Create Documentation

Tom Williams

September 5, 2014

Contents

1 Basics **2**
1.1 Accessing into the robot 2
1.2 Building your first assignment 2
1.3 Running your first assignment 2
1.4 Building the Create code 2
1.5 Running the Create demo 3
1.6 Starting the Create 3
1.6.1 Starting the create in Safe Mode 3
1.6.2 Staring the create quietly 3
1.7 Stopping the Create 3

2 Safety **3**
2.1 Get both bumper readings 3
2.2 Get one bumper reading 3

3 Motion **3**
3.1 Stop the create 3
3.2 Spin the create 3
3.3 get / set translational or rotational velocities 4
3.4 get / set both velocities 4
3.5 get default velocities 4

4 Sounds **4**
4.1 Playing a Sound 4

5 Laser **4**
5.1 Check if the LRF is ready to operate 4
5.2 Get current scan data 4
5.3 Get/Set "critical distance" information used for obstacle / safety detection. 5
5.4 get information on safety to the right, in front, to the left, behind 5
5.5 Check if there's an obstacle in front of the robot 5

1 Basics

1.1 Accessing into the robot

1. Connect to the HRILAB network. A labmember can log you in or give you the password.

2. `ssh Team<your team number>@192.168.0.175`

3. Enter your password. Tom Williams can provide you your initial team password.

In your home folder you will find (1) a basic create demo that shows how to connect to and run commands on the robot, and (2) a script that compiles and runs that demo. You will want to adapt these two files to fit your needs.

1.2 Building your first assignment

As seen in line 1 of build.sh:

```
javac -cp /middleearth/Create.jar:/usr/share/java/RXTXComm.jar:/usr/lib/jni/*:.
<YOURFILENAME>.java
```

1.3 Running your first assignment

As seen in line 2 of build.sh:

```
java -cp /middleearth/Create.jar:/usr/share/java/RXTXComm.jar:.
-Djava.library.path=/usr/lib/jni/ <YOURCLASSNAME>
```

1.4 Building the Create code

You shouldn't need to rebuild this code, but here's how to do it. `ant clean compile`

1.5 Running the Create demo

```
ant demo
```

1.6 Starting the Create

```
Create create = new Create("/dev/createport","/dev/laserport");
```

1.6.1 Starting the create in Safe Mode

```
Create create = new Create("/dev/ttyACM0","/dev/ttyAMA0").safe();
```

1.6.2 Staring the create quietly

```
Create create = new Create("/dev/ttyACM0","/dev/ttyAMA0"").stfu();
```

1.7 Stopping the Create

```
create.localshutdown();
```

2 Safety

2.1 Get both bumper readings

```
create.getBumperReadings();
```

2.2 Get one bumper reading

```
create.getBumper(0);
```

3 Motion

3.1 Stop the create

```
create.stop();
```

This is functionally equivalent to calling setVels(0.0,0.0);

3.2 Spin the create

```
create.stationarySpin(5.0)
```

3.3 get / set translational or rotational velocities

Translational velocities are in mm/s. Thus setting translational velocity to 250 for 4 seconds should bring the robot forward one meter. To rotate the robot 360*, Set rotational velocity to ~220 for 16 seconds.
`create.getTV(); create.getRV(); create.setTV(2.0); create.setRV(2.0);`

3.4 get / set both velocities

`create.getVels(); create.setVels(2.0,2.0);`

3.5 get default velocities

`create.getDefaultVels();`

4 Sounds

4.1 Playing a Sound

`create.playSound("funsound.wav");`

5 Laser

5.1 Check if the LRF is ready to operate

`create.lrf.isReady();` Even once the LRF starts up, it is not ready to use for a few seconds; for the first few seconds, it returns 0 for every range. You may find it useful to wait until some mid-scan reading is nonzero.

5.2 Get current scan data

`create.lrf.getPolarScanData();` This returns a $\text{polar}_{\text{laserscan}}$. The most important field of the $\text{polar}_{\text{laserscan}}$ is the ranges[] field, an array of floats that updates on the order of 20Hz. These datapoints are the distances of the nearest object from the robot at each ~1/3 of a degree for 240 degrees centered around the front of the robot. These measurements are taken 2-3 inches off the ground, so if, for example, there is a table in front of the robot, it will only be able to detect the table's legs.

The public fields of interest available in a $\text{polar}_{\text{laserscan}}$ are:

- public long timestamp;

- public int scanID;
- public float startAngle;
- public float angularResolution;
- public float[] ranges;
- public short numRanges;
- public float maxRange;
- public float offset_{x};
- public float offset_{y};
- public float offset_{t};

You can also make a copy of a scan with the $\text{polar}_{\text{laserscan's}}$ copy() method.

5.3 Get/Set "critical distance" information used for obstacle / safety detection.

```
create.lrf.getCritDist();
create.lrf.setCritDist(1.5);
```

5.4 get information on safety to the right, in front, to the left, behind

```
create.lrf.getOpenSpaces();
```

5.5 Check if there's an obstacle in front of the robot

```
create.checkObstacle();
```