Laboratory focused on Trilobot

ROBa - Laboratory number 5

https://github.com/Adam-Fabo/ROB-laboratories

Faculty of Information Technology
Brno University of Technology
Adam Fabo
29.4.2023

Laboratory focused on Trilobot

Welcome to the fifth laboratory. This laboratory is going to connect ROS with Trilobot

Prerequisites

Prerequisites are that you already know basics of ROS and ROS commands (start ROS master, run node, etc.) and know how to create custom nodes using Python.

Goals of the laboratory

Goals of this laboratory are to use knowledge about ROS, motors and sonars in real life. You will learn:

- Architecture of robot trilobot
- Controlling motors using ROS
- Reading data from sonars using ROS
- Controlling robot trilobot
- Creating obstacle avoiding algorithm

Theory

Trilobot

Trilobot is a robot platform that is designed to be used by students that are learning robotics. It can be controlled by ROS.

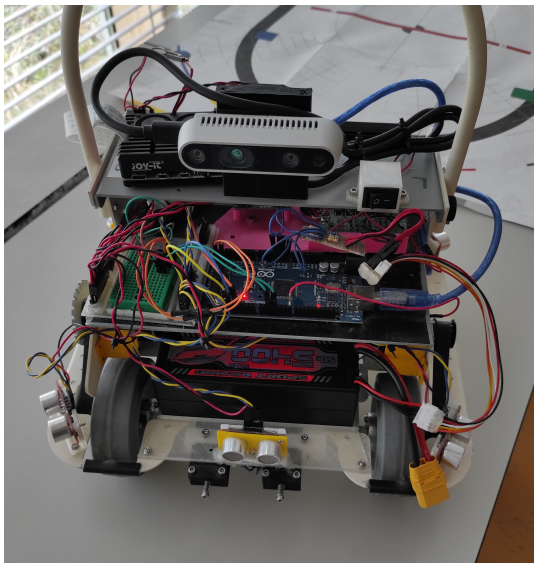


Figure 1: Robot Trilobot

The main "brain" of Trilobot is Raspberry Pi (Presne RPI here). This means that there is operating system running on each Trilobot (Presny linux here). This Rpi has touchscreen connected to it, which can be used for running scripts. Another microprocessor onboard is Arduino mega. Arduino is connected to Rpi with serial bus and is responsible for controlling its sensors and actuators.

Motor and sensors used by Trilobot were already taken in first part of this course. Motors used are Faulhaber 16002. Trilobot contains 2 of these motors - one for each wheel. Sensors used are SRF08. Architecture of Trilobots data flow can be seen at Figure 2.

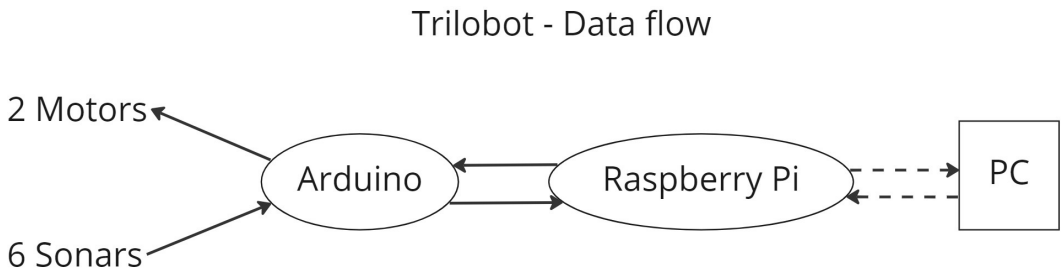


Figure 2: Data flow of the Trilobot

ROS - More complex messages

ROS allows for sending more complex messages than a simple string. But these message objects need to be created first and then sent. Two more complex messages used in this lab are Twist and Sonar_data.

Twist message comes from geometry_msg package. This message has 2 parts: linear and angular. Each part is a 3D vector consisting of 3 parts: x, y, z. This message can and is used for controlling movement of the Trilobot. Its alternative Python definition looks like following:

```
class Twist:
    def __init__(self):
        self.linear  = Linear()
        self.angular = Angular()


class Linear:
    def __init__(self, x = 0.0, y = 0.0, z = 0.0):
        self.x = x
        self.y = y
        self.z = z


class Angular:
    def __init__(self, x = 0.0, y = 0.0, z = 0.0):
        self.x = x
        self.y = y
        self.z = z
```

This means that Twist message can be used for example like this:

```
message = Twist()
message.linear.x  = 5
message.angular.z = 5

pub.publish(message)
```

Sonar_data message is used for reading data from sonars. Its alternative Python definition looks like following:

```
class Sonar_data:
    def __init__(self, front=0, front_left=0, front_right=0,
                       back=0, back_left=0, back_right=0,):
        self.front       = front
        self.front_left  = front_left
        self.front_right = front_right
        self.back        = back
        self.back_left   = back_left
        self.back_right  = back_right
```

This means that Sonar_data message can be used for example like this:

```
message = Sonar_data()
message.front  = 5
message.back_left = 5

pub.publish(message)
```

Trilobot - Motors and sonars

As already mentioned, Trilobot contains 2 motors and 6 sonars. It is needed to know the position and architecture of these parts for correct usage of Trilobot.

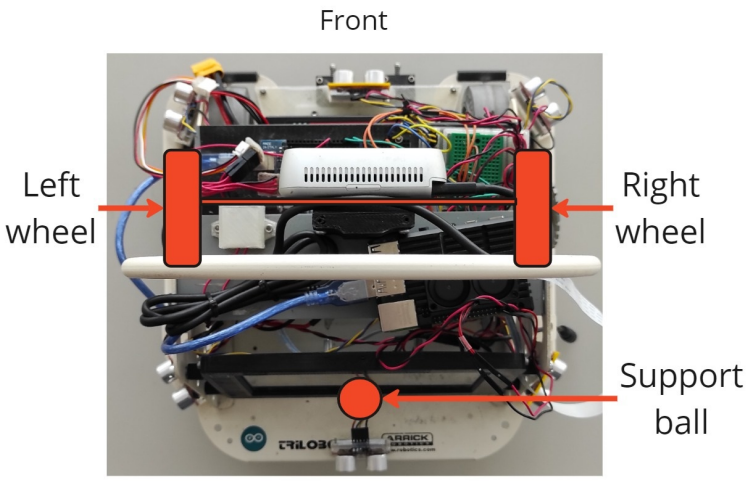
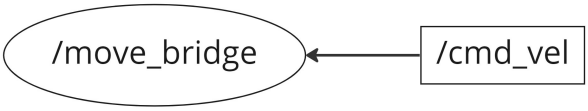


Figure 3: Location of wheels and support ball on Trilobot

Motors that Trilobot uses are Faulhaulber 16002. This is robot's main and only source of movement. It has only one axle and each wheel can move independently. Instead of second axle with wheels, Trilobot has a ball that supports the weight of the robot.

Motors are connected to Arduino which is connected to Rpi via serial bus and communication is provided by rossserial. This allows for simple controlling of motors. Motors can be controlled by writing Twist message into /cmd_vel topic. Trilobot only reacts to linear.x and angular.z values.



Sonars used in Trilobot are SRF08. These sonars communicate via I2C with Arduino, which communicates with Rpi. Trilobot has 6 sonars that are positioned around the perimeter of Trilobot.

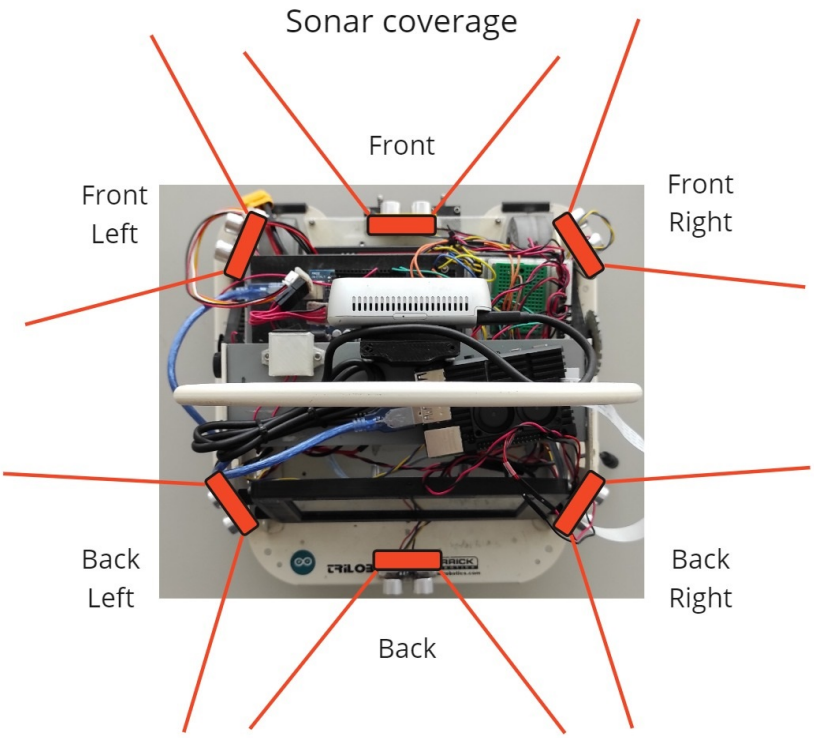
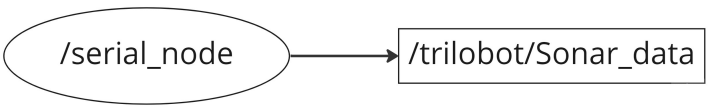


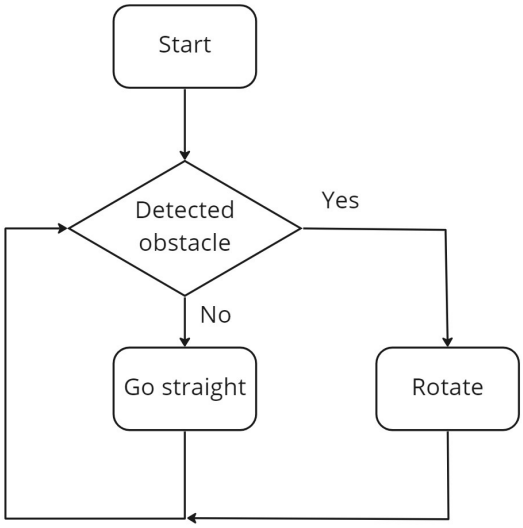
Figure 4: Data flow of the Trilobot

Measured distances from sonars are stored in message type sonar_data. This data are sent into /trilobot/Sonar_data topic.



Simple obstacle avoiding algorithm

Simplest obstacle avoiding algorithm can be described as following: Go straight until obstacle is detected. If an obstacle is detected, rotate to a given angle and continue straight again. This can be also seen on image below.



Controlling robot with keyboard - Pygame

Another interesting way of controlling Trilobot is using arrows on the keyboard. For this Pygame library was chosen, which is primarily used for creating games in Python, but offers API for getting key inputs. Firstly Pygame needs to be initialized.

```
import pygame
from pygame.locals import *

# Initialize pygame window
pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption('Pygame - Trilobot')
pygame.mouse.set_visible(1)
```

Getting pressed keys can be done using pygame.key.get_pressed() function. This function returns array of states of all keys. If a key is pressed, value is true. To access state of keys, constants should be used. Constants can be imported from pygame.locals import *. Sample constants are: K_UP, K_DOWN, K_LEFT, K_RIGHT and K_LSHIFT.

```
keys = pygame.key.get_pressed()

# If UP arrow is pressed
if keys[K_UP]:
    print("UP Arrow has been pressed!")
```

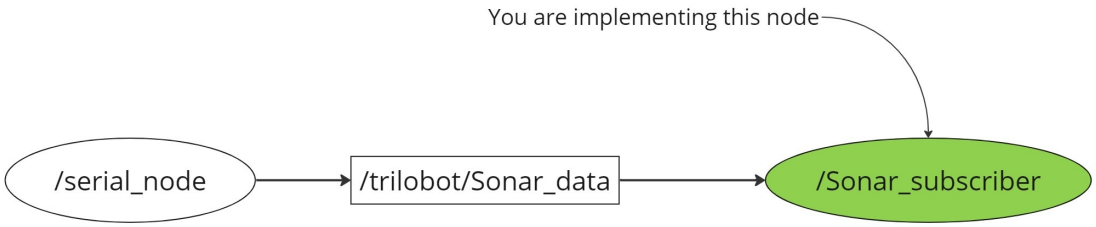
On order to get key inputs, Pygame window needs to be on top of all windows and be in focus.

Exercises

Reading sonar data

Test if Trilobot's sonars are working. Read the data from topic /trilobot/Sonar_data and print them in console.

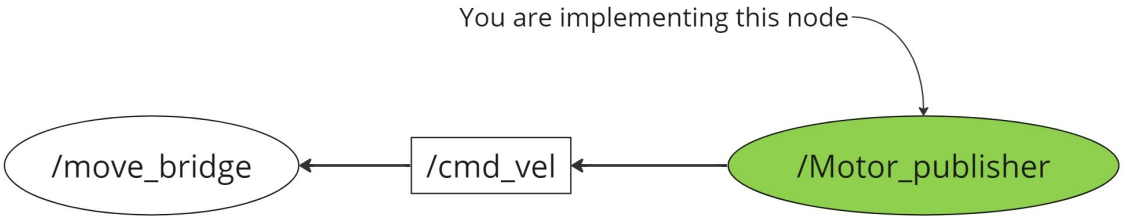
1. Open file **trilobot_sonars/src/sonars.py**
2. Implement subscriber that reads messages from topic /trilobot/Sonar_data
3. Run your node with `rosrun trilobot_sonars sonars.py`
4. Test your implementation by placing obstacles around Trilobot and observing read values.



Basic motor control

Test if Trilobot's motors are working

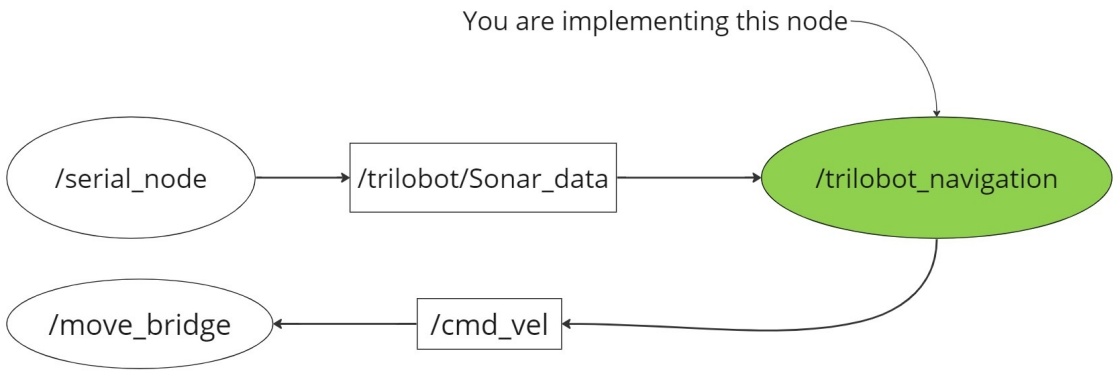
1. Place Trilobot on a box in a way, that its wheels are in the air
2. Open file **trilobot_motors/src/motors.py**
3. Implement publisher that publishes Twist messages into /cmd_vel topic
4. Run your node with `rosrun trilobot_motors motors.py`
5. Test your implementation by writing different values into Twist message.



Obstacle avoiding

Program simple obstacle avoiding algorithm.

1. Open file **trilobot_navigation/src/navigation.py**
2. Implement node that reads sonar data and based on that controls motors
3. Run your node with `rosrun trilobot_navigation navigation.py`
4. Test your implementation by placing Trilobot on the ground and placing an obstacle in front of it.



Controlling Trilobot using arrows

Make it possible to control Trilobot using keyboard arrows.

1. Open file **trilobot_arrows/src/arrows.py**
2. Implement arrow control using pygame library. Dictionary keys for keyboard keys are: K_UP, K_DOWN, K_LEFT and K_RIGHT.
3. Run your code by running python command or running it from IDE (Do not run it using rosrun).
4. Test your implementation by placing Trilobot on ground and pressing keyboard arrow keys
5. BONUS: make robot go even faster when left shift (K_LSHIFT) is pressed