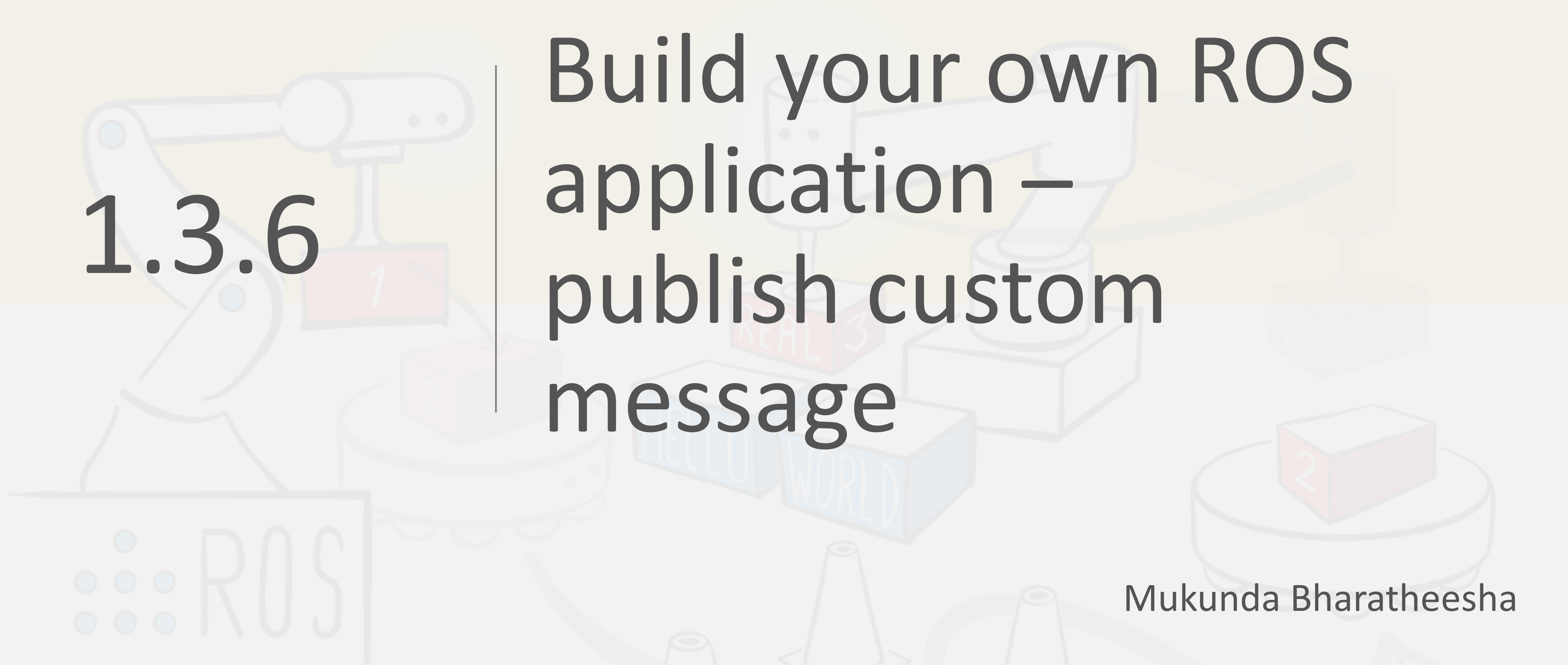

1.3.6

Build your own ROS application – publish custom message

Mukunda Bharatheesha

Preparing your script file

If you've entered a new CCS, don't forget to source your workspace if you want to use ROS commands! If you've followed the procedure under the Course Setup section correctly, you only need to execute the following in the shell:

```
$ source $HOME/hrwros_ws/devel/setup.bash
```

Let's first navigate to our Week 1 workspace folder

```
$ roscd hrwros_week1
```

Enter the scripts folder

```
$ cd scripts
```

Preparing your script file

Then, create our node python script:

```
$ touch sensor_info_publisher.py
```

We can now copy the contents of the template file template_publisher_script.py to our new python script:

```
$ cp template_publisher_script.py sensor_info_publisher.py
```

Editing your script file

For editing our script file you can use any text editor on Linux (like nano/gedit/etc.) like with the following command outside the CCS:

```
$ gedit sensor_info_publisher.py
```

First, change the last line of the header comment to ensure its consistency (#L36). Import our newly created message type SensorInformation which is possible because we executed the catkin build command.

Give the function a relevant name, for example, SensorInfoPublisher. (#L41)

We do the same for the Publisher object. The rospy.publisher function takes in the topic name and a message type. In this case, we will use 'sensorinfo' and as message type SensorInformation. (#L43) Now give the node a meaningful name. (#L44)

Create a new SensorInformation object and fill in its contents.

sensor_info = SensorInformation()

Editing your script file

Important Note:
The object sensor_info should not be confused with the topic sensor_info.

Fill in the header information with the timestep and distance data of the sensor.

sensor_info.sensor_data.header.stamp = rospy.Time.now()

sensor_info.sensor_data.header.frame_id = 'distance_sensor_frame'

Now Fill in the sensor data information. There you can determine the minimum and maximum range of the sensor.

sensor_info.sensor_data.radiation_type = sensor_info.sensor_data.ULTRASOUND

Fill in the manufacturer name and part number.

sensor_info.maker_name = 'The Ultrasound Company'

sensor_info.part_number = 123456

Editing your script file

Sensor data needs to be updated frequently. Therefore a loop is necessary. (#L65)

sensor_info.sensor_data.range = getSensorData(sensor_info.sensor_data.radiation_type, sensor_info.sensor_data.min_range, sensor_info.sensor_data.max_range)

Publish the updated sensor data on this sensor_infor topic. (#L70)

si_publisher.publish(sensor_info)

Let's call the correct function in the if loop. (#L75)

sensorInfoPublisher()

Starting your new ROS node

First run roscore in a CCS after sourcing the setup files. Then, in another CCS, the ROS node with the following command, after sourcing the setup files:

```
$ rosrun hrwros_week1 sensor_info_publisher.py
```

This will lead to an executable error. This error can be resolved by a simple command which makes the node executable.

```
$ roscd hrwros_week1/scripts
$ chmod +x sensor_info_publisher.py
```

Try running the ROS node with the previous command. This will lead to an import error. This is due to an incorrect message definition. We can change this with the gedit editor.

```
$ gedit sensor_info_publisher.py
```

Starting your new ROS node

There we need to change std_msgs.msg to hrwros_msgs.msg.

Try to run the node again. Which leads to a third error, the getSensorData function is not known.

For that reason, we always need to tell python where we got functions which means it is necessary to import the utilities. So we add the following with the gedit editor in sensor_info_publisher.py.

```
from hrwros_utilities.sim_sensor_data import distSensorData as
getSensorData
```

Now you can run the ROS node without errors.

Inspecting our node

This node is a publisher, publishing to a topic. So let's see if we can find the topic.

```
$ rostopic list (we look for /sensor_info topic)
```

Let's look at its contents.

```
$ rostopic echo /sensor_info
```

It's always nice to receive some acknowledgement that a ROS node is working. To do that we open our node in the gedit editor.

```
$ gedit sensor_info_publisher.py
```

After publishing our topic we print a log message by adding the following linen of code to the file:

```
rospy.loginfo('All went well. Publishing topic')
```