nao_interfaces Documentation

Kenji Brameld

Jul 24, 2021

CONTENTS

This is a ROS2 interface package for the Softbank NAO robot. Custom message types specific to the NAO robot are defined in this package, and are explained in these docs.

The project is hosted on Github.

Sensor Msgs page explains all msgs that can be used to read sensor information, such as joint angles, sonar readings, etc.

Command Msgs page explains all msgs that can be used to command the robot, to move joints, light up leds, etc.

Joints page explains in detail, how to read sensor information and write commands for the NAO's joints, along with some examples.

Leds page describes in detail, the name and position of each led on the robot, with examples on how to set the intensity and color for them.

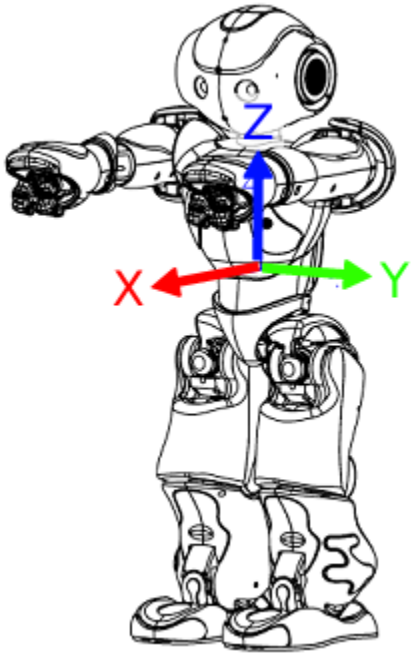
CHAPTER

ONE

SENSOR MSGS

The package **nao_sensor_msgs** defines msgs used to store sensory information specific to the NAO robot.

1.1 Accelerometer

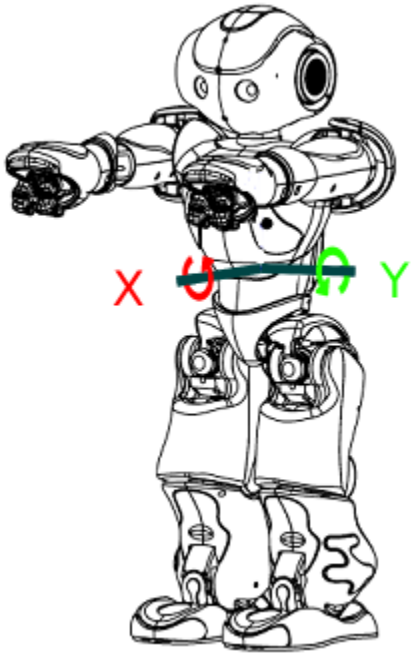


Units are in m/s ^{2}

Note that gravity is measured by an accelerometer. When standing, the robot will measure 9.8 m/s ^{2} POSITIVE in the z-direction, since its receiving an acceleration upwards from the floor when compared to a freefall state.

```
float32 x  # m/s/s
float32 y  # m/s/s
float32 z  # m/s/s
```

1.2 Angle



Units are in rad.

An absolute angle of the torso. Both x and y are zero when upright.

```
float32 x  # rad
float32 y  # rad
```

1.3 Battery

```
float32 charge  # 0% - 100%
bool charging  # True if robot is getting charged
float32 current  # In Amperes, negative for discharge, positive for charge
float32 temperature  # Celcius (°C)
```

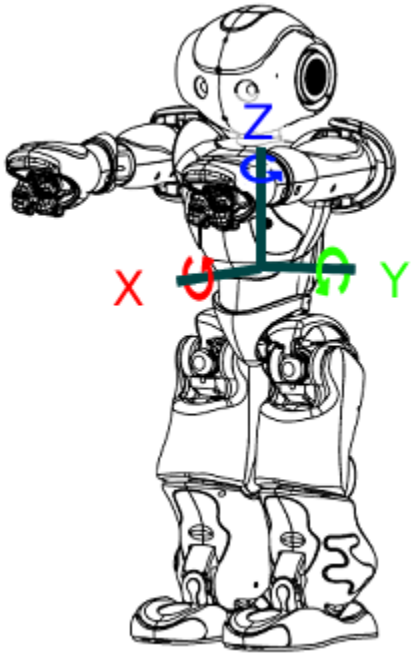
1.4 Buttons

```
bool chest  # true if being pressed
bool l_foot_bumper_left  # true if being pressed
bool l_foot_bumper_right  # true if being pressed
bool r_foot_bumper_left  # true if being pressed
bool r_foot_bumper_right  # true if being pressed
```

1.5 FSR

```
float32 l_foot_front_left  # kg
float32 l_foot_front_right  # kg
float32 l_foot_back_left  # kg
float32 l_foot_back_right  # kg
float32 r_foot_front_left  # kg
float32 r_foot_front_right  # kg
float32 r_foot_back_left  # kg
float32 r_foot_back_right  # kg
```

1.6 Gyroscope



Units are in rad/s.

```
float32 x  # rad/s
float32 y  # rad/s
float32 z  # rad/s
```

1.7 Joints

1.7.1 JointCurrents

Electrical current, reported from the current sensors in each motor joint of the NAO.

```
float32[25] currents  # Amperes (A), in order of JointIndexes.msg
```

1.7.2 JointPositions

Joint positions of each motor joint.

```
# An array of joint positions, corresponding to their indexes in the JointIndexes.msg.

float32[25] positions # radians
```

1.7.3 JointStatuses

Temperature status enums, computed accordingly to the temperature limitation to protect the motors.

```
int32 STATUS_NORMAL=0           # normal
int32 STATUS_HOT=1              # high, start to reduce stiffness
int32 STATUS_VERY_HOT=2         # very hot, stiffness reduced over 30%
int32 STATUS_CRITICALLY_HOT=3   # critically hot, stiffness is set to 0

int32[25] statuses  # Status codes, in order of JointIndexes.msg
```

1.7.4 JointStiffnesses

Joint stiffnesses in each motor joint.

```
# An array of joint stiffnesses, corresponding to their indexes in the JointIndexes.msg.

float32[25] stiffnesses  # 0.0 - 1.0
```

1.7.5 JointTemperatures

Temperature reported for each motor joint in the NAO.

Tip: **The motor temperature is a simulated one**, using electric current value of the motor. The motor board implements a temperature limitation to protect the motor. The temperature limitation depends on robot version.

```
float32[25] temperatures  # Celcius (°C), in order of JointIndexes.msg
```

1.8 RobotConfig

```
string body_id  # eg."P0000073A07S94700012"
string body_version  # eg. "6.0.0"
string head_id  # eg. "P0000074A05S93M00061"
string head_version  # eg. "6.0.0"
```

1.9 Sonar

Sonar distance measurements.

```
float32 left  # m
float32 right  # m
```

1.10 Touch

```
bool head_front  # true if being touched
bool head_middle  # true if being touched
bool head_rear  # true if being touched
```

CHAPTER

TWO

COMMAND MSGS

The package **nao_command_msgs** defines msgs used to send commands to the underlying NAO hardware.

2.1 Joints

2.1.1 JointPositions

By being able to specify the indexes, we can have different nodes sending partial joint position commands.

```
# Message to specify positions for the NAO's motor joints.
#
# Each joint is uniquely identified by its index (See JointIndexes.msg)
#
# The two arrays in this message should have the same size, where the first item
# in indexes, corresponds to the first item in positions, etc.

uint8[] indexes  # See JointIndexes.msg (eg. JointIndexes::HEADYAW)
float32[] positions # radians
```

2.1.2 JointStiffnesses

By being able to specify the indexes, we can have different nodes sending partial joint stiffness commands.

```
# Message to specify stiffnesses for the NAO's motor joints.
#
# Each joint is uniquely identified by its index (See JointIndexes.msg)
#
# The two arrays in this message should have the same size, where the first item
# in indexes, corresponds to the first item in stiffnesses, etc.

uint8[] indexes  # See JointIndexes.msg (eg. JointIndexes::HEADYAW)
float32[] stiffnesses  # 0.0 - 1.0
```

2.2 RGB Leds

Msgs that use std_msgs/ColorRGBA to specify colors for the RGB LEDs.

Note: Expect ranges for R, G and B are 0.0 - 1.0. The alpha value (A) is ignored.

2.2.1 ChestLed

A single RGB led in the chest.

```
std_msgs/ColorRGBA color  # r, g, b should be 0.0 - 1.0. a is ignored
```

2.2.2 LeftEyeLeds

See *Left Eye Leds* to see which indexes correspond to which led.

```
std_msgs/ColorRGBA[8] colors
```

2.2.3 LeftFootLed

A single RGB led in the left foot.

```
std_msgs/ColorRGBA color
```

2.2.4 RightEyeLeds

See *Right Eye Leds* to see which indexes correspond to which led.

```
std_msgs/ColorRGBA[8] colors
```

2.2.5 RightFootLed

A single RGB led in the right foot.

```
std_msgs/ColorRGBA color
```

2.3 Blue Leds

Msgs that specify intensity of the blue leds.

2.3.1 HeadLeds

See *Head Leds* to see which indexes correspond to which led.

```
float32[12] intensities  # 0.0 - 1.0
```

2.3.2 LeftEarLeds

See *Left Ear Leds* to see which indexes correspond to which led.

```
float32[10] intensities  # 0.0 - 1.0
```

2.3.3 RightEarLeds

See *Right Ear Leds* to see which indexes correspond to which led.

```
float32[10] intensities  # 0.0 - 1.0
```

2.4 SonarUsage

Command to tell Lola whether to enable/disable the sonar.

```
bool left  # Set to true, to use left sonar
bool right  # Set to true, to use right sonar
```

CHAPTER

THREE

JOINTS

3.1 Tutorials

3.1.1 Reading a joint position from sensor

To read a joint position for a specific joint:

```
// Assume joints is of type nao_sensor_msgs::msg::JointPositions
joints.positions[nao_sensor_msgs::msg::JointIndexes::HEADYAW]
```

nao_sensor_msgs::msg::JointPositions contains an array of joint positions for all joints of the NAO. The indexes for each joint are stored as constants in *nao_sensor_msgs::msg::JointIndexes*.

3.1.2 Writing a joint position command

To populate the message,

- Append index of the joint to the *indexes* vector
- Append position of the joint to the *positions* vector

For example:

```
// Assume joints is of type nao_sensor_msgs::msg::JointPositions
nao_command_msgs::msg::JointPositions command;

command.indexes.push_back(nao_command_msgs::msg::JointIndexes::HEADYAW);
command.positions.push_back(0.2);
```

Note: Joint indexes are stored as constants in *nao_command_msgs::msg::JointIndexes*.

3.1.3 Iterating over all joints

To iterate over all joints, use a for loop that iterates `NUM_JOINTS` times to iterate over all joints, as following:

```
// Create JointStiffness command with stiffness 1.0 for all joints
nao_command_msgs::msg::JointStiffnesses command;

for (unsigned i = 0; i < nao_command_msgs::msg::JointIndexes::NUMJOINTS; ++i)
{
    command.indexes.push_back(i);
    command.positions.push_back(1.0);
}
```

3.2 Joint Indexes

A msg file that defines the indexes of joints as constants. Used in indexes for *Sensor Joint Msgs* and *Command Joint Msgs*.

This msg does not have any fields, and doesn't serve a purpose in transmitting any information on topics.

The following list are the joint indexes copied from `JointIndexes.msg`:

```
uint8 HEADYAW=0
uint8 HEADPITCH=1
uint8 LSHOULDERPITCH=2
uint8 LSHOULDERROLL=3
uint8 LELBOWYAW=4
uint8 LELBOWROLL=5
uint8 LWRISTYAW=6
uint8 LHIPYAWPITCH=7
uint8 LHIPROLL=8
uint8 LHIPPITCH=9
uint8 LKNEEPITCH=10
uint8 LANKLEPITCH=11
uint8 LANKLEROLL=12
uint8 RHIPROLL=13
uint8 RHIPPITCH=14
uint8 RKNEEPITCH=15
uint8 RANKLEPITCH=16
uint8 RANKLEROLL=17
uint8 RSHOULDERPITCH=18
uint8 RSHOULDERROLL=19
uint8 RELBOWYAW=20
uint8 RELBOWROLL=21
uint8 RWRISTYAW=22
uint8 LHAND=23
uint8 RHAND=24
uint8 NUMJOINTS=25
```

Note: Currently, there are two identical copies of *Joint Indexes* msg file, in the *Sensor Msgs* and *Command Msgs* package. This is because we haven't found a nice way to avoid duplication. Possibly, this shouldn't even be a msg file

but should be just a plain hpp file.

CHAPTER

FOUR

LEDS

4.1 Tutorials

4.1.1 Set blue led intensity

To set the intensity of *Blue Leds*:

```
// Set led L0 in left ear to intensity 1.0
nao_command_msgs::msg::LeftEarLeds left_ear_leds;
left_ear_leds.intensities[nao_command_msgs::msg::LeftEarLeds::L0] = 1.0;
```

4.1.2 Set color of RGB led

```
// Set red led in L0 of left eye to intensity 1.0
nao_command_msgs::msg::LeftEyeLeds eye_leds;
eye_leds.colors[nao_command_msgs::msg::LeftEyeLeds::L0].r = 1.0;
```

4.1.3 Set color across all leds

To populate all LEDs to be a certain color, you can iterate over NUM_LEDS, in the corresponding msgs.

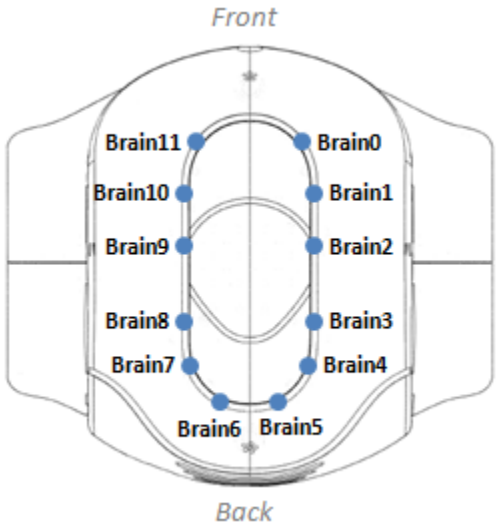
```
// Set all leds to yellow in right eye
nao_command_msgs::msg::RightEyeLeds right_eye_leds;

std_msgs/ColorRGBA yellow;
yellow.r = 1.0;
yellow.g = 1.0;

for (unsigned i = 0; i < nao_command_msgs::msg::RightEyeLeds::NUM_LEDS; ++i)
{
  right_eye_leds.colors[i] = yellow;
}
```

4.2 LED Indexes

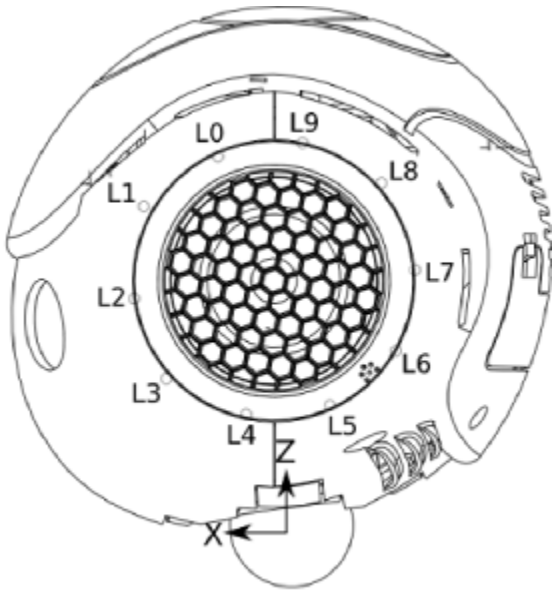
4.2.1 Head Leds



The following list is from `HeadLeds.msg`:

```
int32 B0=0
int32 B1=1
int32 B2=2
int32 B3=3
int32 B4=4
int32 B5=5
int32 B6=6
int32 B7=7
int32 B8=8
int32 B9=9
int32 B10=10
int32 B11=11
int32 NUM_LEDS=12
```

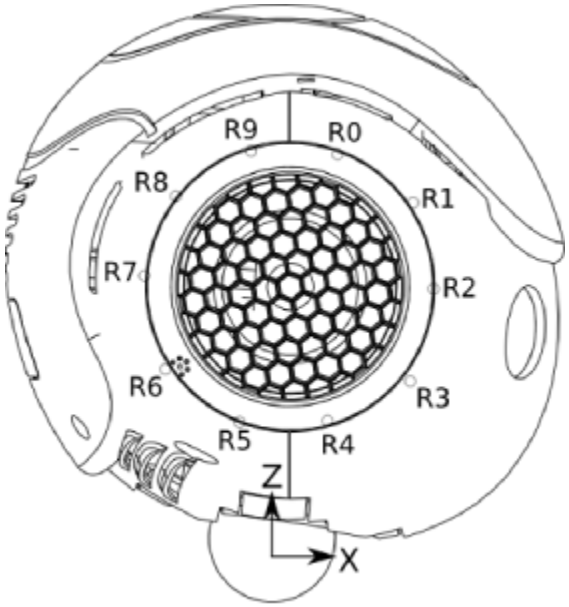
4.2.2 Left Ear Leds



The following list is from `LeftEarLeds.msg`:

```
int32 L0=0
int32 L1=1
int32 L2=2
int32 L3=3
int32 L4=4
int32 L5=5
int32 L6=6
int32 L7=7
int32 L8=8
int32 L9=9
int32 NUM_LEDS=10
```

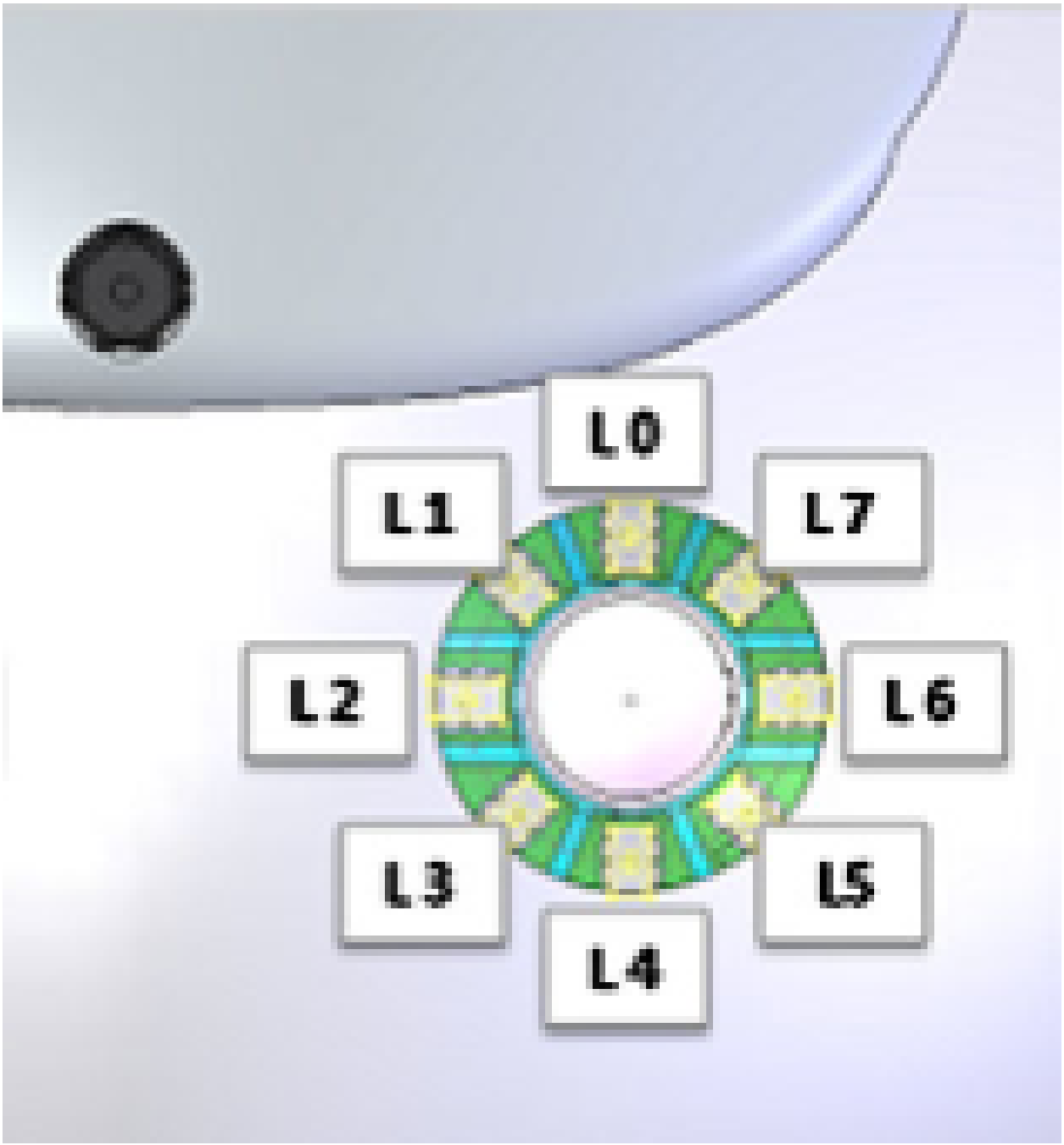
4.2.3 Right Ear Leds



The following list is from `RightEarLeds.msg`:

```
int32 R0=0
int32 R1=1
int32 R2=2
int32 R3=3
int32 R4=4
int32 R5=5
int32 R6=6
int32 R7=7
int32 R8=8
int32 R9=9
int32 NUM_LEDS=10
```

4.2.4 Left Eye Leds



The following list is from `LeftEyeLeds.msg`:

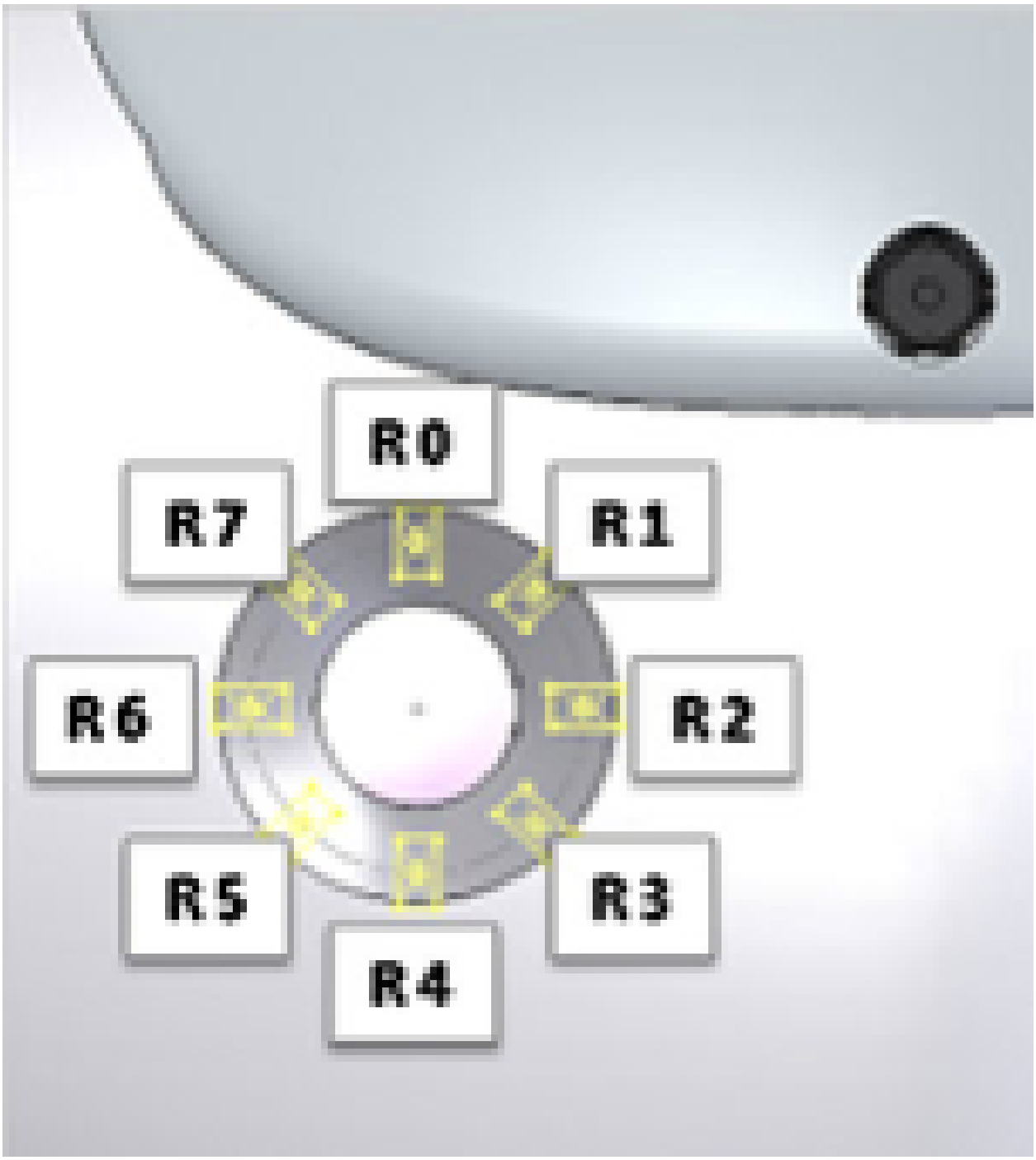
```
int32 L0=0
int32 L1=1
int32 L2=2
int32 L3=3
int32 L4=4
int32 L5=5
```

(continues on next page)

(continued from previous page)

```
int32 L6=6
int32 L7=7
int32 NUM_LEDS=8
```

4.2.5 Right Eye Leds



The following list is from `RightEyeLeds.msg`:

```
int32 R0=0
int32 R1=1
int32 R2=2
int32 R3=3
int32 R4=4
int32 R5=5
int32 R6=6
int32 R7=7
int32 NUM_LEDS=8
```

CHAPTER

FIVE

RELATED ROS2 PACKAGES

- nao_lola package

  Deals with the NAO's RoboCup-tailored Lola middle-ware.

- naosoccer_sim package

  Simulates a Nao in the SimSpark 3D soccer simulator.

- soccer_visualization package

  Visualizes topics in a soccer domain, in RViz.

- naosoccer_visualization package

  Visualizes topics from a Nao in a soccer domain, in RViz.

- nao_interfaces package

  Defines ROS2 interfaces for the Nao

- nao package

  Provides packages specific to the Nao

- soccer_interfaces package

  Defines ROS2 interfaces specific to Soccer.

- team_ijnek package

  Example of team independent code that makes the robot play soccer.

- naosoccer_pos_action package

  Makes the robot perform a keyframe motion