Task Sheet 1

Introduction to the Thymio II

Deadline 09:00am November 3, 2023

Review on November 3, 2023

Lecture: *Introduction to Autonomous Robotics*, Winter Term 2023/24

Lecturer: Prof. Dr.-Ing. Heiko Hamann

Tutor: Jonas Kuckling & Simay Atasoy

In the practical exercises, we will use the mobile robot Thymio II in the Webots simulator and in the real world. The Thymio II robot is a small mobile robot with a differential drive, five buttons, seven horizontal infrared (IR) sensors, and two ground IR sensors. Its full specification can be found at `http://wiki.thymio.org/en:thymiospecifications`. We extend the Thymio II with an external battery and a Raspberry Pi (see Fig. 1) to be able to program the robot in Python 3 and to extend it with additional sensors.

In this exercise sheet, we set up our system for the practical exercises and learn the basics of programming the Thymio II with Python 3.

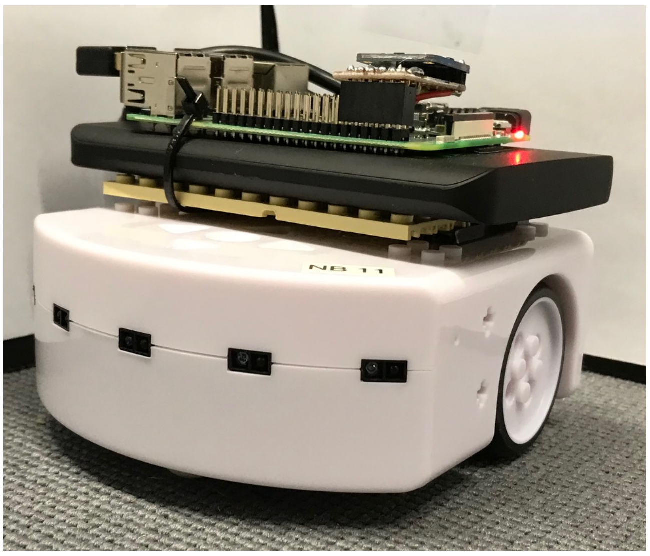

Figure 1: Thymio II extended with an external battery and a Raspberry Pi.

Task 1.1 Installation

On your PC, install the following programs and packages:

- Python 3 $\geq$ 3.6
- Webots simulator (version R2022b): `https://cyberbotics.com`
- Python package *thymiodirect*: `https://pypi.org/project/thymiodirect/`

```
pip3 install thymiodirect
```

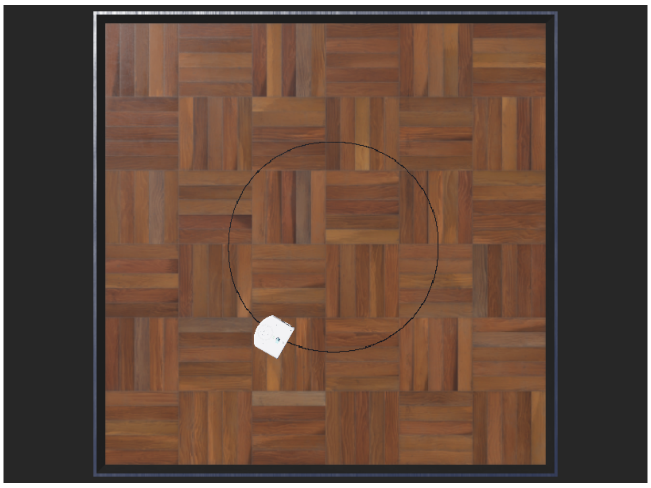

Figure 2: Test of Setup: Thymio II driving in a large circle in Webots.

We have tested the setup on Ubuntu and macOS.[1]

The Python package *thymiodirect* connects to the internal control system of the Thymio II robot via a serial or TCP port and enables us to program the Thymio II in Python 3. This allows us to execute the same code on the real and the simulated robots by setting the connection parameters appropriately. The Python package uses a native binary protocol to update (read and write) the internal robot variables up to 10 times per second. You can find a list of all variables of the Thymio II robot here: `http://wiki.thymio.org/en:thymioapi`. We will introduce you to all variables and functionalities required for the exercises in this exercise sheet.

Task 1.2 Test of Setup

After installing all required software, we test if it works as expected using a simple sample program.

a) Download and extract the ZIP-file *IAR1.zip* from ILIAS.

b) Launch the Webots simulator:

   macOS/Windows: Double click on the application icon

   Ubuntu: Open a new terminal and run the command

```
webots
```

c) In Webots, click *File > Open World*. Navigate to the folder *IAR1/worlds* (see a)) and open *IAR_test.wbt*.

d) Run the simulation (Play button).

e) Open a new terminal and navigate to the folder *IAR1*. Run the sample program using the command

```
python3 IAR1_test.py -s
```

   If everything was set up successfully, your Thymio II robot should drive in a large circle now as illustrated in Fig. 2.

Task 1.3 Basics of Programming the Thymio II

As a first introduction to programming the Thymio II robot, we provide example code showing the most important functionalities for the next exercises.

First, we import the required classes of the *thymiodirect* module.

```
from thymiodirect import Connection
from thymiodirect import Thymio
```

[1]Potentially, it also works on Windows. Unfortunately, we were not able to test it on time and cannot guarantee the functionality.

Next, we create a Thymio II object and connect the Raspberry Pi and the Thymio II via serial port. For this purpose, you can use the helper class *Connection* to use the standard interface. As soon as the Thymio II object was successfully created, we can connect with the robot and declare the first node as our robot. All variables internal to the robot will be updated automatically 10 times per second. To allow the robot to initialize all variables and to guarantee that each variable was updated at least once, we add a delay of one second before starting our robot control.

```
# Get default port and create Thymio connection object
port = Connection.serial_default_port()
th = Thymio(serial_port=port, on_connect=lambda node_id:print(f"Thymio {node_id} connected"))

# Thymio connection object is ready -> connect -> select first connection node as robot
th.connect()
robot = th[th.first_node()]

# Delay to allow robot initialization of all variables
time.sleep(1)
```

In the Webots simulator, we do not connect via serial port but via the local network using a TCP connection. Consequently, we have to modify the connection parameters. In our exercises, we always use port 2001 for the first robot and additional robots on the consecutive ports (2002,2003,...). As long as our code is executed on the same PC as the simulation, we use the host “localhost”. Otherwise, the IP address of the PC running the simulation has to be specified.

```
# Chose TCP port (default for first robot=2001) and create Thymio connection object
port = 2001
th = Thymio(use_tcp=True, host="localhost", tcp_port=port, on_connect=lambda node_id:print(f"
                                                Thymio {node_id} connected"))

# Thymio connection object is ready -> connect -> select first connection node as robot
th.connect()
robot = th[th.first_node()]

# Delay to allow robot initialization of all variables
time.sleep(1)
```

Now, we can communicate with the robot to read and to set variables. Next, you will explore the functionalities of the Thymio II robot

a) Restart the Webots simulation as explained in Exercise 1.2 b) to d).

b) Open the file *IAR1_exercise,py* in the folder *IAR1* and add the following code in the main loop:

   ```
   # Print all available variables
   print(th.variables(th.first_node()))
   ```

   The function *th.variables()* outputs all available variables of the Thymio II robot.

   In a new terminal, navigate to the folder *IAR1*. Run your program using the command

   ```
   python3 IAR1_exercise.py −s
   ```

   You should see the list of available variables printed in the terminal.

   In the terminal, press *Ctrl + C* to stop the program.

c) Now, extend your code by

   ```
   while True:
     print(robot['prox.horizontal'])
   ```

   Run your program again using the command

   ```
   python3 IAR1_exercise.py −s
   ```

In an infinite loop, the current values of the horizontal proximity sensor values will be printed into the terminal. The values of the horizontal proximity sensors range from 0 to ca. 4500.

In Webots, click on the Thymio II robot and move it around in the arena. You can see the sensor values changing based on the closeness to obstacles (i.e., the walls here).

You can access individual sensor values by using indices, for example,

```
robot['prox.horizontal'][2]
```

to access the value of the front middle proximity sensor.

In our exercises, we need the values of the following sensors:

```
  # Returns an array with the readings of all 7 horizontal proximity sensors
  proxH = robot["prox.horizontal"]

  # Returns an array with the reflection readings of both ground proximity sensors
  proxG = robot["prox.ground.reflected"]

  # Returns the state of the buttons
  buttonC = robot["button.center"]
  buttonF = robot["button.forward"]
  buttonB = robot["button.backward"]
  buttonL = robot["button.left"]
  buttonR = robot["button.right"]
```

To press buttons of the Thymio in the Webots simulator, you have to open its robot window. Right click on the robot and select *Show Robot Window*. A new tab in your browser opens, showing a sketch of the Thymio II robot that visualizes the current sensor values and allows you to click the robots five buttons.

In the terminal, press *Ctrl + C* to stop the program.

d) Last, but not least, we set the motor values of the Thymio II robot. The robot should drive straight forward and stop in front of the wall.

To accomplish this, add the following lines in the main loop:

```
if robot["prox.horizontal"][2] < 1000:
 robot['motor.left.target'] = 200
 robot['motor.right.target'] = 200
else:
 robot['motor.left.target'] = 0
 robot['motor.right.target'] = 0
```

The target speeds for both robot values can be set in a range of $[-500, 500]$ corresponding to a maximum linear speed of ± 0.2 m/s.

The rest of the code in the provided template *IAR1_exercise.py* facilitates the use in simulation and on the real robot by providing parameters to configure the connection to the robot. The execution of your scripts can be done in the following ways (*your_programm.py* has to be replaced with your respective Python file):

- Execution on a real Thymio II robot

```
python3 your_program.py
```

- Execution on a simulated Thymio II robot using the standard port (2001)

```
python3 your_program.py -s
```

- Execution on a second simulated Thymio II robot using the port 2002

```
python3 your_program.py -s -p 2002
```

- Overview of all these options

```
python3 your_program.py -h
```

Task 1.4   Basics of Programming the Thymio II

Read through the following example code for a simple robot controller. The robot is started by pressing the forward button and stopped when pressing the center button. When the controller is running, the robot turns on spot.

```python
import argparse
import time
from thymiodirect import Connection
from thymiodirect import Thymio

def main(use_sim=False, ip='localhost', port=2001):
    try:
        # Configure Interface to Thymio robot
        if use_sim:
            th = Thymio(use_tcp=True, host=ip, tcp_port=port,
                        on_connect=lambda node_id: print(f' Thymio {node_id} is connected'))
        else:
            port = Connection.serial_default_port()
            th = Thymio(serial_port=port,
                        on_connect=lambda node_id: print(f'Thymio {node_id} is connected'))

        # Connect to Robot
        th.connect()
        robot = th[th.first_node()]

        # Delay to allow robot initialization of all variables
        time.sleep(1)

        # initialize state variable
        state = "STOP"

    # Main loop
    while True:
      # Check buttons to switch between states
      if robot["button.forward"] == 1:
        state = "GO"
      elif robot["button.center"] == 1:
        state = "STOP"

      # Check the state and turn circle if GO otherwise stop
      if state == "GO":
        robot["motor.left.target"]=200
        robot["motor.right.target"]=-200

        # Print ground prox sensor variables
        print(robot["prox.ground.reflected"])
      else:
        robot["motor.left.target"]=0
        robot["motor.right.target"]=0

    except Exception as err:
        # Stop robot
        robot["motor.left.target"] = 0
        robot["motor.right.target"] = 0
        print(err)

if __name__ == '__main__':
    # Parse commandline arguments to cofigure the interface for a simulation.
    parser = argparse.ArgumentParser(description='Configure optional arguments to run the code
                                                  with simulated Thymio.')

    # Add optional arguments
    parser.add_argument('-s', '--sim', action='store_true', help='set this flag to use
                                                  simulation')
    parser.add_argument('-i', '--ip', help='set the TCP host ip for simulation. default=
                                                  localhost', default='localhost')
    parser.add_argument('-p', '--port', type=int, help='set the TCP port for simulation.
                                                  default=2001', default=2001)

    # Parse arguments and pass them to main function
    args = parser.parse_args()
    main(args.sim, args.ip, args.port)
```