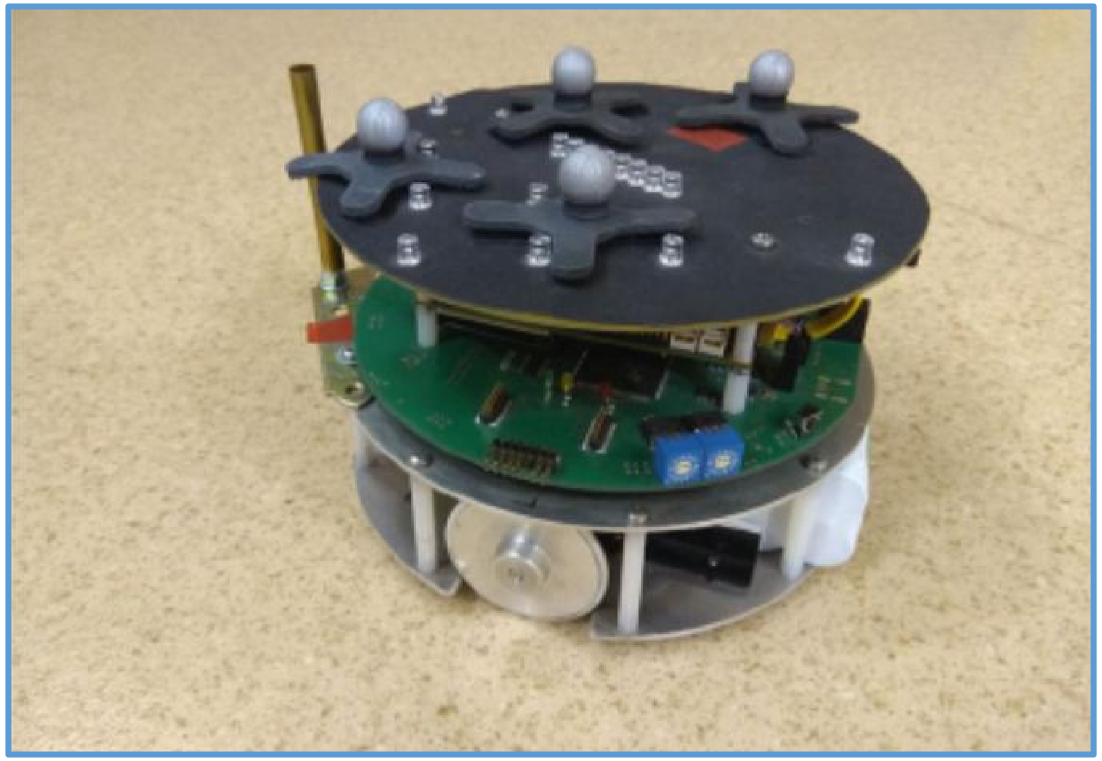
Simulation of Robots in ROS

Er. Arpit Joon
Doctorate Student
Poznan University of Technology

Unicycle-Like Mobile Robot / Two Wheeled Mobile Robot

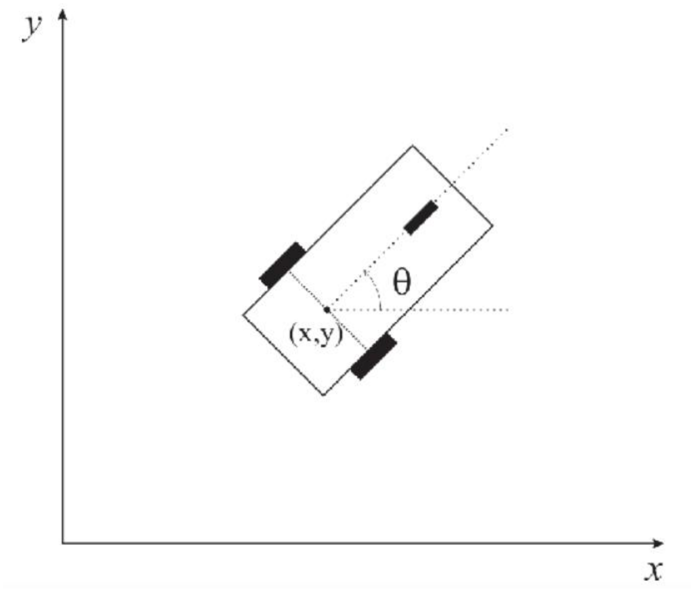
[11]


[12]

[11].Sun, Chung-Hsun et al. “Design of T-S fuzzy controller for two-wheeled mobile robot.” Proceedings 2011 International Conference on System Science and Engineering (2011): 223-228.
[12]. Kowalczyk, Wojciech. (2019). Rapid Navigation Function Control for Two-Wheeled Mobile Robots. Journal of Intelligent & Robotic Systems. 93. 10.1007/s10846-018-0879-4.

Kinematic Model of Unicycle-Like Mobile Robot

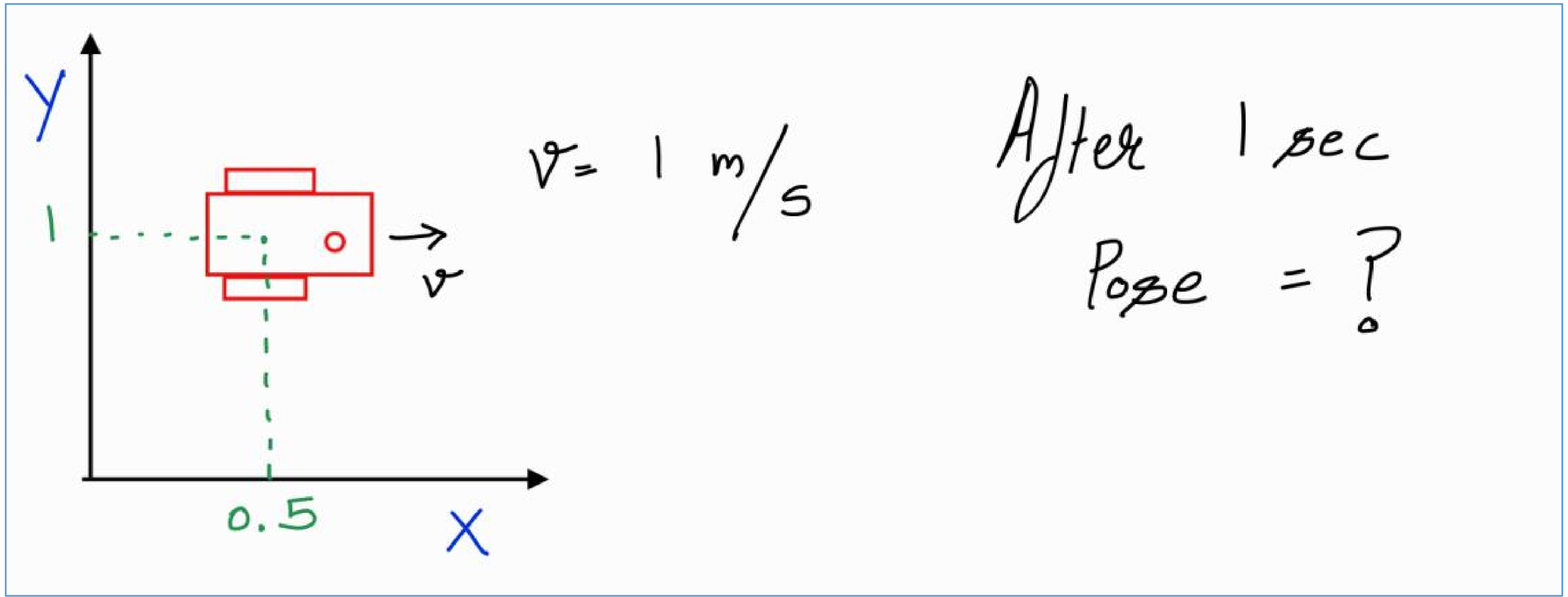
The kinematic model of the mobile robot is written as:



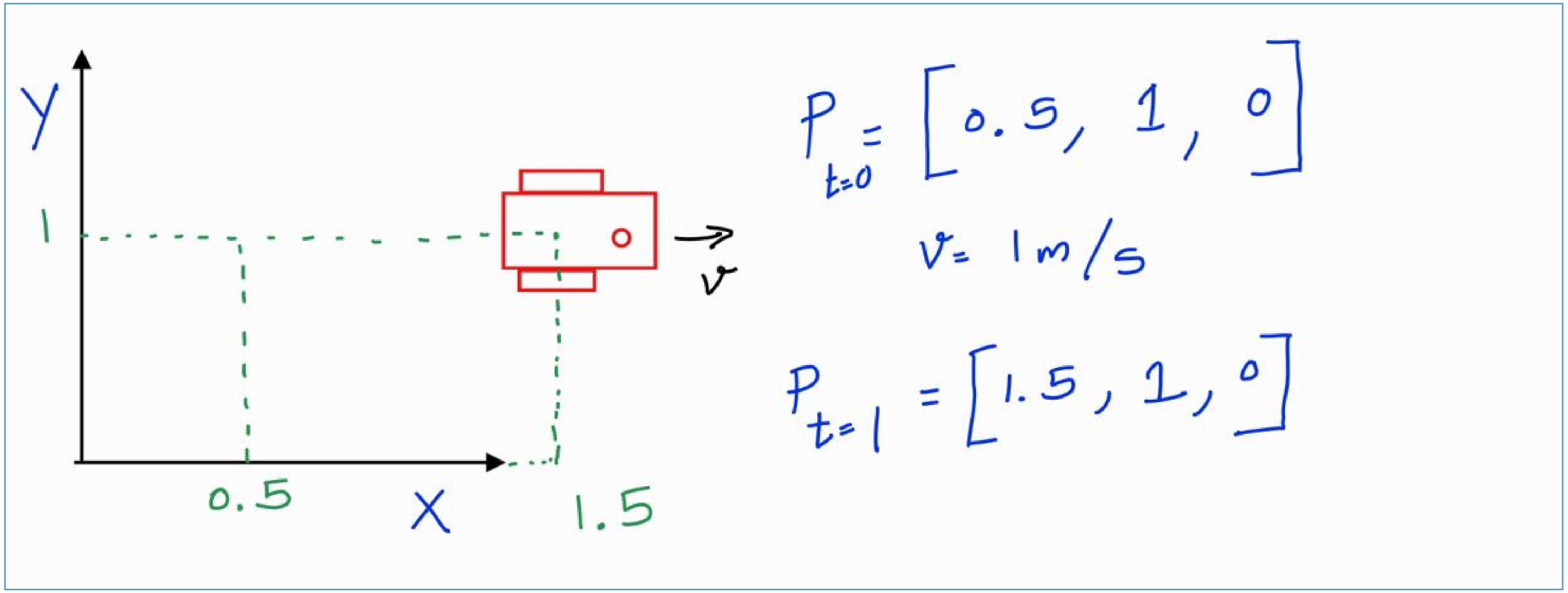
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

The robot pose is represented by vector $[x \quad y \quad \theta]^\top$. x, y, θ are the variables representing the pose of the robot in the global reference frame. The control vector is $[v \quad \omega]^\top$ where v and ω are linear and angular velocity controls of the robot respectively.

What after 1 sec?



What after 1 sec?

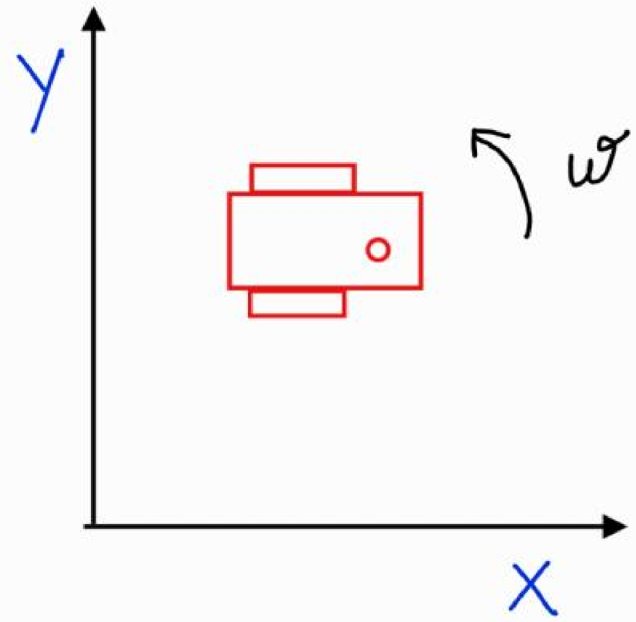


$$P_{t=0} = [0.5,\ 1,\ 0]$$

$$v = 1m/s$$

$$P_{t=1} = [1.5,\ 1,\ 0]$$

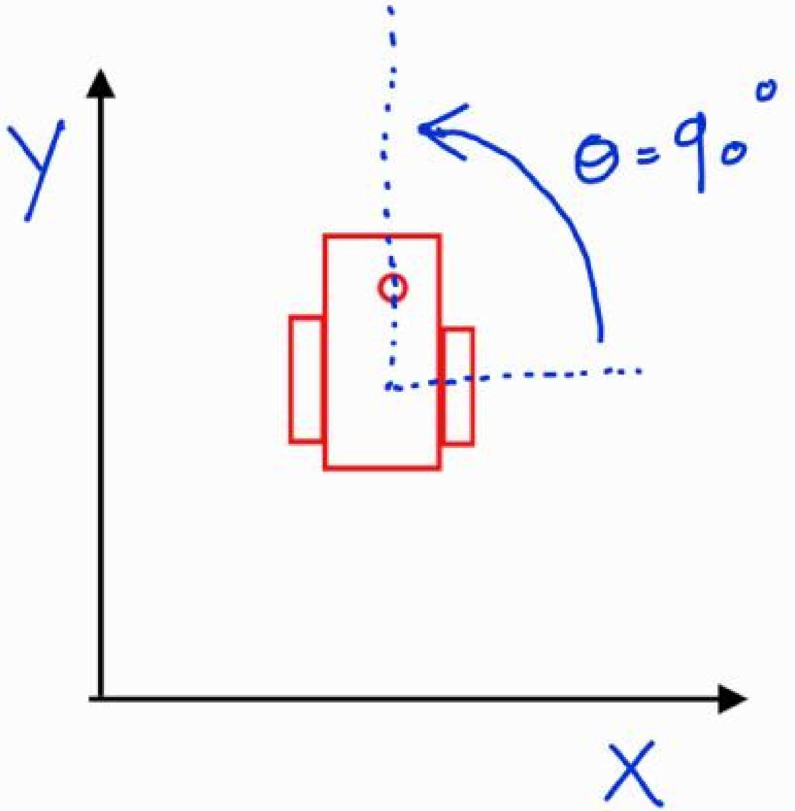
What after 1 sec?



$v = 0$

$\omega = 1.57 \text{ rad}/s$

After 1 sec = ?



$P_{t=0} = [0.5, 1, 0]$

$v = 0\ m/s\ ,\ \omega = 1.57$

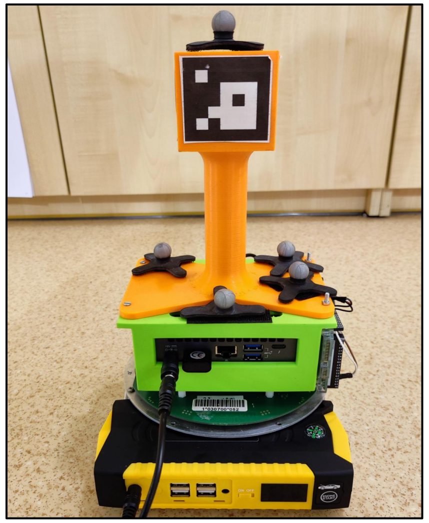
$P_{t=1} = [0.5, 1, 1.57]$

Control Algorithm

1) Control Leader and Follower Robots

The kinematic model for leader and follower mobile platform $R_i (i = 1, 2)$ is written as:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \quad (1)$$



Control Algorithm

where i=1 for leader robot and i=2 for follower robot. The pose of robots is represented by vector $[x_i \ \ y_i \ \ \theta_i]^\top$ where x_i represents the distance in the global x-axis, y_i in the y-axis and θ_i is the angle in the global reference frame. The control vector for robots is represented as $[v_i \quad \omega_i]^\top$ where v_i and ω_i are linear and angular velocity controls of the robot respectively. The task of the leader robot is to mimic the pose $[x_0 \ \ y_0 \ \ \theta_0]^\top$ of the virtual leader:

$$\begin{aligned} x_{1d} &= x_0 \\ y_{1d} &= y_0 \\ \theta_{1d} &= \theta_0 \end{aligned} \qquad (2)$$

The desired velocities vector is $[v_0 \ \ \omega_0]^\top$ where v_0 is the linear velocity and ω_0 is the angular velocity. With some constant displacement $[d_{2x} \ \ d_{2y}]^\top$ follower has to mimic the motion of the leader mobile platform:

$$\begin{aligned} x_{2d} &= x_1 + d_{2x} \\ y_{2d} &= y_1 + d_{2y} \end{aligned} \qquad (3)$$

Control Algorithm

and with the same orientation:

$$\theta_{2d} = \theta_1 \qquad (4)$$

which brings the following quantities to zero:

$$\begin{aligned} p_{ix} &= x_{id} - x_i \\ p_{iy} &= y_{id} - y_i \\ p_{i\theta} &= \theta_{id} - \theta_i . \end{aligned} \qquad (5)$$

The system errors in the fixed frame to the robot are written as follows:

$$\begin{bmatrix} e_{ix} \\ e_{iy} \\ e_{i\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & 0 \\ -\sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{i\theta} \end{bmatrix} . \qquad (6)$$

Control Algorithm

The trajectory tracking control algorithm is taken from [4] because of its effectiveness and simplicity. The robot control for i-th robot is as follows:

$$\begin{aligned} v_i &= v_{i-1}\cos e_{i\theta} + k_1 e_{ix} \\ \omega_i &= \omega_{i-1} + k_2 sgn(v_{i-1}) e_{iy} + k_3 e_{i\theta}, \end{aligned} \qquad (7)$$

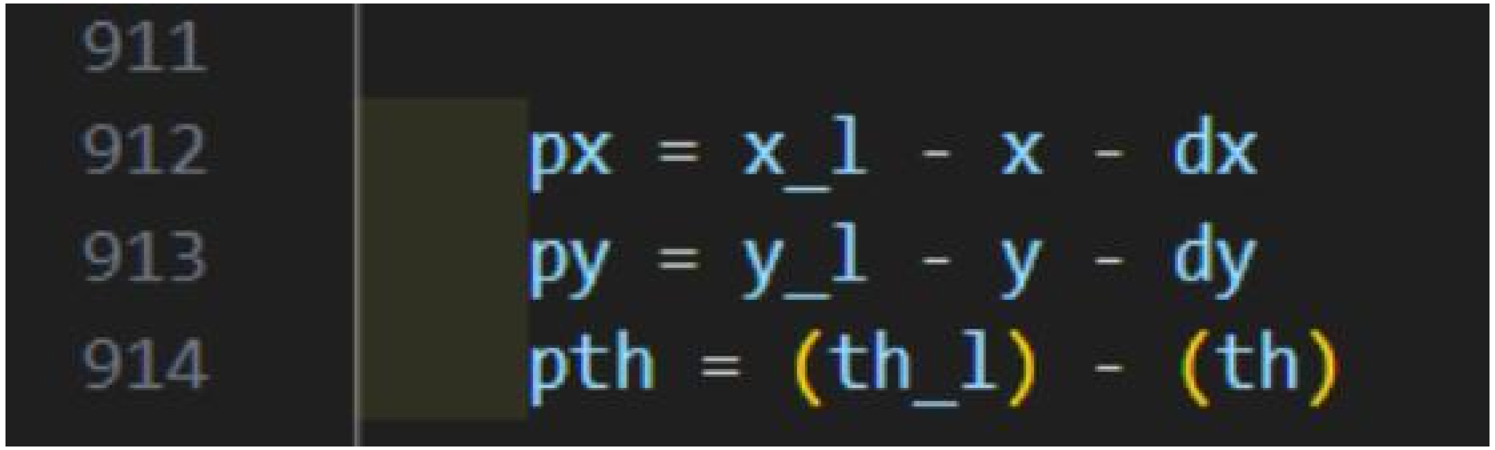
where k_1 ,k_2 and k_3 are constant parameters greater then zero and function $sgn(\bullet)$ is defined as follows:

$$sgn(\xi) = \begin{cases} -1 & \text{for} \quad \xi < 0 \\ 0 & \text{for} \quad \xi = 0. \\ 1 & \text{for} \quad \xi > 0 \end{cases} \qquad (8)$$

Control Algorithm

which brings the following quantities to zero:

$$\begin{aligned} p_{ix} &= x_{id} - x_i \\ p_{iy} &= y_{id} - y_i \\ p_{i\theta} &= \theta_{id} - \theta_i. \end{aligned} \qquad (5)$$

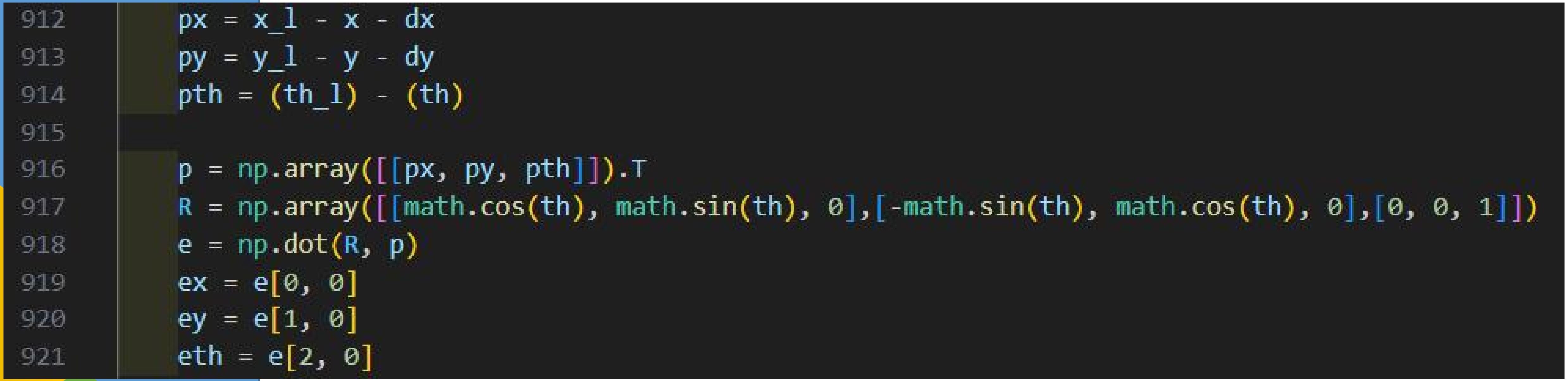


```
px = x_l - x - dx
py = y_l - y - dy
pth = (th_l) - (th)
```

Control Algorithm

The system errors in the fixed frame to the robot are written as follows:

$$\begin{bmatrix} e_{ix} \\ e_{iy} \\ e_{i\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & 0 \\ -\sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{i\theta} \end{bmatrix}. \qquad (6)$$

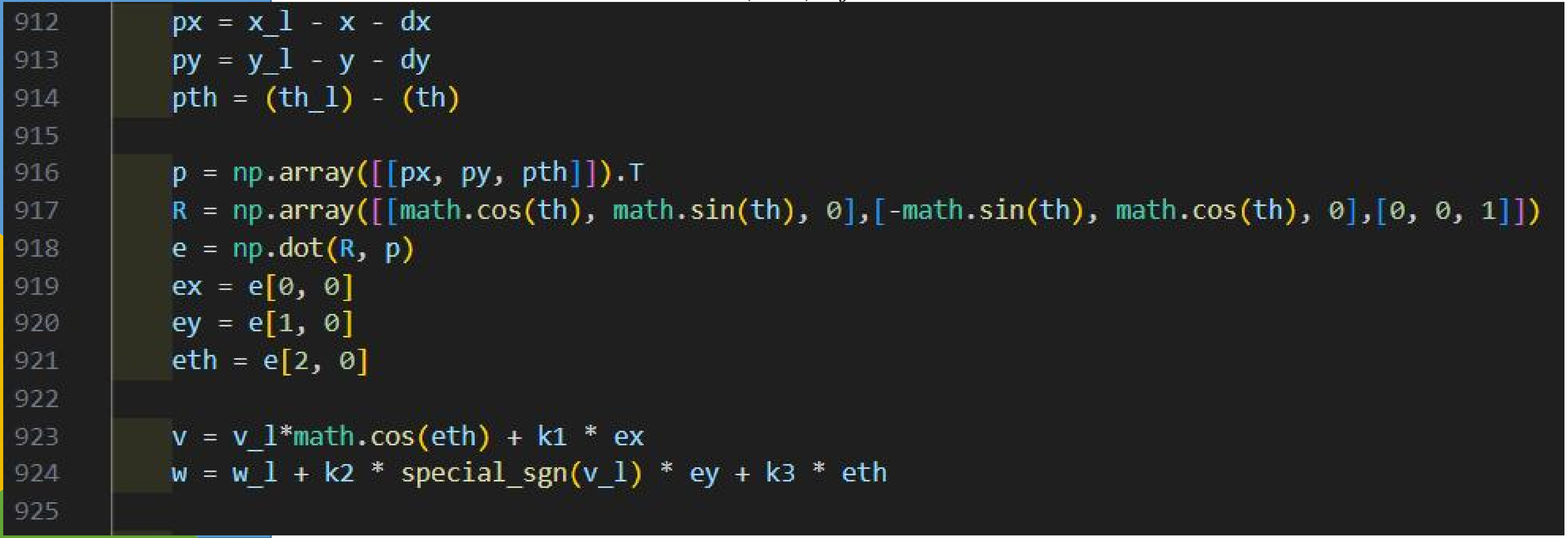


```
    px = x_l - x - dx
    py = y_l - y - dy
    pth = (th_l) - (th)

    p = np.array([[px, py, pth]]).T
    R = np.array([[math.cos(th), math.sin(th), 0],[-math.sin(th), math.cos(th), 0],[0, 0, 1]])
    e = np.dot(R, p)
    ex = e[0, 0]
    ey = e[1, 0]
    eth = e[2, 0]
```

Control Algorithm

The trajectory tracking control algorithm is taken from [4] because of its effectiveness and simplicity. The robot control for i-th robot is as follows:

$$\begin{aligned} v_i &= v_{i-1} \cos e_{i\theta} + k_1 e_{ix} \\ \omega_i &= \omega_{i-1} + k_2 sgn(v_{i-1}) e_{iy} + k_3 e_{i\theta}, \end{aligned} \quad (7)$$

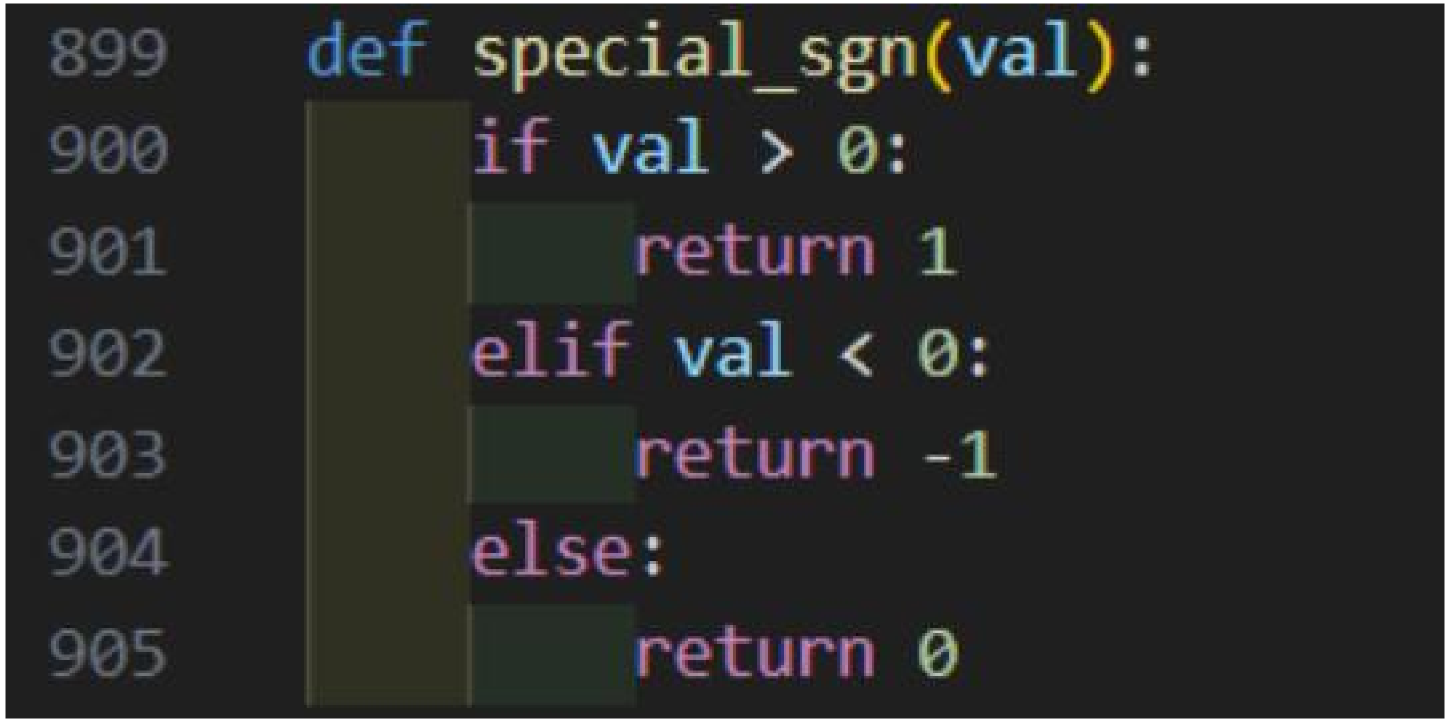


```
    px = x_l - x - dx
    py = y_l - y - dy
    pth = (th_l) - (th)

    p = np.array([[px, py, pth]]).T
    R = np.array([[math.cos(th), math.sin(th), 0],[-math.sin(th), math.cos(th), 0],[0, 0, 1]])
    e = np.dot(R, p)
    ex = e[0, 0]
    ey = e[1, 0]
    eth = e[2, 0]

    v = v_l*math.cos(eth) + k1 * ex
    w = w_l + k2 * special_sgn(v_l) * ey + k3 * eth
```

Control Algorithm

where k_1 ,k_2 and k_3 are constant parameters greater then zero and function $sgn(\bullet)$ is defined as follows:

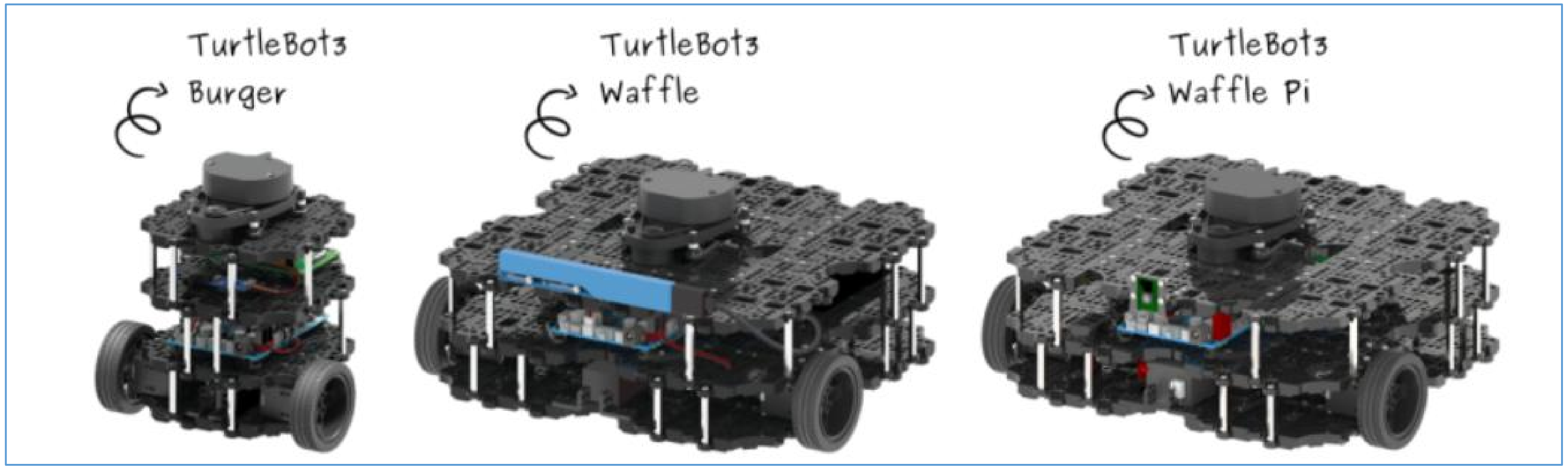
$$sgn(\xi) = \begin{cases} -1 & \text{for} \quad \xi < 0 \\ 0 & \text{for} \quad \xi = 0. \\ 1 & \text{for} \quad \xi > 0 \end{cases} \qquad (8)$$



```
def special_sgn(val):
    if val > 0:
        return 1
    elif val < 0:
        return -1
    else:
        return 0
```

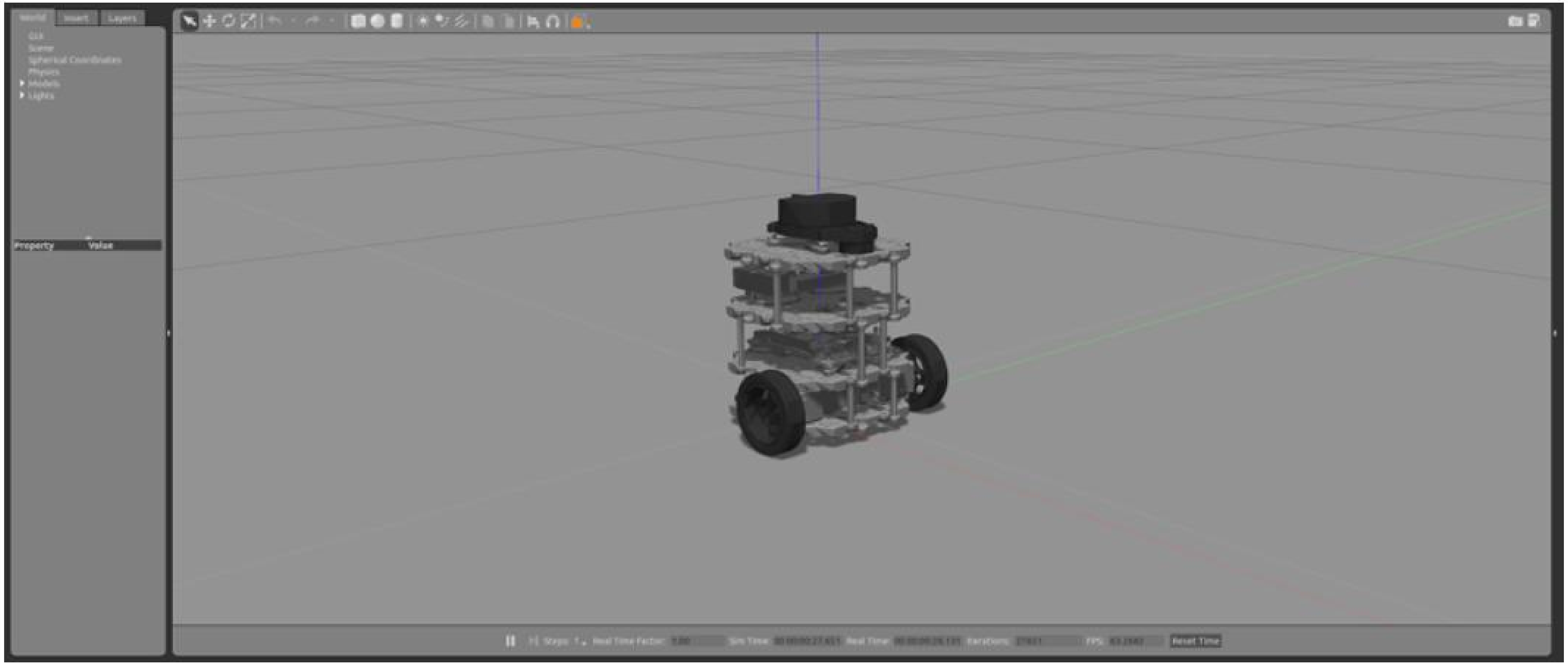
What is a TurtleBot?

TurtleBot is a low-cost, personal robot kit with open-source software. TurtleBot was created at Willow Garage by Melonee Wise and Tully Foote in November 2010. With TurtleBot, you'll be able to build a robot that can drive around your house, see in 3D, and have enough horsepower to create exciting applications.



Turtlebot in Empty world

1. Empty World



```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

Video 1

Thanks

THANK YOU FOR LISTENING

Eamil id's:

joonarpit@gmail.com

arpit.joon@doctorate.put.poznan.pl