ETSA03 RoboTalk

v1.4

- A Robocode Communication Protocol for the LU Rumble

2020-03-23

Developed by Markus Borg

Approved by the ETSA03 Regulatory Body

Revision history

Name	Date	Description	Version
Markus Borg	2018-03-23	Initial version of the communication protocol.	1.0
Markus Borg	2018-03-29	Renamed to RoboTalk. Added friendPos.	1.1
Markus Borg	2018-04-29	Added team color. Added arbitrary number of friendPos. Added name to friendPos and enemyPos. Specified newlines as ‘\n’.	1.2
Markus Borg	2018-05-02	Added two commands: bullet and enemyDetails.	1.3
Markus Borg	2019-03-25	Removed 2018 from the document.	1.4
Markus Borg	2020-03-23	Updated course code and year.	1.5

Table of contents

1. Fundamentals of Robocode team communication

Robots that extend TeamRobot can communicate with team members, as specified in the Robocode API: http://robocode.sourceforge.net/docs/robocode/robocode/TeamRobot.html
Serializable objects can be sent between robots, e.g., point objects containing coordinates, color objects to set uniform team colors, and message strings with orders from the leader. The following three methods represent the backbone of robot communication.

Broadcasting message to all team members:
```
public void broadcastMessage(Serializable message)
```

Sending messages to one specific team member:
```
TeamRobot.sendMessage(String name, Serializable message)
```

When a robot receives a message, the following method is called:
```
onMessageReceived(MessageEvent e)
```

The MessageEvent always contains the name of the sending robot and a specific message. As stated above, the message can be instances of various classes. Please refer to the Robocode Sample Team to see an example implementation.

The only type of communication that must be implemented by the robots entering LU Rumble is **team color - this is mandated by the Regulatory Body** (please refer to Section 3). During the LU Rumble, all robots of the same team must share the same color scheme. With 10 robots battling at the same time, it is critical to help spectators understand the action on the battlefield. This introduces one of the following regulatory requirements to all robots:

- For normal robots and droids:
    - The robot shall set its color scheme upon request by a leader bot according to ETSA03 RoboTalk v1.2.
- For leader bots:
    - The robot shall set its color scheme at the start of the battle, and request the other team robots to follow suit according to ETSA03 RoboTalk v1.2.

ETSA03 RoboTalk also specifies a string-based communication protocol. Teams are free to implement as much of the communication protocol as they find useful. While it is completely optional, it might be expected by potential customers on Robot Market. Section 4 specifies the latest version of the string-based communication protocol. Please refer to the classes MessageReader and Message Writer of BasicMeleeBot for an implementation suggestion.

2. Evolution of ETSA03 RoboTalk

To allow communication between robots developed by different suppliers, the ETSA03 Regulatory Body has standardized a communication protocol: ETSA03 RoboTalk. All messages (except the team color) are to be transmitted as plain text strings, with one command per line. ETSA03 RoboTalk will be version controlled, and groups that decide to implement communication should carefully specify in their requirements specifications which version of the protocol is implemented - and which commands if a subset is implementation.

ETSA03 RoboTalk will evolve during the course, i.e., the Regulatory Body will release new versions. All groups are welcome to provide feedback and requests to the standardization work done by the Regulatory Body. To do so, the project manager should submit a “Change Request” to the Regulatory Body. The Change Request should carefully describe what changes are proposed, and motivate why this request should be considered.

3. ETSA03 RoboTalk - Team colors

All robots shall implement team colors to prepare for the LU Rumble. All robots must include the class RobotColors in their object-oriented design. The following section shows an example of how team colors can be implemented.

3.1 Sending a color scheme from the leader bot to the team

The leader bot should set the team color before the battle. The Regulatory Body recommends robot developers to do first in the `run()` method. Also, remember to set the color schema for the leader bot itself. Teams developing a leader bot shall set the color scheme requested by their customer(s).

```
// Set team colors in a RobotColors object
RobotColors c = new RobotColors();
c.bodyColor = Color.darkGray;
c.gunColor = Color.blue;
c.radarColor = Color.blue;
c.scanColor = Color.darkGray;
c.bulletColor = Color.darkGray;

try {
// Send RobotColors object to the entire team
  broadcastMessage(c);
} catch (IOException ignored) {
  System.out.println("Could not broadcast team colors.");
}
```

3.1 Receiving a color scheme from the leader bot

Normal bots and droids shall add the following code to the method `onMessageReceived(MessageEvent e)`, or implement similar behavior.

```
if (e.getMessage() instanceof RobotColors) {
  RobotColors c = (RobotColors) e.getMessage();
  setBodyColor(c.bodyColor);
  setGunColor(c.gunColor);
  setRadarColor(c.radarColor);
  setScanColor(c.scanColor);
  setBulletColor(c.bulletColor);
}
```

4. ETSA03 RoboTalk - String-based commands

A message string should have the following overall structure, including the new lines. Note that the newline control character to use in RoboTalk is '\n'.

```
[0-1] leadership;[followMe|leadMe]
[0-1] teamMode;[offensive|defensive]
[0-1] myPos;x;y
[0-*] friendPos;name;x;y
[0-*] enemyPos;name;x;y
[0-*] enemyDetails;name;x;y;velocity;energy;heading;gunHeading
[0-1] bullet;x;y;absBearing;bulletPower
[0-1] targetEnemy;name
[0-1] targetPos;x;y
[0-1] moveTo;x;y
```

Example message:

```
teamMode;offensive
myPos;100;100
enemyPos;bot1;50;50
enemyPos;bot2;200;150
targetPos;50;50
```

The commands shall have the following meaning:

`leadership`
: Tells the receiving robot(s) that the sender takes on leadership (`followMe`) or reports as a follower (`leadMe`).

`teamMode`
: Tells the receiving robot(s) that the team shall enter offensive mode (`offensive`) or defensive mode (`defensive`).

`myPos`
: Tells the receiving robot(s) its current position on the 2D battlefield.

`friendPos`
: Tells the receiving robot(s) the position of an identified robot in the same team.

`enemyPos`
: Tells the receiving robot(s) the position of an identified enemy robot.

`enemyDetails`
: Tells the receiving robot(s) detailed information about an enemy robot.

`bullet`

Tells the receiving robot(s) that a bullet has been fired and details its trajectory.

`targetPos`

Tells the receiving robot(s) to fire at a specific position on the 2D battlefield.

`targetEnemy`

Tells the receiving robot(s) to target a specific enemy robot.

`moveTo`

Tells the receiving robot(s) to move to a specific position on the 2D battlefield.