TÉCNICO LISBOA

RoCKIn@Home

Referee, Scoring and Benchmarking Box User Manual

João Reis

RoCKIn Project
Grant Report

March 2014

Contents

1 Introduction

The Robot Competitions Kick Innovation in Cognitive Systems and Robotics (RoCKIn) project aims at fostering robotics research using competitions as an incentive. These competitions are organized as a set of benchmarks, designed to test each team's solution to a challenge, allowing for comparison and evaluation of solutions. During the benchmarks, a human referee enforces the rules. This referee must have a way to transmit his decisions to the robot, and receive some progress information. To achieve this in a practical way, an assistant referee is seated at a computer communicating verbally with the main referee. The assistant referee uses the Referee, Scoring and Benchmarking Box (RSBB) systems described by this manual. Besides basic starting and stopping functionality, the RSBB is also designed to receive scoring input and provide fine grained benchmark control for functionality benchmarks that require so.

The Referee, Scoring and Benchmarking Box was designed to support the following features:

Clock synchronization monitoring: The RSBB is not the best solution to synchronize robot clocks but is able to warn the referee if clock skew grows above a certain threshold. This warning is used both to alert the teams that the clock synchronization is at fault and to invalidate any received timestamps.

Benchmark starting and stopping: Benchmark can only start if robot clock skew is below 100 milliseconds. Stop can be issued manually by the referee, by the robot if it completed the benchmark or automatically by the RSBB if the time for the benchmark is over or if the robot does not declare that it is saving offline data.

Goal specification: for Functionality Benchmarks (FBMs), goals must be issued by the RSBB. Goal generation is done by the `rockin_benchmarking` module.

Devices communication: the RoCKIn@Home area includes automated home devices such as lights and window blinds. The RSBB provides an interface to control these devices, enabled only in certain benchmarks, so that the robot does not command the devices directly. The assistant referee can control the devices from his graphical interface.

Tablet communication: the RoCKIn@Home area includes a tablet device that can be used to communicate with the robot. Tablet communication passes through the RSBB and is enabled only for certain benchmarks.

Schedule: the full schedule of the competition is stored in the RSBB, allowing for automated progression with no setup time before each benchmark.

Scoring: the RSBB keeps track of the scoring items that can be evaluated during the competition (not using offline data). For Task Benchmarks (TBMs), the assistant referee marks scoring items in the interface. For FBMs, scoring is handled by the `rockin_benchmarking` module.

Online data: data produced by the robot during benchmarks falls in two categories: online and offline. Offline data is saved in a USB stick for latter analysis. Online data is transmitted to the RSBB. The RSBB displays and saves the data.

Logging: the RSBB saves a full log for each benchmark.

Referee interface: the RSBB includes a fully featured graphical interface to be used by the assistant referee.

Public interface: the RSBB includes an informative interface without controls to be displayed to the public.

Single client communication interface: the RSBB includes all features in a single communication interface. This way, participating teams only have to implement one communication mechanism.

State information: the RSBB continuously displays what state the benchmark is in.

Multiple simultaneous benchmarks: in some situations, it is useful to run task and functionality benchmarks simultaneously for different teams. The RSBB handles this transparently, supporting an arbitrary number of simultaneous benchmarks and connected referee interfaces. To support multiple simultaneous benchmarks, the RSBB uses the concept of zones. A zone is a group of scheduled benchmarks, not necessarily associated with a physical area. The RSBB handles all known zones independently, but a robot may only be participating in a benchmark in a single zone simultaneously (during competitions, at least two zones are expected to exist: Task Benchmarks and Functionality Benchmarks zones). An arbitrary number of referee Interfaces can be used to control different zones or to provide multiple views of the same zone.

2 Installation

The RoCKIn@Home Referee, Scoring and Benchmarking Box (RoAH RSBB) package is available from `http://github.com/joaocgreis/roah_rsbb`.

2.1 Dependencies

To install the RSBB, a few system dependencies must be satisfied. A C++11 compiler, CMake, Boost, Protobuf and OpenSSL must be installed. If you are using Ubuntu, install the dependencies with:

```
sudo apt-get install \
build-essential cmake libboost-all-dev \
libprotoc-dev protobuf-compiler libssl-dev
```

Furthermore, at least ROS Hydro is needed, instructions can be found at `http://wiki.ros.org/ROS/Installation/`.

This package also depends on the `roah_devices` (`http://github.com/joaocgreis/roah_devices`) and the `rockin_benchmarking` (`http://users.isr.tecnico.ulisboa.pt/~jreis/rockin/rockin_benchmarking_latest.tar.xz`) ROS packages, which must be available in the Catkin workspace.

2.2 Compilation

After `git clone` please do:

```
git submodule update --init --recursive
```

Compile as a normal ROS package with `catkin_make`, in your Catkin workspace.

2.3 Running

The following launch files are available:

- `roah_rsbb.launch` — Main launch file. This includes the RSBB core and a GUI client.
- `roah_rsbb_client.launch` — Launch only a GUI client. This assumes a RSBB core is already running in the same ROS system (`roah_rsbb.launch`).
- `roah_rsbb_public.launch` — Launch only a public interface. This assumes a RSBB core is already running in the same ROS system (`roah_rsbb.launch`).
- `roah_rsbb_dummy.launch` — Launch the full RSBB, for testing purposes. Like the main launch file, but also includes a dummy devices server, that simulates tesbed devices and dummy benchmarking modules (to generate fake goals for testing).

The main and the dummy launch files accept the following parameters:

- `rsbb_host` — IP address to use. Should be set to the `Bcast` address of the interface to use, as displayed by `ifconfig`. Defaults to 10.255.255.255, usually needs to be changed for testing.
- `rsbb_port` — UDP port to use. Defaults to 6666. Private communication ports start on this address.
- `smartif_host` — IP address of the SmartIF devices controller (not available in the dummy launch file).
- `benchmarks_file` — YAML file with benchmark definitions.
- `schedule_file` — YAML file with competition schedule.
- `passwords_file` — YAML file containing team passwords.
- `log_dir` — Directory to store the log files.
- `bell_ring_command` — Command to sound the bell ring.
- `timeout_ring_command` — Command to sound the timeout ring.

2.4 Network

The RSBB uses the `protobuf_comm` library for communication. All communication uses UDP, over two types of channels. A single public channel uses UDP broadcast to communicate with all robots at the same time. Multiple private channels are used to communicate with a single robot while it is running a benchmark. Private channels use UDP unicast.

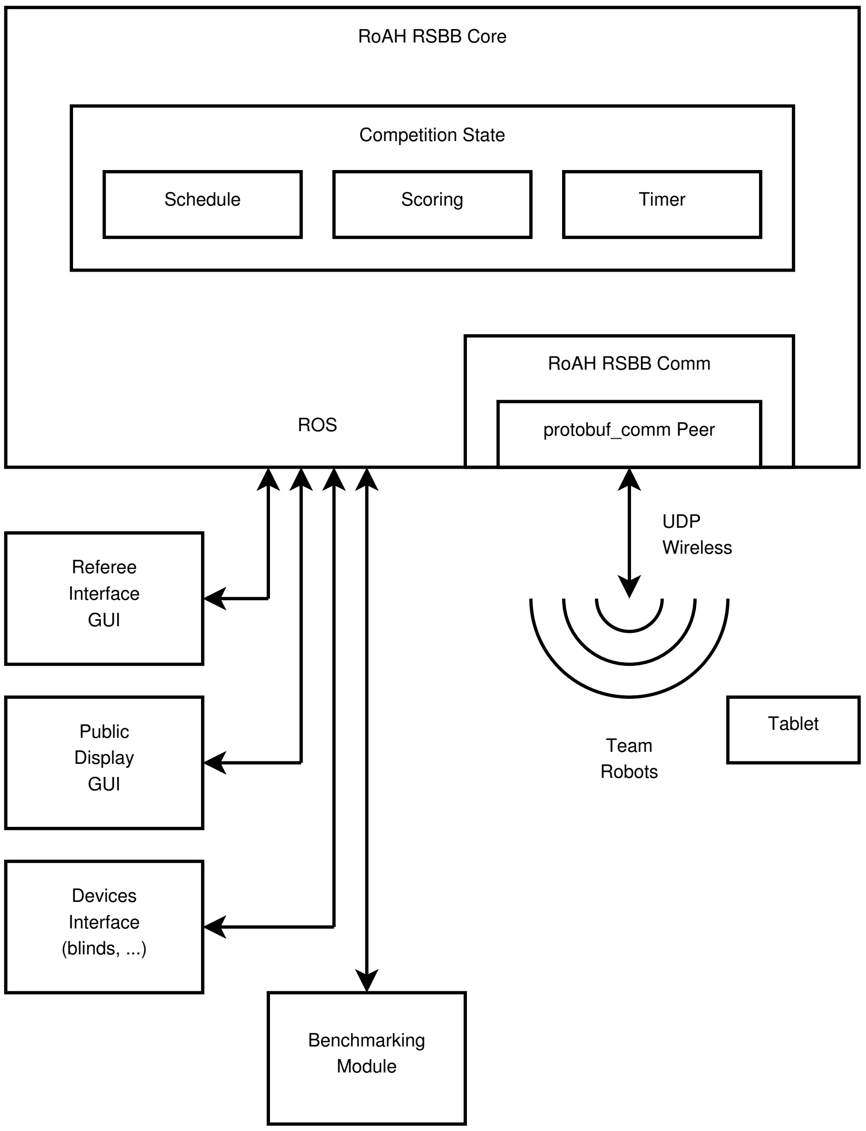
The robots and the RSBB must be on the same network, and the `rsbb_host` parameter must be the broadcast address of the network. The public channel is set up at the port specified in the `rsbb_port` parameter. The private channels use `rsbb_port`+1, `rsbb_port`+2 and so on without reusing.

2.5 Configuration Files

Under `config/`, three YAML configuration files can be found:

- `benchmarks.yaml` — Description of existing benchmarks, including scoring items.
- `passwords.yaml` — Team passwords. Should change on the beginning of every competition.
- `schedule.yaml` — Competition schedule, divided by zones.

3 Architecture



The main component is the Core. It controls all defined zones and handles communication with the robots. It functions as the brain of the system, and must be running for the RSBB.

The referee Interface, public display, and benchmarking module are part of the RSBB and connect to the Core using ROS. The home devices are controlled by an external package that also connects through ROS. The tablet runs an external Android application that connects to the public channel (Sec. 4.1), since it is a wireless device.

4 Communication

4.1 Public Channel

The public channel is used to transmit information that is relevant to all robots unencrypted. The RSBB transmits the `RoahRsbbBeacon` every second, containing:

- Information about which robots are active in benchmarks. Robots listed here should set up private channels for further information.
- The full state of the home devices.
- The full state of the tablet.

Active robots should also transmit their beacon, `RobotBeacon`, every second. This contains:

- Identification of the robot, to be listed as active.
- The current time at the time of transmission, to detect problems in clock synchronization.

When a robot sets up a private channel, it should stop transmitting on the public channel.

The tablet also transmits its state on the public channel, but robots should ignore it. Robots should only trust tablet information received in the `RoahRsbbBeacon`.

4.2 Private Channel

When a robot is active in a benchmark, a private channel is set up for communication. This channel is encrypted with the team password, to avoid interference from other sources. This method of protection is only sufficient to avoid honest mistakes in a practical way, not deliberate forgery of messages.

The RSBB transmits `benchmark_state` containing:

- The code of the specific benchmark to execute.
- The benchmark state.
- An acknowledgment of the last message received from the robot, to avoid eternal retransmission of data.

The robot transmits the `robot_state` containing:

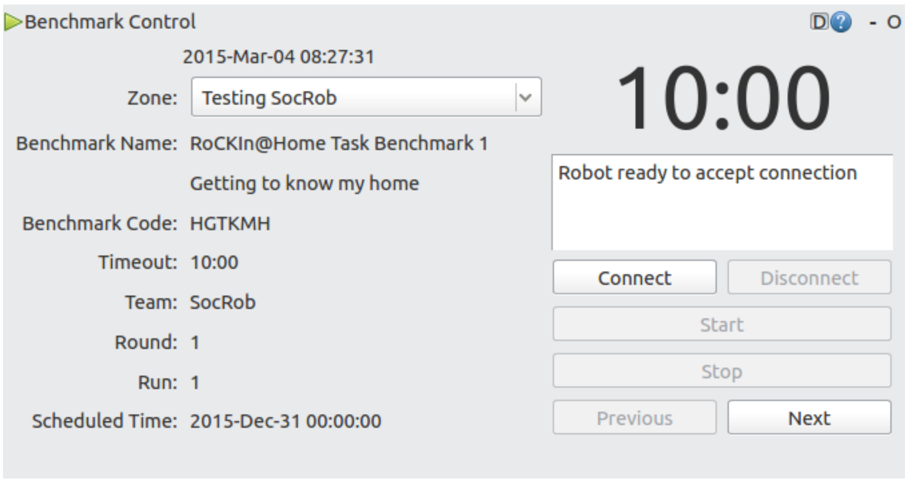
- The current time at the time of transmission, to detect problems in clock synchronization.

- Number of offline data messages saved. Can also be the size of saved data. This is considered to be a fail-safe feature to check whether the robot is recording the mandatory offline data.
- The robot state.
- Notifications issued by the robot. Used in TBM “Welcoming Visitors” and “Object Manipulation” FBM.
- Activation event. Used in TBM “Welcoming Visitors”.
- Visitor identification. Used in TBM “Welcoming Visitors”.
- Final command identified by the robot. Used in TBM “Catering for Granny Annie’s Comfort”.
- Home devices control commands. Used in TBM “Catering for Granny Annie’s Comfort”.
- Tablet map display command. Used in TBM “Catering for Granny Annie’s Comfort”.
- Object identification. Used in “Object Perception” FBM.

5 User Interface

The user interface is composed by a number of distinct components. The client launch file launches a single (ready to use) window with the components already laid out. However, they can also be launched individually and rearranged.

5.1 Benchmark Control



2015-Mar-04 08:27:31

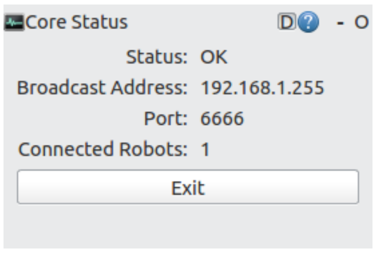
10:00

Zone:	Testing SocRob
Benchmark Name:	RoCKIn@Home Task Benchmark 1 Getting to know my home
Benchmark Code:	HGTKMH
Timeout:	10:00
Team:	SocRob
Round:	1
Run:	1
Scheduled Time:	2015-Dec-31 00:00:00

Robot ready to accept connection

This window allows for the assistant referee, to control the execution of the benchmark. It has a zone selector that will affect all other windows of the same client. The *next* and *previous* buttons can be used to select events, from the benchmarks scheduled for this zone. The *connect* button establishes a connection with the robot and prepares the benchmark. The *start* and *stop* buttons start and stop the benchmark. The *disconnect* button ends the benchmark.

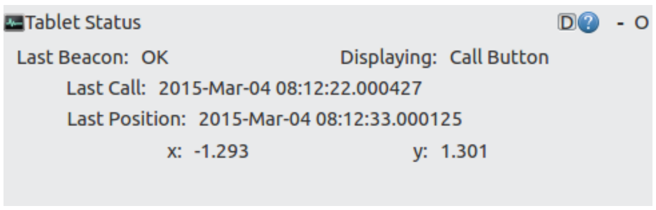
5.2 Core Status



Status:	OK
Broadcast Address:	192.168.1.255
Port:	6666
Connected Robots:	1

The core status window displays basic information about the RSBB. It also has an *exit* button that terminates the RSBB or only the current client, depending on which launch file was used.

5.3 Tablet Status


Tablet Status

Last Beacon: OK Displaying: Call Button

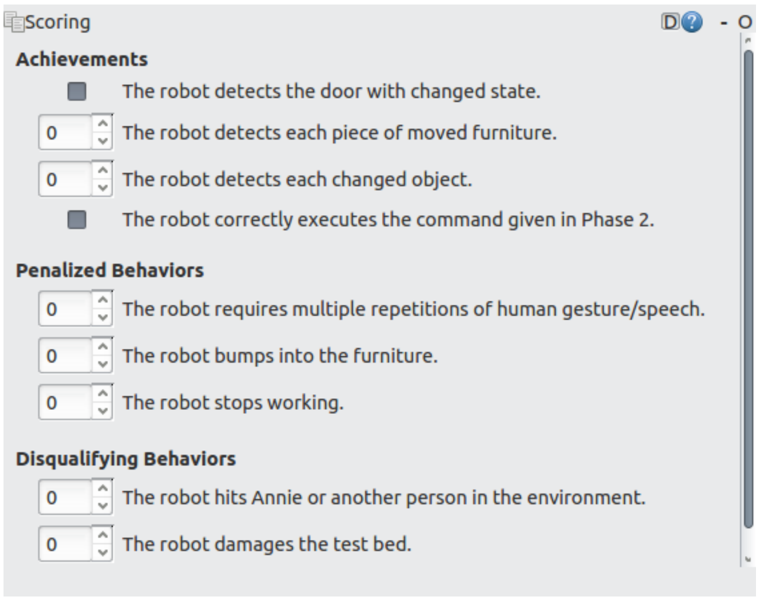
Last Call: 2015-Mar-04 08:12:22.000427

Last Position: 2015-Mar-04 08:12:33.000125

x: -1.293 y: 1.301

The tablet status window displays the last information received from the tablet.

5.4 Scoring


Scoring

Achievements

▢	The robot detects the door with changed state.
0	The robot detects each piece of moved furniture.
0	The robot detects each changed object.
▢	The robot correctly executes the command given in Phase 2.

Penalized Behaviors

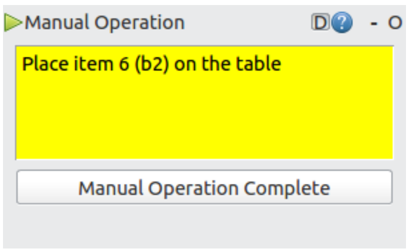
0	The robot requires multiple repetitions of human gesture/speech.
0	The robot bumps into the furniture.
0	The robot stops working.

Disqualifying Behaviors

0	The robot hits Annie or another person in the environment.
0	The robot damages the test bed.

The scoring window is used to input benchmark scoring, in benchmarks that support it.

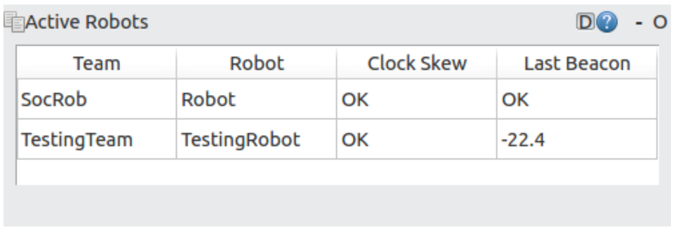
5.5 Manual Operation


Manual Operation

Place item 6 (b2) on the table

The manual operation window is used in FBMs to ask the referee to take action of some sort. After the action is complete, the *manual operation complete* button should be clicked.

5.6 Active Robots



Team	Robot	Clock Skew	Last Beacon
SocRob	Robot	OK	OK
TestingTeam	TestingRobot	OK	-22.4

This window displays a list of active robots, detected by the RSBB.

5.7 Online Data

This is a simple text scroll window, where the online data received from the robot is displayed during benchmarks.

5.8 Log Display

This is a simple text scroll window, where the logs recorded by the RSBB are displayed (in real time).

6 Client Software

Two client libraries are available to ease communication with the RSBB.

6.1 RoAH RSBB Comm

This software package can be downloaded from `http://github.com/joaocgreis/roah_rsbb_comm`. This package does not use and does not depend on ROS. Hence, it can be used by teams not using ROS. It has an header for teams using C++ but, on the limit, the messages defined within can be used on any system capable of usin Protobuf and UDP broadcast communication. It contains:

- The `protobuf_comm` library repacked for ease of deployment.
- The proto messages used by the RSBB.
- An header-only C++ library to ease the direct use of `protobuf_comm`, abstracting public and private channels.
- A very simple test executable, `capture_comm`.

This is used by both the *RoAH RSBB* and the *RoAH RSBB Comm ROS* ROS packages.

6.2 RoAH RSBB Comm ROS

This ROS package contains a full implementation of the communication protocol as a ROS node. This can be readily deployed by competing teams, having only to write glue code to activate the correct code for each benchmark. This is available and fully documented at `http://github.com/joaocgreis/roah_rsbb_comm_ros`. It includes:

- The `comm` node, to be used in ROS systems.
- An example dummy robot, that shows how to use the `comm` node.

7 Conclusion

The software presented in this manual is fully functional and ready to be used. It presents some complexity for anyone trying to understand it in depth, but should be simple enough for untrained users during competitions. Stability and quality have been constant concerns during development. Still, as with all software creations, it is not possible to guarantee perfection. Thus, care has been taken to ensure log files are saved constantly and in a well tested format. Many error messages are spread throughout the code base, to give the user a good chance of identifying errors in case anything goes wrong. Well tested and proven technologies have been used in all possible situations.