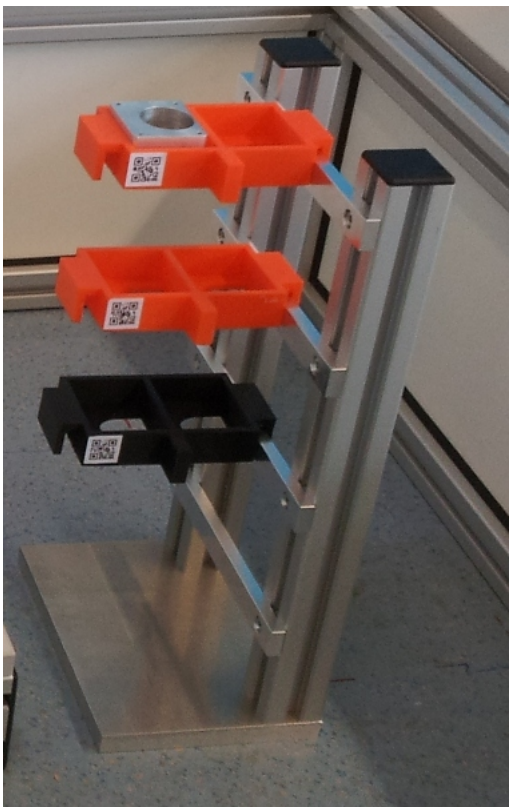


Figure 5: RoCKIn@Work Aid tray rack.

4.1.1 Task Description

The robots task is to collect bearing boxes from stock (shelves) and insert them into specialized aid trays. Once the assembly aid tray is filled with the bearing boxes, these aid trays are loaded to a force fitting machine, where bearings are force fitted into bearing boxes. After the bearing boxes of the assembly aid tray are force fitted, the robot needs to do a final examination before delivering the final product. By scanning the identifiers as part of the task, the robot ensures tracking of the production process and the parts belonging to a particular product itself.

4.1.2 Feature Variation

The bearing boxes can occur in different shapes (see list of parts in Table 1). This is caused by a modular concept of the final product where the bearing box has to be inserted in different chassis. The robots are allowed to collect and insert the bearing boxes in the assembly aid tray individually or collectively.

4.1.3 Input Provided

The team will be provided with the following information:

- description of the set of possible assembly aid tray and bearing boxes.
- description and location(s) of the container(s) used for the bearing boxes.

During the execution of the task, the robot should perform the task autonomously and without any additional input.

4.1.4 Expected Robot Behavior or Output

The robot goes to the central station and registers to the Central Factory Hub. The communication with the CFH is described in detail in Section 4. After receiving the task of Assembly Aid Tray for Force Fitting, the robot locates the assembly aid tray in the *shelf*. The robot proceeds with identifying the identifiers on the assembly aid tray. The identifier contains the information regarding the assembly aid tray's serial number and the type of the bearing box which can be fitted. Based on the examination of the assembly aid tray, the robot needs to find the correct bearing boxes in the set of shelves area. After finding the right bearing boxes, the robot records the identifiers of their containers, collects the bearing boxes and places them into the assembly aid tray. The robot has the option to deliver the bearing boxes collectively or individually. After placing the bearing boxes in the assembly aid tray, the robot delivers the assembly aid tray to the force fitting workstation. In the force fitting workstation, the assembly aid tray will be processed and the robot will be informed when the process is completed. The robot will check the final product and can request for another force fitting process when the result is unsatisfactory.

4.1.5 Procedures and Rules

During the execution of this task, which needs to be carried out as per the next steps, an additional robot might be randomly moving in the arena which has to be avoided by the participating robot.

Step 1 The robot is provided with multiple assembly aid trays and the information regarding the storage area of the bearing boxes.

Step 2 Based on the identifier provided beforehand to the teams, the robot must identify the appropriate bearing boxes needed to be put on a tray.

Step 3 The robot must pick (from the storage area) and insert the bearing boxes, identified in Step 2 above, in the provided assembly aid tray.

Step 4 The robot must deliver the assembly aid tray (with the bearing boxes) to the force fitting workstation to be processed.

4.1.6 Communication to CFH

For this task benchmark the robot does not have to control any networked device in the environment. The force fitting machine will be operated by a human worker. All necessary CFH communication is described in Section 4.

4.1.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in section 4. There, also the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- assembly_aid_tray_id (type: string)
- container_id (type: string)

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/qrcode[12]	std_msgs/Int32	–	when recognized

4.1.8 Scoring and Ranking

Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes. Classes are defined in dependence to:

1. The fact that the robot correctly identifies the assembly aid tray or not;
2. ~~The fact that the robot correctly identifies the container or not~~;
3. The number of bearing boxes successfully inserted by the robot into the aid tray;
4. The successful execution of the force fitting procedure.

Achievments The set A of achievements for this task includes:

- The robot correctly identifies the assembly aid trays identifier.
- ~~The robot correctly identifies the containers identifier.~~
- The robot correctly grasps the assembly aid tray.
- The robot correctly grasps the first bearing box.
- The robot correctly grasps the second bearing box.
- The robot inserts the first bearing box into the aid tray.
- The robot inserts the second bearing box into the aid tray.
- The robot correctly delivers the tray to the force fitting station.
- The robot completely processes a part (from identifying to delivering)
- The robot cooperates with CFH and Networked Devices throughout the task
- The team delivers the benchmarking data appropriatly

Penalties The set PB of penalized behaviours for this task includes:

- The robot bumps into obstacles in the test bed.
- The robot drops an object.
- The robot stops working.

[12] ID of the assembly aid tray or container, deteted by the robot by analyzing the QR code.

Disqualifying Behaviours The set DB of disqualifying behaviours for this task are:

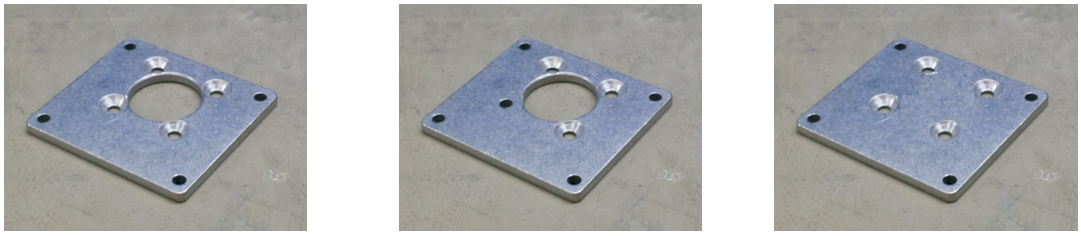
- The robot damages or destroys the objects requested to manipulate.
- The robot damages the test bed.

4.2 Task *Plate Drilling*

This task simulates handling of an incomplete or faulty parts from an external component supplier. The factory has to quickly react on such issues and create a process to correct the faulty parts.

4.2.1 Task Description

The cover plate of the bearing box has eight holes for connecting the motor with the bearing box and the four central holes need to have a cone sink (see Figure 6(a)). There are two possible defects of a cover plate which need to be accommodated in this task. The first case is where the supplier forgot to drill one of the cone sinks which results in a faulty cover plate (see Figure 6(b)). The faulty cover plates can be corrected by drilling the cone sink with the drilling machine available in the factory. The second case is where the cover plate is unusable (see Figure 6(c)) and needs to be returned to the supplier for replacement.



(a) Perfect (b) Faulty (c) Unusable

Figure 6: Three possible states of the cover plate

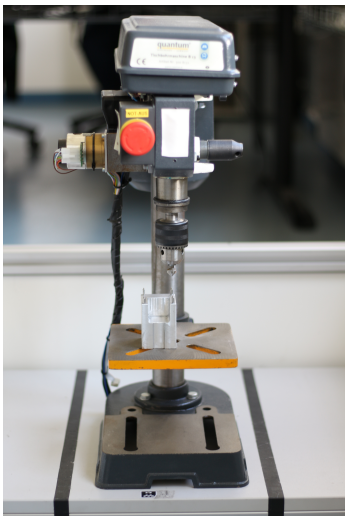
The robot has to pick up the incoming cover plates and process each cover plate based on its type accordingly. The robot performs the task with two networked devices in the factory. The first networked device is the ~~quality control camera or QCC~~triggered conveyor belt or TCB which is ~~placed on top of~~a composite of the quality control camera and the conveyor belt (see Figure 7(b)). The ~~QCC~~TCB is responsible for delivering the cover plate (by operating the conveyor belt) and detecting the type of the cover plate being delivered. The second networked device is the drilling machine (see Figure 7(a)) which is operated by the robot to drill a cone sink to the faulty cover plate.

4.2.2 Feature Variation

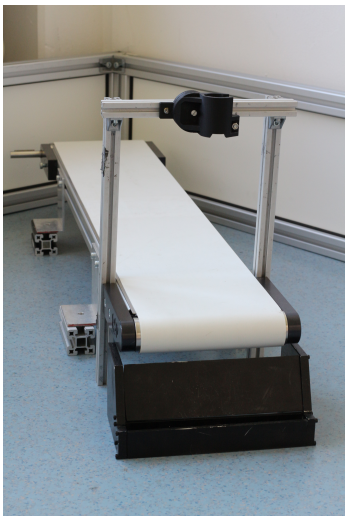
The task can have different variations as shown in the following examples.

- The sequence of faulty, unusable and perfect plates flowing through the conveyor belt.
- The cover plate orientation on the conveyor belt.
- The number of plates delivered in each category (faulty, unusable and perfect).

Furthermore, the solutions can vary depending on the sequence of activities being performed by the robot. The robot can choose to:



(a) Drilling machine



(b) Quality control camera

Figure 7: Networked devices for the plate drilling task

- collect all cover plates from the conveyor belt first and process them collectively or
- perform the task for one cover plate at a time before collecting the next cover plate from the conveyor belt

4.2.3 Input Provided

The team will be provided with the following information:

- 3D CAD textured models of the plates
- Description of three different states of the plate (faulty, unusable, perfect). The three different states of the cover plate are shown in Figure 6.
- Location of objects related to the task.
- ~~Commands for operating the QCC and the drilling machine.~~

4.2.4 Expected Robot Behavior or Output

The robot starts by receiving the task from the CFH. The communication with the CFH is described in detail in 4 and 4.1.6. While performing the task, the robot has control over the ~~QCC~~TCB which allows the robot to regulate the flow of the incoming cover plates. The robot send a command to the CFH to operate the ~~QCC~~TCB. Once the ~~QCC~~TCB receives a command from the CFH, the ~~QCC~~TCB will operate the conveyor belt until a cover plate is placed on the exit ramp of the conveyor belt. During this process, the QCC will detect the type of cover plate which is being delivered (either faulty or unusable) and the TCB will broadcast this information to the CFH. Finally, the CFH will broadcast this information so that the robot will know that a faulty or an unusable cover plate is placed on the exit ramp of the conveyor belt.

For each cover plate that arrives in the conveyor belt exit ramp, the robot needs to process them according to their type. The unusable cover plate needs to be delivered to the trash container box (see Figure 8(a)) in the factory. For faulty cover plates, the robot needs to perform correction to the cover plates by delivering them to the drilling machine, operate the drilling machine to fix the missing cone sink and place the corrected cover plate in the file card box (see Figure 8(b)).

(a) Trash container box

(b) Cover plate file card box

Figure 8: Designated storage for each cover plate type.

4.2.5 Procedures and Rules

Step 1 The robot ~~must control the QCC until it receive a feedback that a cover plate of type unusable or faulty is in the conveyor belt's exit ramp~~commands the TCB to provide a cover plate in the conveyor belt's exit ramp and waits for the result of the plate recognition.

Step 2 The robot should pick up the cover plate and either:

- proceed with processing the cover plate received as described in [Step 3] or
- request for another cover plate as described in [Step 1]

Step 3 There are two possible sequence of actions that need to be executed by the robot depending on the type of the cover plate. If the cover plate is faulty, the robot needs to deliver this to the trash container box. If the cover plate is unusable, the robot needs to:

- place the cover plate inside the drilling machine
- perform correction of the cover plate with the drilling machine
- place the corrected cover plate in the file card box.

4.2.6 Communication with CFH

The communication and interaction between the robot and the networked device is as follows:

- ~~**Quality control camera** or **QCC**~~**Triggered conveyor belt** or TCB. ~~The robot can command the QCC to perform its task which includes operating the conveyor belt and detecting the type of cover plate [3].~~ This TCB is a composite of the quality control camera and the conveyor belt. The operation of the TCB involves the message types TriggeredConveyorBeltCommand [3], TriggeredConveyorBeltStatus [3] and Inventory [3] For each TriggeredConveyorBeltCommand::START command and *next cycle id* received, the ~~QCC~~TCB will run the conveyor belt until~~place~~ a cover plate has been recognized and placed in the conveyor belt exit ramp and stop the belt again.

  The *next cycle id* is used to determine the cycle for which the command is executed. It must be exactly one greater than the cycle received via the TriggeredConveyorBeltStatus (next_cycle = cycle + 1). The cycle determines how often the TCB has already been activated. Every time that the TCB is commanded by a robot, the cycle counter is increased.

  The TCB provides information on the type of cover plate (*faulty* or *unusable*) by updating the inventory of the CFH. ~~The robot can acquire the status of the QCC which is being broadcast continuously (e.g. idle state, current operation and availability of unusable/faulty cover plate in the exit ramp of the conveyor belt).~~

- **Drilling machine**. The operation of the drilling machine involves message type DrillingMachineCommand [3] and DrillingMachineStatus [3] The drill of the drilling machine will spin continuously and the robot needs to command the drill to move down and up (message of type DrillingMachineCommand [3]). In order to move down the drill, the variable command needs to be set to Command::MOVE_DOWN. To move the drill up again the variable needs to be set to Command::MOVE_UP. An example in C++ is provided [4]. Additionally, the CFH sends a status message of type DrillingMachineStatus [3] indicating whether the drill is at the top position (state is equal to State::AT_TOP), at the bottom position (state is equal to State::AT_BOTTOM), moving down (state is equal to State::MOVING_DOWN) or moving up (state is equal to State::MOVING_UP). In case of a problem, the state is equal to State::UNKOWN.

4.2.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in section 4.

Online Data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- after_receiving (type: PlateState)
- after_drilling (type: PlateState)

The **BenchmarkFeedback** message can be found at [3].

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/dril_command[13]	std_msgs/Int32	–	when issued
/rockin/qcc_command[14]	std_msgs/Int32	–	when issued
/rockin/plate_condition[15]	std_msgs/Int32	–	when issued

4.2.8 Scoring and Ranking

Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes. Classes are defined in dependence to:

1. the number and percentage of correctly processed faulty cover plates;
2. the number and percentage of correctly processed unusable cover plates;
3. execution time (if less than the maximum allowed for the benchmark).

Achievements The set A of achievements for this task includes:

- the robot communicates with the CFH throughout the test;
- the team submits the benchmarking data appropriately by the end of the test;
- the robot picks up a cover plate from the conveyor belt's exit ramp;
- the robot places an unusable cover plate into the trash container box;

[13] Drilling commands issued by the robot
[14] QCC commands issued by the robot
[15] Condition of each plate, as evaluated by the robot, after drilling

- the robot completely processes an unusable cover plate (pick up an unusable cover plate from the exit ramp of the conveyor belt and place it into the trash container box);
- the robot places a faulty cover plates inside the drilling machine;
- the robot performs the drilling of a faulty cover plate using the drilling machine;
- the robot completely corrects a faulty cover plate (pick up a faulty cover plate from the exit ramp of the conveyor belt, place it inside the drilling machine and operate the drilling machine);
- the robot picks up a corrected cover plate from the drilling machine;
- the robot places a corrected cover plate into the file card box;
- the robot completely delivers a corrected cover plate (pick up a corrected cover plate from the drilling machine and place it inside the file card box).

Penalties The set PB of penalized behaviours for this task includes:

- the robot bumps into obstacles in the test bed;
- the robot drops an object;
- the robot stops working.

Disqualifying Behaviours The set DB of disqualifying behaviours for this task are:

- The robot damages or destroys the objects requested to manipulate;
- The robot damages the test bed.

4.3 Task *Fill a Box with Parts for Manual Assembly*

This task reflects one of the primary requisites of a mobile robotic service assistant, i.e., to work together with humans. In this case the goal is to assist humans at a manual assembly workstation.

4.3.1 Task Description

The robot must compose boxes with parts for the manual, final assembly of the drive axle. The boxes have no special subdivisions; they only can have foam material at the bottom to guarantee safe transport. Therefore, the robot has to plan the order of collecting the parts to arrange them next to each other.

4.3.2 Feature Variation

The standardized boxes (see figure 9) can be used for several groups of parts. Because of variations in containing parts (e.g., bearing box variations) the groups of parts in this task vary the same way.

4.3.3 Input Provided

The team will be provided with the following information:

- The list of possible parts used in the task;
- Description of the box used for collecting the parts;
- Location of the parts in the arena.

Figure 9: RoCKIn@Work container with identifier

4.3.4 Expected Robot Behavior or Output

The task execution is triggered by the robot receiving the list of parts required for the assembly process from the CFH. The communication with the CFH is described in detail in sections 4 and 4.3.6. The robot proceeds with collecting an empty box from the shelves and begins collecting the parts (individually or collectively). When the parts are placed in the box, the robot needs to deliver the box to the assembly workstation and provide the human worker with the list of parts in the box and, (if exist any), the missing parts.

4.3.5 Procedures and Rules

There can be multiple obstacles present in the scene that might block the direct path of the competing robot. The robot must avoid all the obstacles/other robots during the execution of the next steps in this task.

Step 1 The robot will receive an order for a final assembly step of a product from the CFH containing a list of objects to be collected and delivered.

Step 2 The robot must plan the best path to the designated workstation, passing through each storage area where the required objects for a requested product in the list can be found.

Step 3 The robot must execute the above path, collect the objects and then deliver them to the designated area for assembly of that product.

Step 4 The Steps 2 and 3 above must be done for all the products in the list mentioned in Step 1. ~~Also, the robot must follow, as much as possible, the priorities as per the philosophy of *first-in first-out* of the products when executing Step 2 and 3.~~

4.3.6 Communication with CFH

For this task benchmark the robot does not have to control any networked device in the environment. Therefore no communication except the one described in 4 is necessary.

4.3.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in section 4.

Online Data

- No online benchmarking data has to be sent to the CFH during this task benchmark.

Topic	Type	Frame Id	Notes
/rockin/notification[16]	std_msgs/String	–	–

Offline data The additional information described in the following table has to be logged:

4.3.8 Scoring and Ranking

Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes. Classes are defined in dependence to the number of parts of the product to be assembled actually provided by the robot to the human worker and their order according to the desired one.

Achievements The set A of achievements for this task consists of:

- The robot communicates with the CFH throughout the test;
- The team submits the benchmarking data by the end of the test;
- The robot picks up a required object (also the container) from its storage location;
- The robot places the required objects into the container;
- ~~The robot delivers a correctly filled container to the designated workstation.~~
- The robot has correctly filled the container and the container is on top of the designated workstation.

Penalties The set PB of penalized behaviours for this task include:

- The robot bumps into obstacles in the test bed;
- The robot drops an object;
- The robot stops working.

Disqualifying Behaviours The set DB of disqualifying behaviours for this task are:

- The robot damages or destroys the objects requested to manipulate;
- The robot damages the test bed.

5 Functionality Benchmarks

Communication with CFH ~~The communication between the CFH and the robot should look like:~~The communication between the CFH and the robot is described in the respective section of each functionality benchmark.

~~**Step 1** The robot sends a **BeaconSignal** message at least every second~~

~~**Step 2** The robot waits for a **BenchmarkState** message. It is supposed to start with testing the functionality as soon as state is equal to RUNNING.~~

[16] The string with the notification of the perceived object should be in a tab separated string: CLASS OBJECT_ID X Y THETA

~~**Step 3** As soon as the robot has produced the benchmarking data, which has to be sent to the CFH online, it has to send a message of type **BenchmarkFeedback** with the required results back to the CFH. The robot should do this until the *state variable* of the **BenchmarkState** messages changes from RUNNING to STOPPED. The robot should continue with Step 2.~~

~~**Step 4** The functionality benchmark ends when the *state variable* of the **BenchmarkState** message changes to FINISHED~~

~~The messages to be sent and to be received can be seen on the Github repository located at [3].~~

5.1 Object Perception Functionality

5.1.1 Functionality Description

This functionality benchmark has the objective of assessing the capabilities of a robot in processing sensor data in order to extract information about observed objects. Objects presented to the robot in this functionality benchmark are chosen to be representative of the type of factory scenario that RoCKIn@Work is based on. Teams are provided with a list of individual objects (object instances), subdivided into classes (see Section 5.1.3). The benchmark requires that the robot, when presented with objects from such list, detects their presence, estimate their class, identity and location. As for example when presented with a segment of T-section metal profile the robot should detect it is in front of a profile (class) with a T-shaped section (instance) and that it is at a given position with respect to a known reference frame (this will be the benchmark setup reference frame 10). Objects that are used here are described in Section 2.

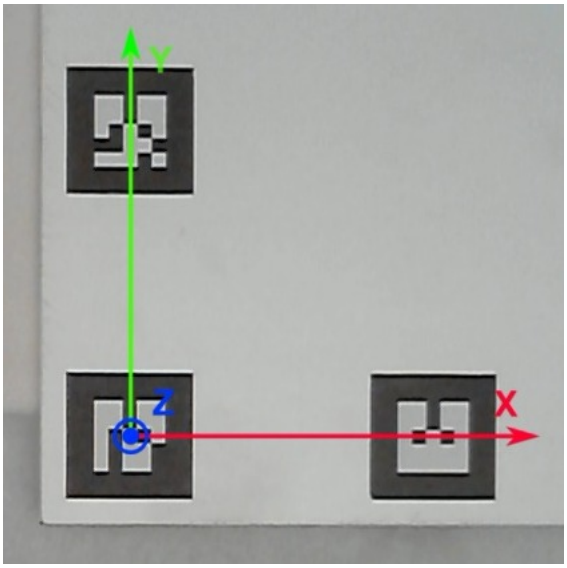


Figure 10: RoCKIn@Work Table reference frame example

5.1.2 Feature Variation

For this benchmark, the variation space for object features is represented by the (known) set of objects that the robot may be presented with. Variation space for object location is the surface of the benchmarking area where objects are located (see Section 5.1.3).

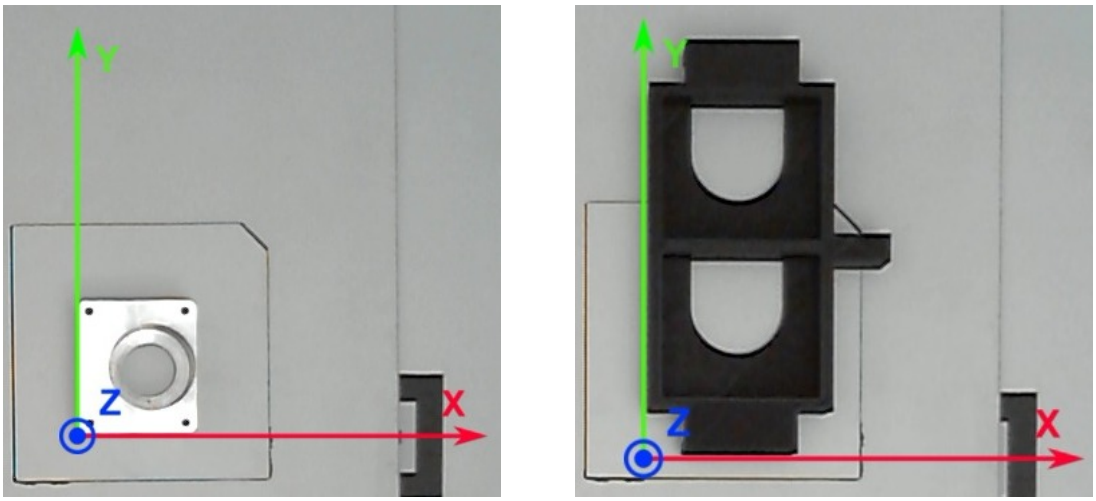
5.1.3 Input Provided

The set of individual objects that will actually be presented to the robot during the execution of the functionality benchmark is a subset of a larger set of available objects ("object instances"). Object instances are subdivided into classes of objects that have one or more properties in common ("object classes"). Objects of the same class share one or more properties, not necessarily related to their geometry (for instance, a class may include objects that share their application domain). Each object instance and each object class is assigned a unique ID.

All object instances and classes are known to the team before the benchmark, but the team does not know which object instances will actually be presented to the robot during the benchmark. More precisely, the team will be provided with the following information:

- Descriptions/models of all the object instances in the form of 3D textured models;
- Subdivision of the object instances into object classes (for instance: profiles, screws, joints);
- Reference systems associated to the table surface and to each object instance (to be used to express object poses).

Object descriptions will be expressed according to widely accepted representations, well in advance of the competition. Figure 11(a) and 11(b) show examples of the objects and their reference frames.



(a) BearingBox (b) AidTray

Figure 11: Examples of objects and their reference frame

5.1.4 Expected Robot Behavior or Output

The objects that the robot is required to perceive are positioned, one at the time, on a table located directly in front of the robot.

The actual pose of the objects presented to the robot is unknown before they are set on the table. The robot, for each object presented to it, must perform all of the following:

- Object detection: perception of the presence of an object on the table and association between the perceived object and one of the object classes (see 5.1.3).

- Object recognition: association between the perceived object and one of the object instances belonging to the selected class (see 5.1.3).
- Object localization: estimation of the 3D pose of the perceived object with respect to the surface of the table.

The following list of classes and instances of objects are going to be used in the object perception functionality benchmark:

- Containers:
  1. EM-01 (aid tray) 2. EM-02 (cover plates file card box)
- Bearing boxes:
  1. AX-01 (bearing box type A) 2. AX-16 (bearing box type B)
- Transmission parts:
  1. AX-02 (bearing) 2. AX-09 (motor with gearbox) 3.AX-03 (axis)

5.1.5 Procedures and Rules

Every functionality benchmark will be preceded by a safety-check similar to that described for the task benchmark procedures.

All teams are required to perform this functionality benchmark according to the steps mentioned below. During the competition, all teams are required to repeat it several times. On the last day, only a selected number of top teams will be allowed to perform it. The maximum time allowed for one functionality run is 2 minutes. A run is defined as the detection, recognition and localization of one object.

Step 1 An object of unknown class and unknown instance will be placed in front of the robot

Step 2 The robot must determine the objects class, its instance within that class as well as the 3D pose of the object and save it in the given format (see Section 5.1.7)

Step 3 The preceding steps are repeated until ~~time runs out or 10 objects have been processed~~10 runs have been completed.

5.1.6 Communication with CFH

For this functionality benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot has finished perceiving the object, it sends a message of type **BenchmarkFeedback** to the CFH with the required results and the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *state* field has changed.

Step 4 The robot continues with Step 2.

Step 5 The functionality benchmark ends when the **BenchmarkState**'s *state* field is equal to FINISHED.

Step 2 ~~The robot waits for a **BenchmarkState** message. It is supposed to start with testing the functionality as soon as state is equal to RUNNING.~~

Step 3 ~~As soon as the robot has produced the benchmarking data, which has to be sent to the CFH online, it has to send a message of type **BenchmarkFeedback** with the required results back to the CFH. The robot should do this until the *state variable* of the **BenchmarkState** messages changes from RUNNING to STOPPED. The robot should continue with Step 2.~~

Step 4 ~~The functionality benchmark ends when the *state variable* of the **BenchmarkState** message changes to FINISHED~~

The messages to be sent and to be received can be seen on the Github repository located at [3].

5.1.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in section 4. There, also the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- object_class_name (type: string)
- object_instance_name (type: string))
- object_pose (type: Pose3d)

The **BenchmarkFeedback** message can be found at [3].

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/notification[17]	std_msgs/String	–	–

5.1.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. The number and percentage of correctly classified objects;
2. The number and percentage of correctly identified objects;
3. Pose errors for correctly identified objects as measured by the ground truth system;
4. Execution time (if less than the maximum allowed for the benchmark).

Being this functionality benchmark focused on object recognition, the previous criteria are in order of importance; the first criterion is applied first and teams will be scored according to their accuracy, the ties are broken by using the second one still applying accuracy metrics, finally the position error is evaluated as well. Since the position error is highly affected by the precision of the ground truth system we will use a set of "distance classes" and in case of ties we will resort to execution time.

[17] The string with the notification of the perceived object should be in a tab separated string: CLASS OBJECT_ID X Y THETA

5.2 Manipulation Functionality

5.2.1 Functionality Description

This functionality benchmark assesses the robot's capability of grasping different objects. An object from a known set of possible objects is presented in the test for the robot to be identified and grasped. After identifying the object, the robot needs to perform the grasping motion, lift the object and notify that the object is acquired.

5.2.2 Feature Variation

The objects used in the benchmark will be selected from the list of parts to manipulate as presented in the Section 2.2. Additionally, the precise position of the object differs in each test.

5.2.3 Input Provided

The team will be provided with the following information:

- The list of possible objects used in the functionality benchmark.
- Possible placement of each object used in the functionality benchmark.

5.2.4 Expected Robot Behaviour or Output

The robot is placed in front of the test area (a planar surface). One object will be placed in the test area and the robot will identify the object and move its end effector on top of it. Afterwards, the robot performs the grasping motion of the object and notifies that the grasping has occurred. The task is repeated with different objects.

The following list of classes and instances of objects are going to be used in the manipulation functionality benchmark:

- Containers:
  1. EM-01 (aid tray) 2. EM-02 (cover plates file card box)
- Bearing boxes:
  1. AX-01 (bearing box type A) 2. AX-16 (bearing box type B)
- Transmission parts:
  1. AX-02 (bearing) 2. AX-09 (motor with gearbox) 3.AX-03 (axis)

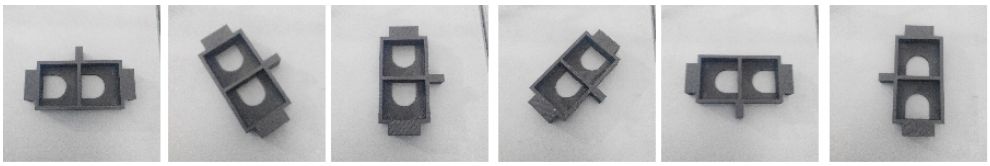
Figure 12 shows the possible placement for each object.

5.2.5 Procedures and Rules

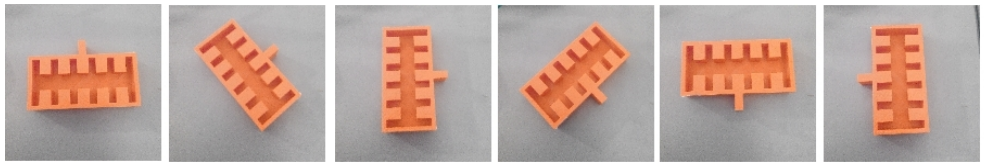
Every functionality benchmark will be preceded by a safety-check similar to that described for the task benchmark procedures.

All teams are required to perform this functionality benchmark according to the steps mentioned below. During the competition, all teams are required to repeat it several times. On the last day, only a selected number of top teams will be allowed to perform it. The maximum time allowed for one functionality run is 4 minutes (30 seconds for preparation and 210 seconds for execution). A run consists of (1) a preparation phase where the robot is going to its initial configuration and releases a previously grasp object and (2) an execution phase in which the robot detects, localizes, recognizes and manipulates one object. ~~Note that all objects within this functionality benchmark are from within the RoCKIn@Work environment.~~

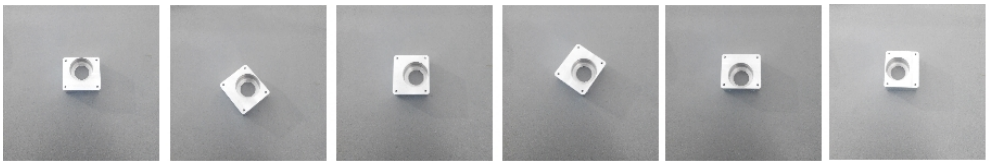
Step 1 An object of unknown class and unknown instance will be placed on a table in front of the robot.



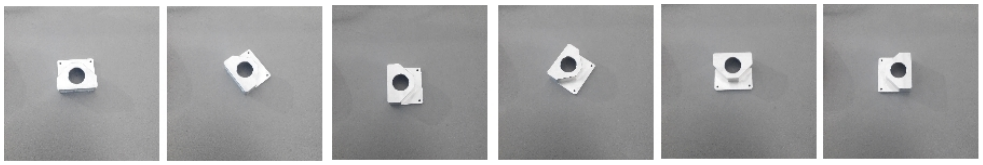
(a) EM-01, aid tray



(b) EM-02, cover plate file card box

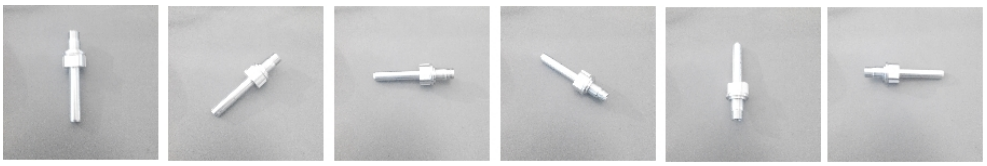


(c) AX-01 bearing box type A

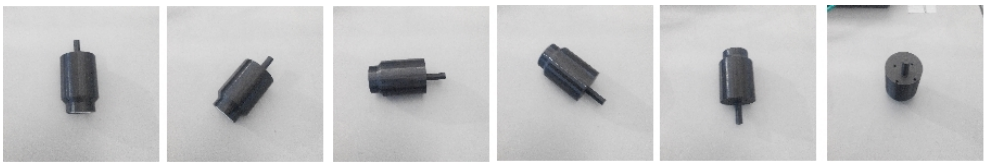


(d) AX-16, bearing box type B

(e) AX-02, Bearing



(f) AX-03, axis



(g) AX-09, motor with gearbox

Figure 12: Object placement for manipulation functionality benchmark

Step 2 The robot must determine the object's class and its instance within that class.

Step 3 The robot must grasp the object, lift it, and notify that grasping has occurred.

Step 4 The robot must keep the grip for a given time while the referee verifies the ~~lifting~~correct manipulation of the object.

Step 5 The preceding steps are repeated with five different objects.

For each presented object, the robot must produce the result consisting of:

- object class name [string], e.g. containers.
- object instance name [string], e.g. EM-02.

5.2.6 Communication with CFH

For this functionality benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the preparation procedure when the *phase* field is equal to PREPARATION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot finishes the preparation phase, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to PREPARATION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 4 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 5 As soon as the robot has finished manipulating the object, it sends a message of type **BenchmarkFeedback** to the CFH with the required results and the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 6 The robot continues with Step 2.

Step 7 The functionality benchmark ends when the **BenchmarkState**'s *phase* field is equal to EXECUTION and the *state* field is equal to FINISHED.

Step 2 ~~The robot waits for a **BenchmarkState** message. It is supposed to start with testing the functionality as soon as state is equal to RUNNING.~~

Step 3 ~~As soon as the robot has produced the benchmarking data, which has to be sent to the CFH online, it has to send a message of type **BenchmarkFeedback** with the required results back to the CFH. The robot should do this until the *state variable* of the **BenchmarkState** messages changes from RUNNING to STOPPED. The robot should continue with Step 2.~~

Step 4 ~~The functionality benchmark ends when the *state variable* of the **BenchmarkState** message changes to FINISHED.~~

The messages to be sent and to be received can be seen on the Github repository located at [3].

5.2.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in section 4. There also the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- grasp_notification (type: bool)
- object_class_name (type: string)
- object_instance_name (type: string)
- end_effector_pose (type: Pose3D)

The **BenchmarkFeedback** message can be found at [3].

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/grasping_pose[18]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/gripper_pose[19]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/arm_joints[20]	geometry_msgs/JointState	/base_link	10 Hz

5.2.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. Number and percentage of correctly identified objects;
2. Number and percentage of correctly grasped objects (the object stops touching the table, see definition below);
3. Execution time (if less than the maximum allowed for the benchmark).

Since this functionality benchmark focuses on manipulation, the scoring of teams is based on the number of objects correctly grasped. A correct grasp is defined as the object being lifted from the table so to be possible for the judge to pass a hand below it. For a grasping to be "correct" the position has to be kept for at least 5 seconds from the time the judge has passes the hand below the object. The time the judge will require to verify the lifting of the object might be up to 10 seconds. In case of ties the overall execution time will be taken into account.

5.3 Control Functionality

5.3.1 Functionality Description

This functionality benchmark assesses the robot's capability to control the manipulator's (and the mobile platform's) motion in a continuous manner. This functionality is essential for precise placement of objects (see TBM2 in Section 4.2) or following a given line in common industrial applications like welding or gluing. A marker set is attached to the robot's manipulator. With the tip of this marker set, the robot has to follow a given path in Cartesian space. The external ground truth system measures the deviation between the given path and the path executed by the robot using this marker. Only for visualization purposes, the path is also displayed on the table (see Figures 14 and 15).

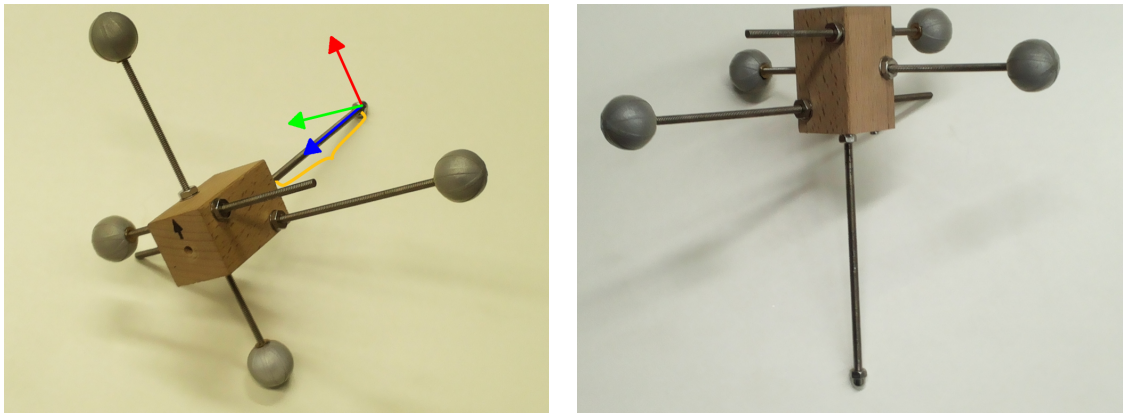
[18] Pose of the grasping position on the object.
[19] Pose of the gripper.
[20] Joints data

5.3.2 Feature Variation

The path is a segment of either a straight line or a sine function. In future competitions the path can be given as a trajectory including velocities and accelerations. The path can be limited to the workspace for a manipulator or of a size that forces the mobile platform to move as well (future competitions).

5.3.3 Input Provided

Marker set An image of the marker set is depicted in Figure 13. The rod of this marker set is attached either directly to the end-effector or to the tool on the robot's manipulator. The tip of the marker set is defined as the center of the spherical cap nut at the end of the rod. This tip is indicated by the coordinate frame in Figure 13. Note that in this FBM only the origin of the coordinate frame is important, while the orientation is irrelevant!



(a) (b)

Figure 13: The marker set used in FBM3. The rod is marked by the yellow brace. The tip of the marker set is marked by the origin of the coordinate frame (the orientation of the coordinate frame is irrelevant here).

Path specification The path to be executed by the robot is specified by:

- a mathematical description of a line in 2D space (see Figure 14 for an example):

  $$f(x) = m \cdot x$$

  where

  - x is a distance in metres $[m]$ along the x-axis of the robot-selected coordinate system
  - $f(x)$ is a distance specified in metres $[m]$ along the y-axis of the robot-selected coordinate system
  - m is the slope specified as a dimensionless scalar

- a mathematical description of a sine in 2D space (see Figure 15 for an example):

  $$f(x) = a \cdot \sin(c \cdot x)$$

  where

  - x is a distance in metres $[m]$ along the x-axis of the robot-selected coordinate system

  - $f(x)$ is a distance specified in metres $[m]$ along the y-axis of the robot-selected coordinate system
  - a is the amplitude specified in metres $[m]$
  - c is a conversion from a distance to an angle specified in radians per metre $[\frac{rad}{m}]$.
- distance between the calibration point and starting point specified in metres $[m]$.
- start of the path, specified in the domain of the mathematical description.
- end of the path, specified in the domain of the mathematical description.
- the concrete selection of the parameters for each benchmark will be announced before the competition.

To exemplify the definition of start and end of the path assume a sine with $a = 0.1\ m$ and $c = 12.5 \cdot \pi\ \frac{rad}{m}$. Let the start of the path be given as $start = 0\ m$ and the end of the path be given as $end = 0.2\ m$. Then the start point is $f(start = 0\ m) = 0\ m$ and the end point is $f(end = 0.2\ m) = 0.1\ m$.

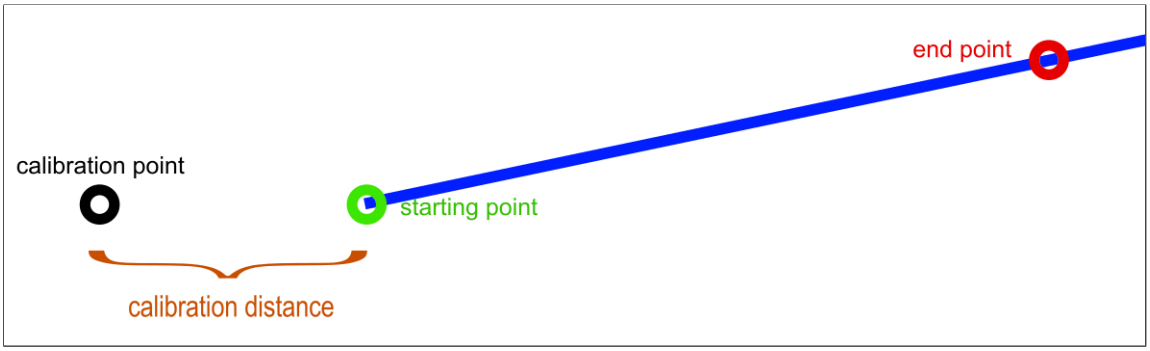


Figure 14: Example of a path on a straight line.

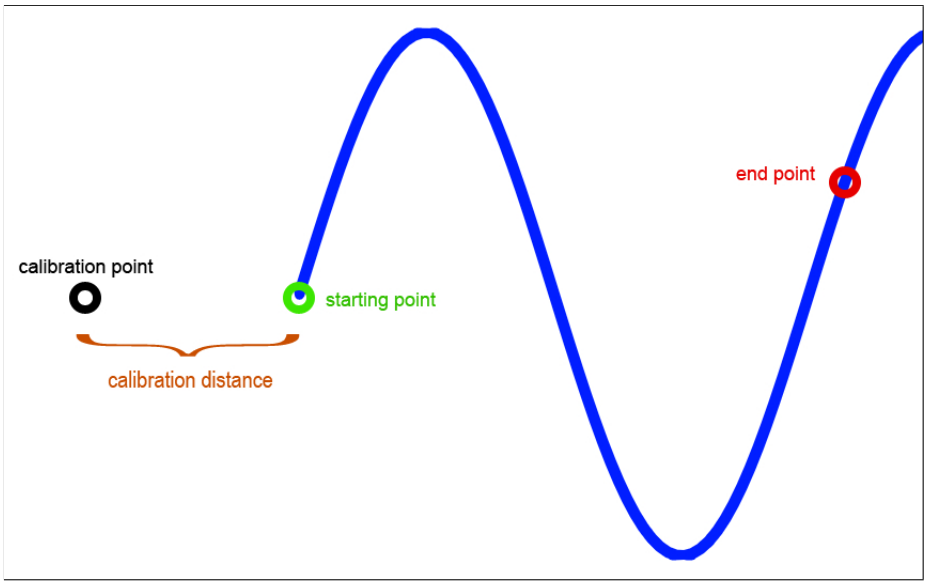


Figure 15: Example of a path on a sine.

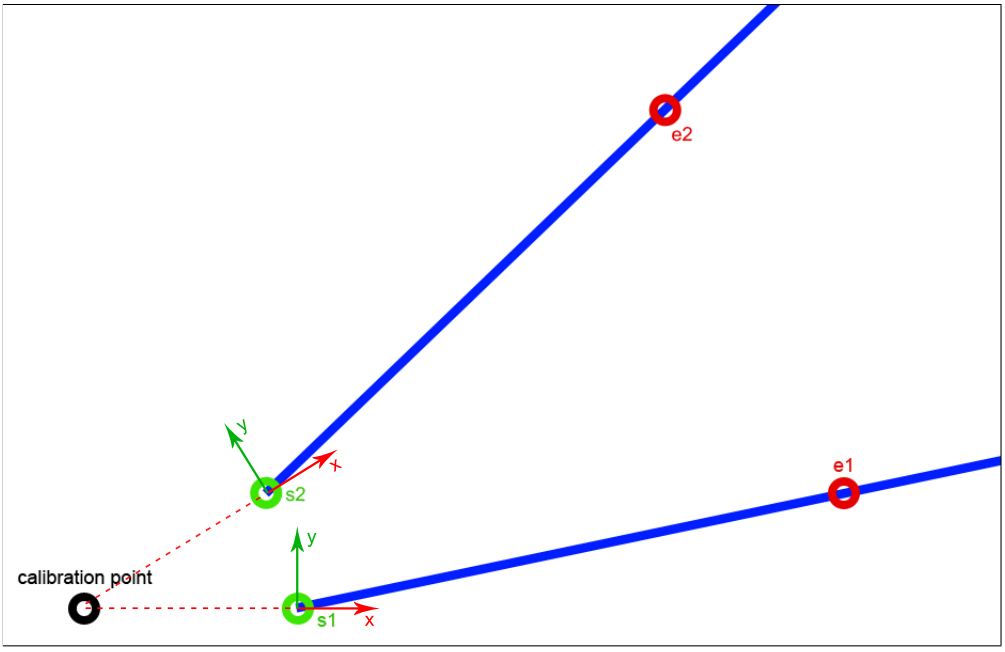


Figure 16: Example of two different starting points, $s1$ and $s2$. Note that the orientation of the path depends on the starting point.

5.3.4 Expected Robot Behaviour or Output

The robot is placed in front of the test area (a planar surface). The robot is required to execute the provided path in a plane above the test area that is parallel to the ground, where the height above the ground is selected by the robot. Because the robot is free to select the reference frame w.r.t. which the path is executed, it must synchronize its internal reference frame with the benchmarking system's reference frame. This is achieved by a two-step calibration procedure as described below.

Each benchmark run consists of three different phases, namely *calibration*, *preparation* and *execution*. In the calibration phase the robot moves the tip of the marker set to an arbitrary point on top of the test area which will be used as the *calibration point*. Afterwards, in the preparation phase, the robot must move the tip of the marker set by the predefined *calibration distance* (see Figure 14 and Figure 15) in a self-selected direction. The position reached after that motion is defined as the *starting point* of the path. Finally, in the execution phase, the robot follows the path until the end point is reached or the time runs out. After each phase, the robot notifies the CFH that it has completed the phase.

The robot-selected coordinate system is derived based on the calibration procedure in the following way:

1. The *x-axis* is defined as the vector from the calibration point to the starting point.

2. The *z-axis* points upwards from the test area.

3. The *y-axis* complements the right-handed coordinate system.

Figure 16 depicts two examples of such robot-selected coordinate systems.

Notice: this task is NOT executed with a feedback from any vision sensor from the team, but only testing a preplanned path and the online continuous path control ability of the robot!

5.3.5 Procedures and Rules

Each benchmark will be preceded by a safety-check similar to that described for the task benchmark procedures.

All teams are required to perform this functionality benchmark according to the steps mentioned below. During the competition, all teams are required to repeat it several times. On the last day, only a selected number of top teams will be allowed to perform it. Maximum time allowed for one functionality run is 4 minutes (60 seconds for calibration, 60 seconds for preparation and 120 seconds for execution). A run consists of (1) a calibration phase where the robot moves the tip of the marker set to the calibration point; (2) a preparation phase in which the tip of the marker set travels the calibration distance and (3) an execution phase where the robot follows the path.

Note: The tip of the marker set must maintain the height of the calibration point throughout a complete run. Only the position of the marker set's tip is considered, but not the orientation of the marker set.

Step 1 The robot is provided with the selection of the specific path and calibration distance in advance.

Step 2 The CFH tells the robot to start the calibration phase.

Step 3 The robot moves the tip of the marker set to its preferred calibration point. The robot reports to the CFH that it has finished the calibration phase.

Step 4 The referee manually adjusts a sheet of paper such that the calibration point on the sheet coincides with the projection of the marker set to the test area.

Step 5 The CFH tells the robot to start the preparation phase.

Step 6 The robot moves the tip of the marker set until the calibration distance has been traveled. The robot reports to the CFH that it has finished the preparation phase.

Step 7 The referee rotates the sheet of paper (with the calibration point acting as the pivot) such that the starting point coincides with the projection of the marker set to the test area.

Step 8 The CFH tells the robot to start the execution phase.

Step 9 The robot follows the path with the tip of the marker set and stops at the end point of the path. The robot reports the termination of the execution phase to the CFH.

5.3.6 Communication with CFH

For this task benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the calibration procedure when the *phase* field is equal to CALIBRATION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot reaches the calibration position, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to CALIBRATION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 4 The robot waits for a **BenchmarkState** message. It starts the preparation procedure when the *phase* field is equal to PREPARATION and the *state* field is equal to RUNNING.

Step 5 As soon as the robot reaches the preparation position, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to PREPARATION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 6 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 7 As soon as the robot reaches the goal position, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 8 The robot continues with Step 2.

Step 9 The functionality benchmark ends when the **BenchmarkState**'s *phase* field is equal to EXECUTION and the *state* field is equal to FINISHED.

The messages to be sent and to be received can be seen on the Github repository located at [3].

5.3.7 Acquisition of Benchmarking Data

General information on the aquisition of benchmarking data is described in Section 4. There also the **offline** part of the benchmarking data can be found.

Online Data No online benchmarking data has to be sent to the CFH during this task benchmark.

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/reference_pose[21]	geometry_msgs/PoseStamped	/base_link	–
/rockin/starting_pose[22]	geometry_msgs/PoseStamped	/base_link	When starting
/rockin/ending_pose[23]	geometry_msgs/PoseStamped	/base_link	When ending
/rockin/gripper_pose[24]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/arm_joints[25]	geometry_msgs/JointState	/base_link	10 Hz

5.3.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. Accuracy of following the given path with the tip of the marker
2. Number of completely executed path movements (maximum 5);
3. Execution time (if less than the maximum allowed for the benchmark).

The accuracy is evaluated as follows. Given the recorded marker path executed by the robot $r(l)$, the actual ground truth path $t(l)$ and l as a parameter in $[0:1]$ with

- $r(l) = (x_{r(l)}, y_{r(l)})$ the parametric representation of the robot path

[21] Pose of the gripper at the reference point.
[22] Pose of the gripper at the starting point.
[23] Pose of the gripper at the end of the trajectory.
[24] Pose of the gripper during the whole trajectory.
[25] Joints data

- $t(l) = (x_{t(l)}, y_{t(l)})$ the parametric representation of the target path

the accuracy is computed by

$$\frac{1}{N} * \sum\nolimits_{l \in L_{sampled}} d(r(l), t(l)) \quad (1)$$

where $L_{sampled}$ is a subset of L_{gt} and L_{gt} are the values of l to which corresponds a measure of the robot's path from the ground truth system, $N = |L_{sampled}|$ and $d()$ is the Euclidean distance. A more detailed analysis of this measure is available in the RoCKIn@Work Wiki [5].

Being this functionality benchmark focused on control, the scoring of teams is based on the accuracy with which the robot follows the given path. In case of ties the overall execution time will be taken into account.

6 RoCKIn@Work Award Categories

RoCKIn Competition 2015 awards will be given in the form of cups for the best teams, as specified below. Every team will also receive a plaquette with the RoCKIn logo and a certificate. Awards will be given to the best teams in RoCKIn@Work *task benchmarks*, *functional benchmarks* and overall.

6.1 Awards for Task Benchmarks

The team with the highest score in each of the three *task benchmarks* will be awarded a cup ("RoCKIn@Work Best-in-class Task Benchmark <*task benchmark* title>"). When a single team participates in a given *task benchmark*, the corresponding *task benchmark* award will only be given to that team if the Executive and Technical Committees consider the team performance of exceptional level.

6.2 Awards for Functionality Benchmarks

The two top teams in the score ranking for each of the three *functionality benchmarks* will be awarded a cup ("RoCKIn@Work Best-in-class Functionality Benchmark <*functionality benchmark* title>" and "RoCKIn@Work Second-Best-in-class Functionality Benchmark <*functionality benchmark* title>").

When less than three teams participate in a given *functionality benchmark*, only the "RoCKIn@Home Best-in-class Functionality Benchmark <*functionality benchmark* title>" award will be given to a team, and this will occur only if the Executive and Technical Committees consider that team's performance as excellent.

6.3 Competition Winners

Teams participating in RoCKIn@Work Competition 2015 will be ranked taking into account their overall rank in all the *task benchmarks*.

The overall ranking will be obtained by combining task benchmark rankings using the Social Welfare principle (see `http://en.wikipedia.org/wiki/Social_welfare_function`); the overall winning team of RoCKIn@Work Competition 2015 will be the top team in this combined ranking, and will receive the corresponding award cup ("Best Team RoCKIn@Work Competition 2015"). The second and third placed teams in the ranking will also receive award cups (respectively "2nd place RoCKIn@Work Competition 2015" and "3rd place RoCKIn@Work Competition 2015")). The three awards will be given only if more than 5 teams participate in the competition. Otherwise, only the best team will be awarded, except if it is the single team participating, in which case the Executive and Technical Committees must consider that team performance of

exceptional level so as for the team to be awarded. Only teams performing the total of the three tasks will be considered for the "Best Team RoCKIn@Work Competition 2015" award.

7 RoCKIn@Work Organization

7.1 RoCKIn@Work Management

The management structure of RoCKIn@Work has been divided into three committees, namely *Executive Committee*, *Technical Committee* and the *Organization Committee*. Pedro Lima (acting as the overall Coordinator of the Challenges Execution, within his role as RoCKIn Coordinator) is acting as **Supra-Chair**. The roles and responsibilities of those committees are described in the following paragraphs.

7.1.1 RoCKIn@Work Executive Committee

The Executive Committee (EC) is represented by the coordinators of each RoCKIn partner related to the respective activity area. The committee is mainly responsible for the overall coordination of RoCKIn@Work activities and especially for dissemination in the scientific community.

- Pedro Lima (Instituto Superior Técnico, Portugal)
- Daniele Nardi (Sapienza Università di Roma, Italy)
- Gerhard Kraetzschmar (Bonn-Rhein-Sieg University, Germany)
- Rainer Bischoff (KUKA Roboter GmbH, Germany)
- Matteo Matteucci (Politecnico di Milano, Italy)

7.1.2 RoCKIn@Work Technical Committee

The Technical Committee (TC) is responsible for the rules of the league. Each member of the committee is involved in maintaining and improving the current rule set and also in the adherence of these rules. Other responsibilities include the qualification of teams, handling general technical issues within the league, deciding about giving awards in case the number of participants is lower than the thresholds specified in Section 6, as well as resolving any conflicts inside the league during an ongoing competition. The members of the committee are further responsible for maintaining the RoCKIn@Work Infrastructure.

The Technical Committee currently consists of the following members:

- Alberto Pretto (Sapienza Università di Roma, Italy)
- Rhama Dwiputra (Bonn-Rhein-Sieg University, Germany)
- Tim Friedrich (KUKA Roboter GmbH, Germany)
- Matteo Matteucci (Politecnico di Milano, Italy)

This committee can also include members of the Executive Committee.

7.1.3 RoCKIn@Work Organizing Committee

The Organizing Committee (OC) is responsible for the actual implementation of the competition, i.e. providing everything what is required to perform the various tests. Specifically, this means providing setting up the test arena(s), providing any kind of objects (e.g. manipulation objects), scheduling the tests, assigning and instructing referees, recording and publishing (intermediate) competition results and any other kind of management and advertisement duties before, during and after the competition.

The Organizing Committee currently consists of the following members:

- **Chair:** Tim Friedrich (KUKA Roboter GmbH, Germany)
- Francesco Amigoni (Politecnico di Milano, Italy)
- Tiago Veiga (Instituto Superior Técnico, Portugal)
- Frederik Hegger (Bonn-Rhein-Sieg University, Germany)
- Graham Buchanan (InnoCentive EMEA, U.K.)

7.2 RoCKIn@Work Infrastructure

7.2.1 RoCKIn@Work Web Page

The official RoCKIn@Work website can be reached at

`http://rockinrobotchallenge.eu/work.php`

On this web page, teams can find introductory information about the league itself as well as relevant information about upcoming events, the most recent version of the rulebook, videos and pictures of past competitions and links to further resources like the official mailing list or wiki.

7.2.2 RoCKIn@Work Mailing List

The official RoCKIn@Work mailing list maintained by the league is as follows

`rockin-at-work@rockinrobotchallenge.eu`

Anyone can subscribe by using the following subscription page.

`http://rockinrobotchallenge.eu/mailman/listinfo/rockin-at-work`

Every subscriber is requested to register either with an email address which already encodes the real name or alternatively specify it in the provided field at the subscription page. In order to prevent the mailing list from spammers, this mailing list is moderated.

The mailing list will be used for any kind of official announcement, e.g. upcoming RoCKIn@Work competitions, rule changes, registration deadlines or infrastructure changes. Teams are also welcome to raise any kind of question regarding the league on this list.

7.3 RoCKIn@Work Competition Organization

7.3.1 Qualification and Registration

Participation in RoCKIn@Work requires successfully passing a qualification procedure. This procedure is to ensure a well-organized competition event and the safety of participants. Depending on constraints imposed by a particular site or the number of teams interested to participate, it may not be possible to admit all interested teams to the competition.

All teams that intend to participate at the competition have to perform the following steps (using the forms at the web site `www.rockincompetition.eu`):

1. Preregistration (deadline: 31 May 2015) – optional
2. Submission of qualification material (e.g. team description paper and video; deadline: 31 August 2015) – mandatory
3. Final registration (between 9 and 30 September 2015) – mandatory, and for qualified teams only

Preregistration A team must provide the following information during the preregistration process:

- Team name + Affiliation
- Team leader name
- Team leader email address
- Expected number of team members
- Whether the team plan to bring their own robot or not
- Middleware used for software development

This step can be considered as an *Intention of Participation* declaration and serves as planning basis for the Organizing Committee.

Qualification The qualification process serves a dual purpose: It should allow the Technical Committee to assess the safety of the robots a team intents to bring to a competition, and it should allow to rank teams according to a set of evaluation criteria in order to select the most promising teams for a competition, if not all interested teams can be permitted. The TC will select the qualified teams according to the qualification material provided by the teams.

The evaluation criteria will include:

- Team description paper
- Team web site
- Relevant scientific contribution/publications
- Professional quality of robot and software
- Novelty of approach
- Relevance to domestic service robotics
- Performance in other competitions
- Contribution to RoCKIn@Work league (e.g. by organization of events or provision and sharing of knowledge)

The Team Description Paper (TDP) is a central element of the qualification process and has to be provided by each team as part of the qualification process. The TDP should at least contain the following information in the author/title section of the paper:

- Name of the team (title)
- Team members (authors), including the team leader
- Link to the team web site
- Contact information

The body of the TDP should contain information on the following: focus of research/research interests:

- Description of the hardware, including an image of the robot(s)
- Description of the software, especially the functional and software architectures
- Main involved research areas in the team work
- Innovative technology (if any)
- Reusability of the system or parts thereof
- Applicability and relevance to industrial robotics

The team description paper should cover in detail the technical and scientific approach, while the team web site should be designed for a broader audience. Both the web site and the TDP have to be written in English.

The length of the team description paper is limited to 6 pages and has to be to submitted in the IEEE Conference Proceedings format[26].

Registration Only if a team has passed the qualification procedure successfully it is allowed to register officially for the competition and has to provide additional information e.g. the exact number of team members. Please be advised that this year, a team participating in TBM(s) for one of the Challenges (RoCKIn@Home or RoCKIn@Work) must participate in all FBMs for that Challenge. Further information about the registration procedure will be communicated through the mailing list of qualified teams. The number of people to register per team may be unlimited, but during the competition the organizers will provide space only for 6 persons to work at tables in the team area. During the final registration, each team has to designate one member as team leader. A change of the team leader must be communicated to the Organizing Committee.

7.3.2 Setup and Schedule

RoCKIn Competition 2015 will take place in the main science museum Pavilion of Knowledge and Portugal Pavilion of Lisboa, from 17-23 November 2015.

17–18 November will be the assembly days, during which the arenas, team areas, power, audiovisual equipment and other infrastructure will be put in place.

19–20 November will be setup days, that the teams can use to unpack their robots, calibrate the robot systems, and get information about the test bed, important objects and other relevant details. The site will be closed to the public.

There will be three competition days: 21, 22 and 23 November. During those days, the competitions will occur following the procedures and rules described in the subsections of this document with the same title. The site will be accessible to the public during the actual competitions.

The award and closing ceremony will take place in the evening of the last day, 23 November 2015.

Several satellite events, with the participation of industry and academia stakeholders, will take place during the five days of the main event. These include talks by members of RoCKIn's Advisory Board, and the assessment of the Competition by the members of RoCKIn's Experts Board.

Schedule: For the scheduling of particular stages, tests, and technical challenges of the competition the following applies:

- The exact schedule of task-functionality tests will be announced one week before the actual competition by the OC on both the website and the mailing list of qualified teams.
- In order to avoid excess of "traffic" inside the testbed, an additional schedule only for test slots will be established on site by the OC.
- A set of test slots will be assigned to each team between the official test slots, where a team has exclusive access to the testbed without any other team/robot inside the arena.

Setup: For the arrival, setup, and preparation of teams participating in the competition, the following procedures apply:

[26] `http://www.ieee.org/conferences_events/conferences/publishing/templates.html`

- A first draft of the rulebook will be made public on June 30th 2015.
- Revisions will be possible and updated in the online versions of the document, based on suggestions of all relevant stakeholders (including pre-registered and registered teams) until July 31th 2015.
- The final version of the rulebook will be made public, no later than eight weeks before the actual event, by the TC, including all the items referred as open in this document (e.g., some benchmarking and scoring items) and revisions resulting from the discussion referred in the previous item.
- The competition side will be divided into a competition arena and a team area.
- The competition arena consists of one or more testbeds (the arena) and is open for public.
- The arena must be kept clean and in a representable condition all the time.
- The team area is a dedicated area only for team members, no public access here.
- Each team will be assigned to a designated area with tables and chairs (based on the number of team members), with power sockets, a shelf internet connection and a reasonable area to park their robot and other equipment.

7.3.3 Competition Execution

- Referees will be determined by the OC out of the group of team leaders and TC members.
- The referees ensure the correct execution of a benchmark, are in charge of keeping the time and counting the scores, being always helped by a TC or OC member.
- In case of any dangerous situation the referees are allowed to immediately stop a benchmark and trigger the emergency stop functionality of the respective robot.
- The official language for all kind of communication within the league is English (e.g., team leader meetings, announcements, schedule).
- The order in which the teams have to perform a particular benchmark will be determined by a draw through the OC.
- The order will be announced on the day before the actual benchmark.
- No team members or other persons are allowed to be in the arena during an official benchmark (only if the rulebook explicitly allows/requires this).
- Regular team leader meetings (every day) will be organized and announced by the TC/OC during the competition in order to discuss open issues for upcoming benchmarks.

7.3.4 Measurements Recording

Several variables of interest will be recorded by the EC, TC and OC during the actual benchmarks of the teams during the competition, while performing their *task* and *functionality benchmarks*. Some of these will be performed by RoCKIn equipment, though requiring the installation of markers on the team robots. Other logging will require the teams to accommodate, in their software, modules that respond to solicitations from test bed-installed software. Details on these procedures will be provided closer to the competition dates, but the teams must be ready to commit to such requirements as one of the key requirements to be selected for the RoCKIn competitions. The logging and benchmarking activities will be under the responsibility of Giulio Fontana (Politecnico di Milano, Italy).

References

[1] J. Berghofer, R. Dwiputra, G. K. (editors), A. Ahmad, F. Amigoni, I. Awaad, J. Berghofer, R. Bischoff, A. Bonarini, R. Dwiputra, F. Hegger, N. Hochgeschwender, L. Iocchi, G. Kraetzschmar, P. Lima, M. Matteucci, D. Nardi, and S. Schneider, "RoCKIn@Work – Innovation in Industrial Mobile Manipulation – RoCKIn@Work in a Nutshell," KUKA Laboratories GmbH, Bonn-Rhein-Sieg University of Applied Sciences, Augsburg, Germany, and Sankt Augustin, Germany, Deliverable D-2.1.4 nutshell (public), The RoCKIn Project (FP7-ICT-601012 grant agreement number 601012), March 2014.

[2] A. Ahmad, I. Awaad, F. Amigoni, J. Berghofer, R. Bischoff, A. Bonarini, R. Dwiputra, G. Fontana, F. Hegger, N. Hochgeschwender, L. Iocchi, G. Kraetzschmar, P. Lima, M. Matteucci, D. Nardi, V. Schaffionati, and S. Schneider, "Description of Ground Truth System," Politecnico di Milano, Milan, Italy," Deliverable D-2.1.7 (public), The RoCKIn Project (FP7-ICT-601012 grant agreement number 601012), 2014.

[3] The RoCKIn Project. Central Factory Hub Messages. The RoCKIn Project (FP7-ICT-601012 grant agreement number 601012). [Online]. Available: https://github.com/rockin-robot-challenge/at_work_central_factory_hub/tree/rockin/rockin/msgs

[4] RoCKIn Project. Central Factory Hub Tools. The RoCKIn Project (FP7-ICT-601012 grant agreement number 601012). [Online]. Available: https://github.com/rockin-robot-challenge/at_work_central_factory_hub/tree/rockin/rockin/tools

[5] (2015) The RoCKIn Wiki. [Online]. Available: http://thewiki.rockinrobotchallenge.eu