

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-the-kobuki-robot-setup?module\_item\_id=818044)

# Boot Camp Part 1: The Kobuki Robot Setup

## The Kobuki Robot

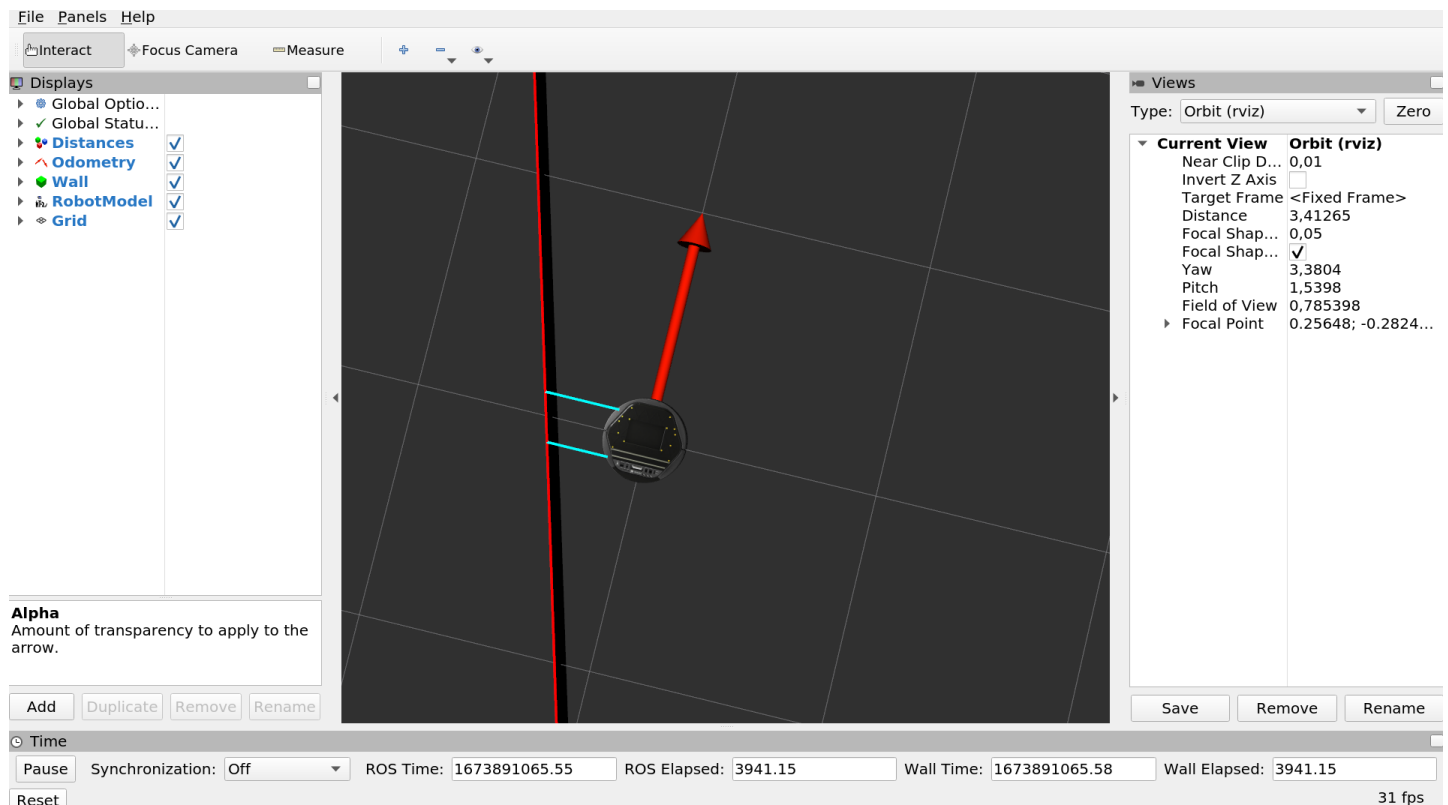
The Kobuki robot is a two-wheeled differential drive mobile robot.

## Launching the Kobuki Robot Setup for the Boot Camp

To launch the Kobuki setup that you will use for the boot camp, with simulated DC motors that control the motion of each of the wheels, distance sensors, and a virtual wall, we have provided a simple launch file where you can run everything in one go. Run the following command:

```
ros2 launch robp_boot_camp_launch boot_camp_part1_launch.xml
```

An RViz window with the following configuration should pop up:






As you can see there is a red virtual wall and some light blue lines which go from the distance sensors mounted on the robot to the wall. The red arrow indicates the odometry of the robot. You can move the robot by publishing to the `/motor/duty_cycle` topic:

Login (<https://app.kth.se/kpm/auth/login?>

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-the-kobuki-robot-setup?

module\_item\_id=818044)

Topic **Robotics** Type **cArray** Freq. **1** Hz   

topic	type	rate	expression
<input checked="" type="checkbox"/> <b>/motor/duty_cycles</b>	<b>robp_msgs/DutyCycles</b>	<b>10.00</b>	
▶ <b>header</b>	<b>std_msgs/Header</b>		
<b>duty_cycle_left</b>	<b>float64</b>		<b>1</b>
<b>duty_cycle_right</b>	<b>float64</b>		<b>1</b>

Inspect the contents of the launch file that you just ran. Run the following in the terminal:

```
gedit ~/dd2419_ws/src/robp_boot_camp/robp_boot_camp_launch/launch/boot_camp_part1_launch.xml
```

or

```
code ~/dd2419_ws/src/robp_boot_camp/robp_boot_camp_launch/launch/boot_camp_part1_launch.xml
```

if you want to open it in VS Code.

As you see, this launch file runs a number of nodes including the Kobuki simulation node, the distance sensors, the "world" node which sets up the virtual wall, the motors node and RViz.

To list all available topics run:

```
ros2 topic list
```

The most important topics for the boot camp are:

1. **/motor/duty\_cycles** (msg type: *robp\_interfaces/msg/DutyCycles*) publish to this topic to set the duty cycle value of each motor and control the motion of the robot.
2. **/motor/encoders** (msg type: *robp\_interfaces/msg/Encoders*) contains the encoder values of each motor on each of Kobuki's two wheels.
3. **/kobuki/adc** (msg type: *robp\_boot\_camp\_interfaces/msg/ADConverter*) contains the measurements from the distance sensors.

You can, e.g., see the values published by the distance sensor by running:

```
ros2 topic echo /kobuki/adc
```

If you want to check what the message type for a particular topic you can run:

```
ros2 topic info /kobuki/adc
```

Login (<https://app.kth.se/kpm/auth/login?>

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-the-kobuki-robot-setup?module\_item\_id=818044)

Every time you launch the launch file the virtual wall will be placed at a different (random) angle. This is to ensure that you properly test your controller in different situations and to make sure that simple open-loop heuristic controllers do not work :)

## The Distance Sensors

Distance sensors are analog electronic sensors that are used to measure the distance between the sensor and whatever is in front of the sensor. The Kobuki setup that you will use in this assignment will include two sensors that will help you measure the alignment with respect to a simulated wall.

**The sensors are separated by a distance of 20 cm.**

The simulated distance sensor that you will use for this assignment resembles *somewhat* the short-range Sharp IR sensors ([ir\\_gp\\_2d120\\_ss\\_datasheet.pdf](https://www.sharpsensors.com/2012/02/28/2d120-ss-datasheet.pdf) (<https://canvas.kth.se/courses/7229/files/1223734/download?wrap=1>)).

The sensor has an operational range from **10 to 80 cm**, in which the voltage output of the sensor decreases as the measured distance increases (look at the [sensor datasheet](https://canvas.kth.se/courses/7229/files/1223734/download) (<https://canvas.kth.se/courses/7229/files/1223734/download>) for more info, **although the simulated sensor response is not 100% like the real one!!!**). There is some gaussian white noise added to the sensor output as well.

The code for running the simulated distance sensor is located in the **robp\_boot\_camp\_distance\_sensor** package. The sensors will provide a simulated adc value that must be converted into a distance before being used. You can use  $d = 1.114e^{(-0.004adc)}$  as the sensor function, where x is the adc value and d the distance in meters.

## Distance sensor topic /kobuki/adc

Run

```
ros2 interface show robp_boot_camp_interfaces/msg/ADConverter
```

to see the message definition for this topic.

When launching the "boot\_camp\_part1\_launch.xml" file as described above, the **front distance sensor** will be published under **ch1** of the **/kobuki/adc** topic while the **back distance sensor** will be published under **ch2** of the **/kobuki/adc** topic.

The values you see under **/kobuki/adc** do not correspond to raw voltage values (i.e., decimal numbers) of the sensor but rather quantized **integer values** that can range from 0 to 1023. This is

Login (<https://app.kth.se/kpm/auth/login?>

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-the-kobuki-robot-setup?

linearly voltage values in the range 0-5V to 10-bit (0 to 1023) integer values.

## The DC Motors

In order to control the motion of the robot, you will have to control the DC motors that move each of the wheels of the robot. If you are not familiar with the operation of (differential) DC motors, please refer to the course book/lectures or the internet and read up a bit. (<http://www.ikalogic.com/closed-loop-speed-and-position-control-of-dc-motors/>) (<http://www.ikalogic.com/closed-loop-speed-and-position-control-of-dc-motors/>)

What you basically need to know is that DC motors are controlled through **duty cycle** signals which can be modulated to regulate the amount of power fed to the motor. This in turn affects the *angular* velocity at which the wheel rotates.

However, a duty cycle signal alone is *not enough* to fully control the speed at which the wheel attached to the motor rotates. External torques applied to the wheel lower the output angular velocity. Moreover, motors usually have different responses even if they come from the same manufacturer. This is why DC motors normally include **encoders**, which provide a feedback signal that allow us to determine the angular position and infer how fast the motor is actually rotating.

The simulated DC motors of this assignment get updated at **10 Hz**. If 0.5 seconds pass without receiving a duty cycle signal message, the motors shut down and the robot stops.

## The /motor/duty\_cycles Topic

Run

```
ros2 interface show robp_interfaces/msg/DutyCycles
```

from a terminal to see the message definition for this topic.

This topic allows you to control the duty cycle signal sent to each of the motors of the robot.

**/motor/duty\_cycles/duty\_cycle\_left** corresponds to the **left wheel** while

**/motor/duty\_cycles/duty\_cycle\_right** corresponds to the **right wheel**. The duty cycle signal is a **float** that can range between  $[-1.0, 1.0]$ . A duty cycle signal of **1.0** will drive the wheel forward at max speed while **-1.0** will drive the wheel at max speed in the opposite direction.

## The /motor/encoders Topic

Run

```
ros2 interface show robp_interfaces/msg/Encoders
```

to see the message definition for this topic.

Login (<https://app.kth.se/kpm/auth/login?>

nextUrl=https://canvas.kth.se/courses/45204/pages/boot-camp-part-1-the-kobuki-robot-setup?module\_item\_id=818044)

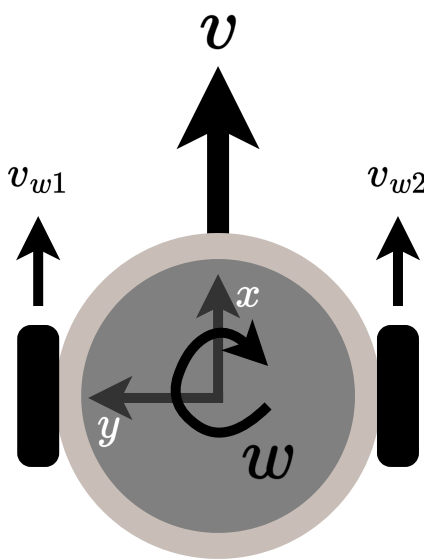
the **encoder\_right** and **delta\_encoder\_right** fields correspond to the **right wheel**.

The **encoder\_left** and **encoder\_right** fields are absolute encoder values. However, for control purposes, we are usually more interested in *differential* encoder values (**delta\_encoder\_left** and **delta\_encoder\_right**) which indicate how much the encoder has changed since the last control cycle, which is directly related to the angular velocity of the wheel.

## Encoder Parameters

The encoders for this lab have 360 ticks per revolution. This means that a change of 1 in the differential encoder signal (delta\_encoder) corresponds to a change of 1 degree in the angular position.

## Robot Kinematics



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

$$v = \frac{v_{w1} + v_{w2}}{2}$$

$$w = \frac{v_{w2} - v_{w1}}{2b}$$

$$v_{w_i} = \frac{2\pi r f \Delta_{\text{enc}}}{\text{ticks per rev}}$$

The kinematic parameters for the differential configuration are:

- ticks per rev: 360
- Wheel radius (r): 0.0352 m
- Base (b): 0.23 m