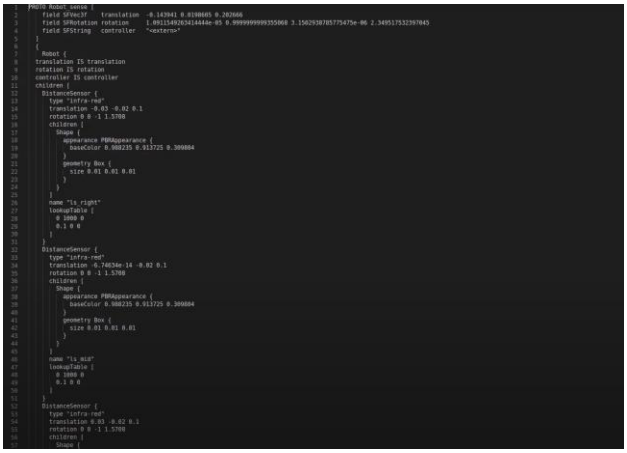
Raken Putra Athallah

TK-44-02

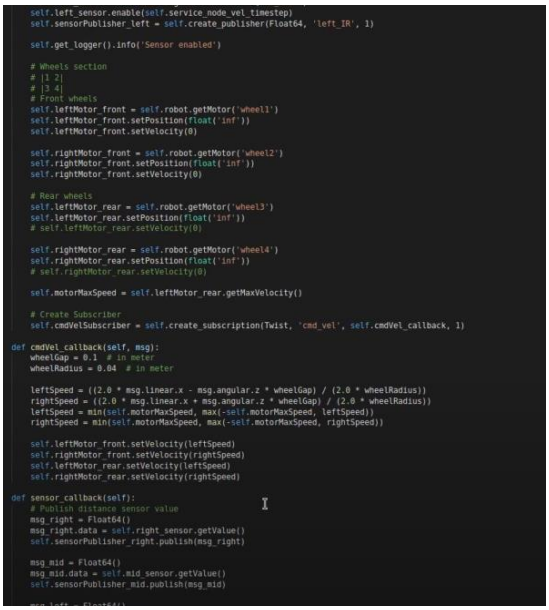
1103204186

Pada video tutorial ini dijelaskan kita akan membuat robot line follower dengan membuat custom robot di webots yang memiliki beberapa sensor untuk menyelesaikan task-task yang diinginkan.Pada kod dibawah ini kita dapat melihat sensor-sensor yang ada didalam robot yang telah dibuat



```
PROTO Robot_sensa [
  field SFVec3f    translation   -0.143041 0.01004605 0.202666
  field SFRotation rotation      1.0911548263614444e-05 0.9999999999355068 3.1502938785775475e-06 2.3495175323907845
  field SFString   controller    "<extern>"
]
{
  Robot {
  translation IS translation
  rotation IS rotation
  controller IS controller
  children [
    DistanceSensor {
      type "infra-red"
      translation -0.03 -0.01 0.1
      rotation 0 0 -1 1.5708
      children [
        Shape {
          appearance PBRAppearance {
            baseColor 0.980235 0.913725 0.300084
          }
          geometry Box {
            size 0.01 0.01 0.01
          }
        }
      ]
      name "ls_right"
      lookupTable [
        0 1000 0
        0.1 0 0
      ]
    }
    DistanceSensor {
      type "infra-red"
      translation -6.74634e-14 -0.02 0.1
      rotation 0 0 -1 1.5708
      children [
        Shape {
          appearance PBRAppearance {
            baseColor 0.980235 0.913725 0.300084
          }
          geometry Box {
            size 0.01 0.01 0.01
          }
        }
      ]
      name "ls_mid"
      lookupTable [
        0 1000 0
        0.1 0 0
      ]
    }
    DistanceSensor {
      type "infra-red"
      translation 0.03 -0.02 0.1
      rotation 0 0 -1 1.5708
      children [
        Shape {
```

Kemudian ada sebuah node yang bernama slave yang akan berinteraksi dengan webots.Isi dari node ini adalah untuk sebagai setup distance sensor dan juga kecepatan roda pada robot.Node ini merupakan bagian dari publisher, nilai sensor yang didapatkan akan dipublikasikan untuk digunakan oleh node master.



```
    self.left_sensor.enable(self.service_mode_vel_timestep)
    self.sensorPublisher_left = self.create_publisher(Float64, 'left_IR', 1)

    self.get_logger().info('Sensor enabled')

    # Wheels section
    # |1 2|
    # |3 4|
    # Front wheels
    self.leftMotor_front = self.robot.getMotor('wheel1')
    self.leftMotor_front.setPosition(float('inf'))
    self.leftMotor_front.setVelocity(0)

    self.rightMotor_front = self.robot.getMotor('wheel2')
    self.rightMotor_front.setPosition(float('inf'))
    self.rightMotor_front.setVelocity(0)

    # Rear wheels
    self.leftMotor_rear = self.robot.getMotor('wheel3')
    self.leftMotor_rear.setPosition(float('inf'))
    # self.leftMotor_rear.setVelocity(0)

    self.rightMotor_rear = self.robot.getMotor('wheel4')
    self.rightMotor_rear.setPosition(float('inf'))
    # self.rightMotor_rear.setVelocity(0)

    self.motorMaxSpeed = self.leftMotor_rear.getMaxVelocity()

    # Create Subscriber
    self.cmdVelSubscriber = self.create_subscription(Twist, 'cmd_vel', self.cmdVel_callback, 1)

def cmdVel_callback(self, msg):
    wheelGap = 0.1  # in meter
    wheelRadius = 0.04  # in meter

    leftSpeed = ((2.0 * msg.linear.x - msg.angular.z * wheelGap) / (2.0 * wheelRadius))
    rightSpeed = ((2.0 * msg.linear.x + msg.angular.z * wheelGap) / (2.0 * wheelRadius))
    leftSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, leftSpeed))
    rightSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, rightSpeed))

    self.leftMotor_front.setVelocity(leftSpeed)
    self.rightMotor_front.setVelocity(rightSpeed)
    self.leftMotor_rear.setVelocity(leftSpeed)
    self.rightMotor_rear.setVelocity(rightSpeed)

def sensor_callback(self):
    # Publish distance sensor value
    msg_right = Float64()
    msg_right.data = self.right_sensor.getValue()
    self.sensorPublisher_right.publish(msg_right)

    msg_mid = Float64()
    msg_mid.data = self.mid_sensor.getValue()
    self.sensorPublisher_mid.publish(msg_mid)
```

Kemudian ada node bernama master yang dimana node ini merupakan subscriber dari node slave yang dimana node ini akan melakukan perkiraan arah dari data atau nilai yang didapatkan dari node slave

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float64
from geometry_msgs.msg import Twist


class LineFollower(Node):
    def __init__(self):
        super().__init__('linefollower_cmdvel')
        # Subscribe IR sensors
        self.subs_right_ir = self.create_subscription(Float64, 'right_IR', self.rightIR_cb, 1)
        self.subs_left_ir = self.create_subscription(Float64, 'left_IR', self.leftIR_cb, 1)
        self.subs_mid_ir = self.create_subscription(Float64, 'mid_IR', self.midIR_cb, 1)
        # Publish cmd_vel
        self.pubs_cmdvel = self.create_publisher(Twist, 'cmd_vel', 1)

        # vehicle parameters
        self.speed = 0.2
        self.angle_correction = 0.01

        # Initialize parameters
        self.GS_RIGHT, self.GS_MID, self.GS_LEFT = 0, 0, 0
        self.DeltaS = 0
        self.cmd = Twist()
        self.stop = False
        self.count = 0
        self.count_threshold = 10

    def LineFollowingModule(self):...

    # Call backs to update sensor reading variables
    def rightIR_cb(self, msg):...

    def leftIR_cb(self, msg):...

    def midIR_cb(self, msg):...


def main(args=None):...


if __name__ == '__main__':
    main()
```