

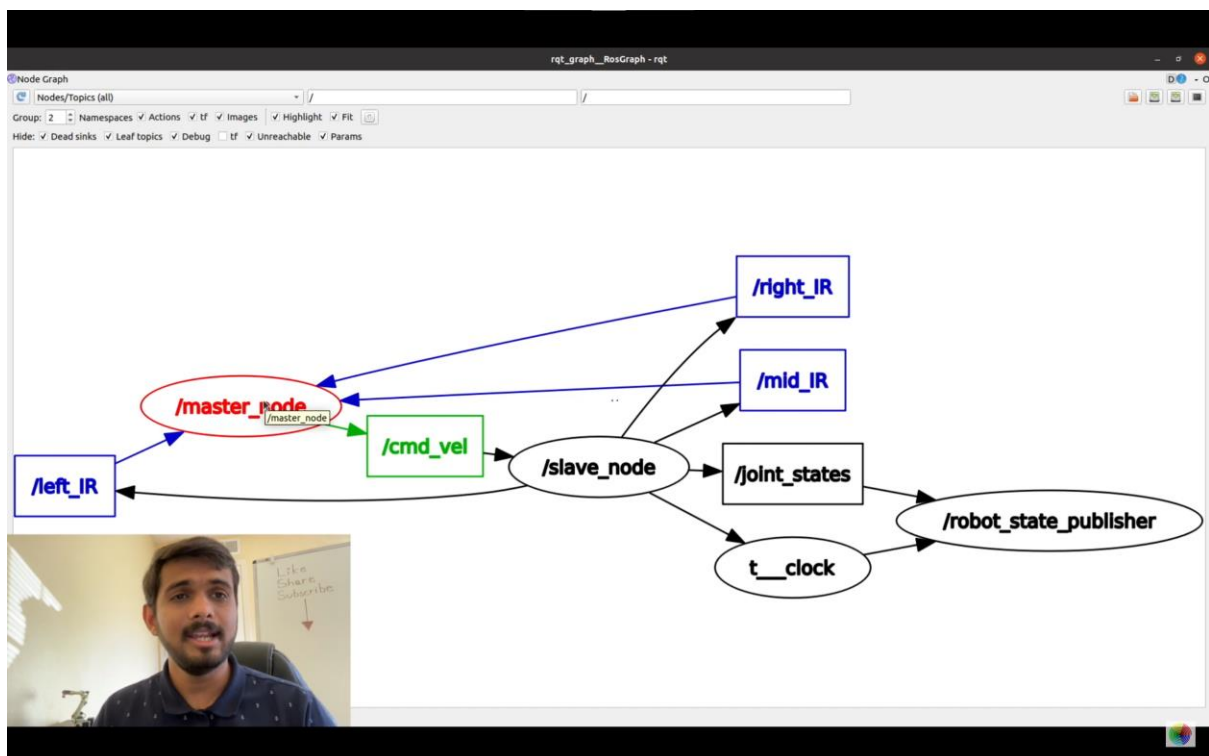
Line Following Custom Robot Project | Webots ROS2 project Tutorial | [Tutorial 6] Summary

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

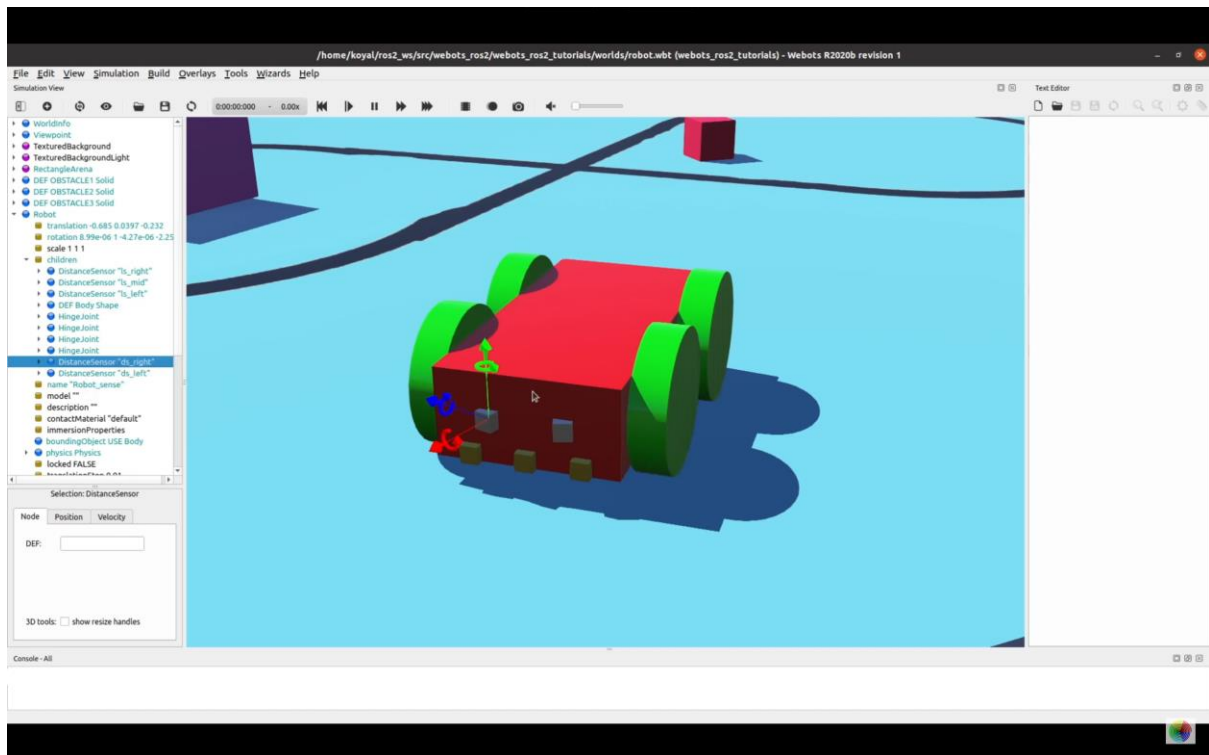
Soft Illusion Webots ROS2



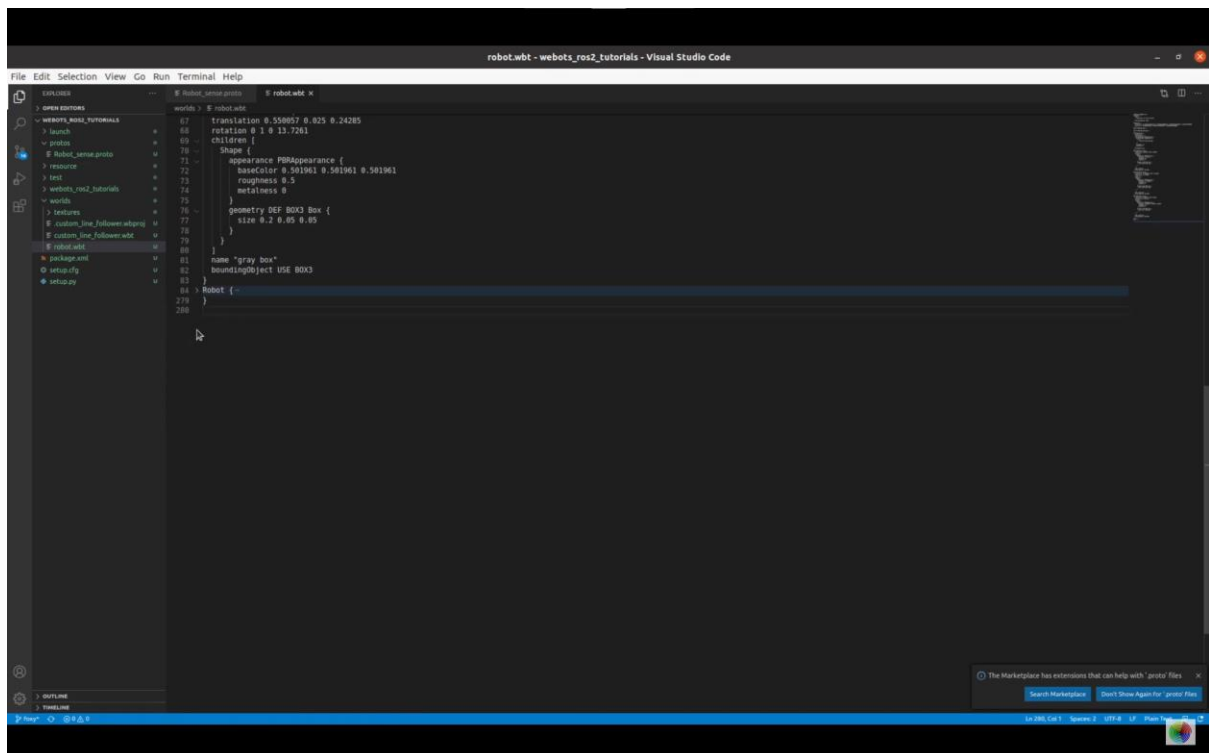
Soft Illusion dan sekilas tentang proyek tutorial menggunakan Webots dan ROS2 untuk membuat robot mengikuti garis.

Edit custom Robot in Webots

Membangun robot kustom untuk mengikuti garis hitam di Webots. Menyesuaikan bentuk dan sensor robot sesuai kebutuhan.



robot.wbt



Overview of Slave

Penjelasan tentang node "slave" yang berfungsi membaca data sensor dan mempublikasikannya ke node lain.

slave.py

```
slave.py - webots_ros2_tutorials - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  webots_ros2_tutorials
    launch
    robot_sensor.proto
    resource
    test
    webots_ros2_tutorials
      __init__.py
      master.py
      slave.py
      world
      webots
    custom_line_follower.wbt
    custom_line_follower.wbt
    robot.wbt
    package.xml
    setup.py
    setup.py

webots_ros2_tutorials > slave.py
1 # Copyright 1994-2020 Soft Illusion.
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 # http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 import rclpy
16 from std_msgs.msg import Float64
17 from std_msgs.msg import Twist
18 from geometry_msgs.msg import Twist
19
20 class ServiceNodeVel(Node):
21     def __init__(self, args):
22         super().__init__(f'slave_node', args)
23
24         # Enable 3 sensors
25         self.service_node_vel_timestep = 16
26
27         # Sensor section
28         self.sensor_timer = self.create_timer(0.01 * self.service_node_vel_timestep,
29                                             self.sensor_callback)
30
31         self.right_sensor = self.robot.getDistanceSensor('ls_right')
32         self.right_sensor.enable(self.service_node_vel_timestep)
33         self.sensor_publisher_right = self.create_publisher(Float64, 'right_IR', 1)
34
35         self.mid_sensor = self.robot.getDistanceSensor('ls_mid')
36         self.mid_sensor.enable(self.service_node_vel_timestep)
37         self.sensor_publisher_mid = self.create_publisher(Float64, 'mid_IR', 1)
38
39         self.left_sensor = self.robot.getDistanceSensor('ls_left')
40         self.left_sensor.enable(self.service_node_vel_timestep)
41         self.sensor_publisher_left = self.create_publisher(Float64, 'left_IR', 1)
42
43         self.get_logger().info('Sensor enabled')
44
45         # Wheels section
46         # [ 1 2 ]
47         # [ 3 4 ]
48         # Front wheels
49         self.leftMotor_front = self.robot.getMotor('wheel1')
50         self.leftMotor_front.setPosition(float('inf'))
51         self.leftMotor_front.setVelocity(0)
52
53         self.rightMotor_front = self.robot.getMotor('wheel2')
54         self.rightMotor_front.setPosition(float('inf'))
55         self.rightMotor_front.setVelocity(0)
56
57         # Rear wheels
58         self.leftMotor_rear = self.robot.getMotor('wheel3')
59         self.leftMotor_rear.setPosition(float('inf'))
60         self.leftMotor_rear.setVelocity(0)
61
62         self.rightMotor_rear = self.robot.getMotor('wheel4')
63         self.rightMotor_rear.setPosition(float('inf'))
64         self.rightMotor_rear.setVelocity(0)
65
66         self.motorMaxSpeed = self.leftMotor_rear.getMaxVelocity()
67
68         # Create Subscriber
69         self.create_subscription(Twist, 'cmd_vel', self.cmdVel_callback, 1)
70
71     def cmdVel_callback(self, msg):
72         wheelGap = 0.05 # ls wheelGap
73         wheelRadius = 0.05 # ls wheelRadius
74
75         leftSpeed = ((2.0 * msg.linear.x - msg.angular.z * wheelGap) / (2.0 * wheelRadius))
76         rightSpeed = ((2.0 * msg.linear.x + msg.angular.z * wheelGap) / (2.0 * wheelRadius))
77         leftSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, leftSpeed))
78         rightSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, rightSpeed))
79
80         self.leftMotor_front.setVelocity(leftSpeed)
81         self.rightMotor_front.setVelocity(rightSpeed)
82         self.leftMotor_rear.setVelocity(leftSpeed)
83         self.rightMotor_rear.setVelocity(rightSpeed)
84
85     def sensor_callback(self):
86         # Publish distance sensor value
87         msg_right = Float64()
88         msg_right.data = self.right_sensor.getValue()
89         self.sensor_publisher_right.publish(msg_right)
90
91         msg_mid = Float64()
92         msg_mid.data = self.mid_sensor.getValue()
93         self.sensor_publisher_mid.publish(msg_mid)
94
95         msg_left = Float64()
96         msg_left.data = self.left_sensor.getValue()
```

slave.py

```
slave.py - webots_ros2_tutorials - Visual Studio Code

File Edit Selection View Go Run Terminal Help

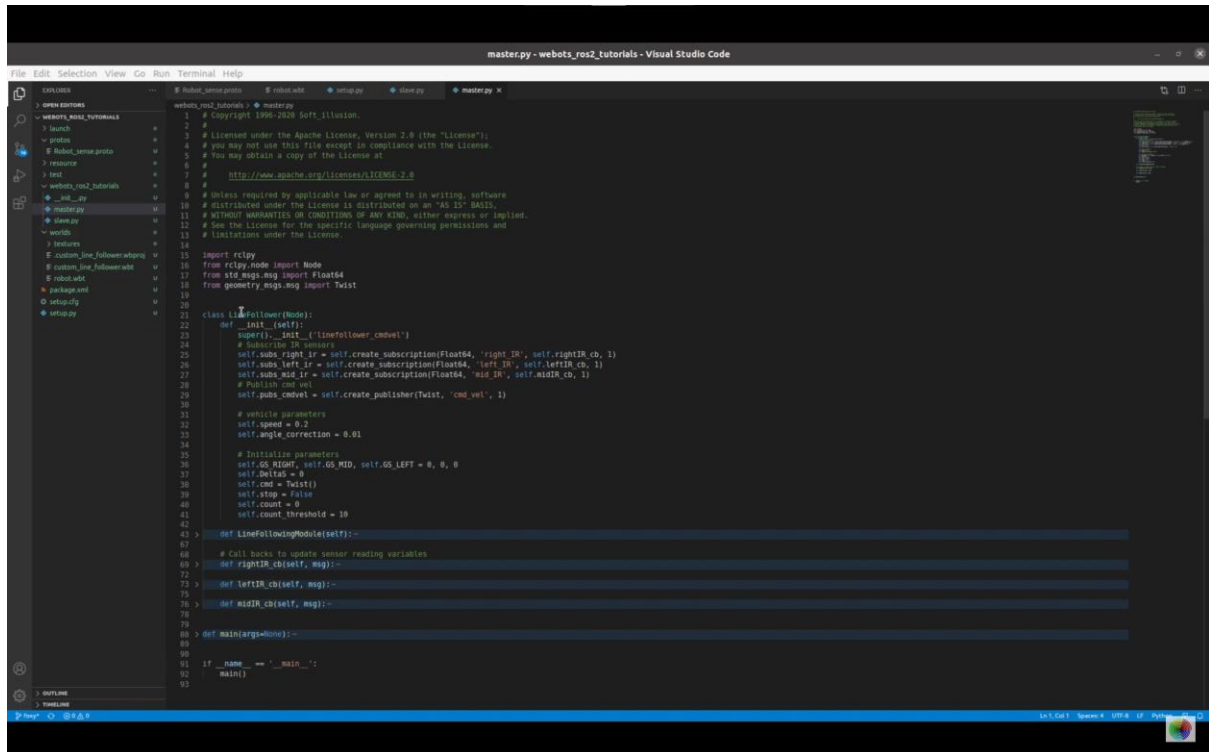
EXPLORER
  webots_ros2_tutorials
    launch
    robot_sensor.proto
    resource
    test
    webots_ros2_tutorials
      __init__.py
      master.py
      slave.py
      world
      webots
    custom_line_follower.wbt
    custom_line_follower.wbt
    robot.wbt
    package.xml
    setup.py
    setup.py

webots_ros2_tutorials > slave.py
38         self.sensor_publisher_mid = self.create_publisher(Float64, 'mid_IR', 1)
39
40         self.left_sensor = self.robot.getDistanceSensor('ls_left')
41         self.left_sensor.enable(self.service_node_vel_timestep)
42         self.sensor_publisher_left = self.create_publisher(Float64, 'left_IR', 1)
43
44         self.get_logger().info('Sensor enabled')
45
46         # Wheels section
47         # [ 1 2 ]
48         # [ 3 4 ]
49         # Front wheels
50         self.leftMotor_front = self.robot.getMotor('wheel1')
51         self.leftMotor_front.setPosition(float('inf'))
52         self.leftMotor_front.setVelocity(0)
53
54         self.rightMotor_front = self.robot.getMotor('wheel2')
55         self.rightMotor_front.setPosition(float('inf'))
56         self.rightMotor_front.setVelocity(0)
57
58         # Rear wheels
59         self.leftMotor_rear = self.robot.getMotor('wheel3')
60         self.leftMotor_rear.setPosition(float('inf'))
61         self.leftMotor_rear.setVelocity(0)
62
63         self.rightMotor_rear = self.robot.getMotor('wheel4')
64         self.rightMotor_rear.setPosition(float('inf'))
65         self.rightMotor_rear.setVelocity(0)
66
67         self.motorMaxSpeed = self.leftMotor_rear.getMaxVelocity()
68
69         # Create Subscriber
70         self.create_subscription(Twist, 'cmd_vel', self.cmdVel_callback, 1)
71
72     def cmdVel_callback(self, msg):
73         wheelGap = 0.05 # ls wheelGap
74         wheelRadius = 0.05 # ls wheelRadius
75
76         leftSpeed = ((2.0 * msg.linear.x - msg.angular.z * wheelGap) / (2.0 * wheelRadius))
77         rightSpeed = ((2.0 * msg.linear.x + msg.angular.z * wheelGap) / (2.0 * wheelRadius))
78         leftSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, leftSpeed))
79         rightSpeed = min(self.motorMaxSpeed, max(-self.motorMaxSpeed, rightSpeed))
80
81         self.leftMotor_front.setVelocity(leftSpeed)
82         self.rightMotor_front.setVelocity(rightSpeed)
83         self.leftMotor_rear.setVelocity(leftSpeed)
84         self.rightMotor_rear.setVelocity(rightSpeed)
85
86     def sensor_callback(self):
87         # Publish distance sensor value
88         msg_right = Float64()
89         msg_right.data = self.right_sensor.getValue()
90         self.sensor_publisher_right.publish(msg_right)
91
92         msg_mid = Float64()
93         msg_mid.data = self.mid_sensor.getValue()
94         self.sensor_publisher_mid.publish(msg_mid)
95
96         msg_left = Float64()
97         msg_left.data = self.left_sensor.getValue()
```

Overview of Master

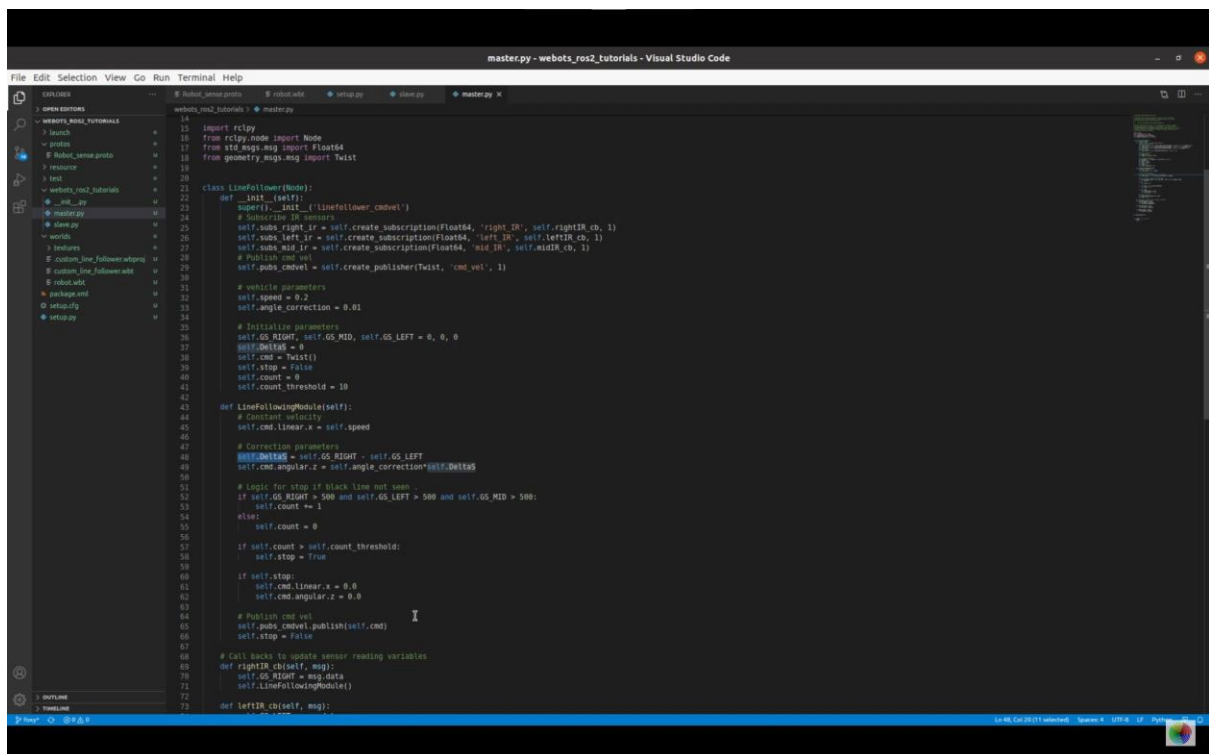
Penjelasan tentang node "master" yang menerima data sensor dari slave, memprosesnya, dan mengirimkan perintah kecepatan ke robot.

master.py



```
1 # Copyright 1996-2020 Soft InSilicon.
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 # http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 import rclpy
16 from rclpy.node import Node
17 from std_msgs.msg import Float64
18 from geometry_msgs.msg import Twist
19
20 class LineFollower(Node):
21     def __init__(self):
22         super().__init__('linefollower_cmsvel')
23         # Subscribe to sensors
24         self.subs_right_ir = self.create_subscription(Float64, 'right_IR', self.rightIR_cb, 1)
25         self.subs_left_ir = self.create_subscription(Float64, 'left_IR', self.leftIR_cb, 1)
26         self.subs_mid_ir = self.create_subscription(Float64, 'mid_IR', self.midIR_cb, 1)
27         # Publish cmd vel
28         self.pubs_cmdvel = self.create_publisher(Twist, 'cmd_vel', 1)
29
30     # vehicle parameters
31     self.speed = 0.2
32     self.angle_correction = 0.01
33
34     # Initialize parameters
35     self.GS_RIGHT, self.GS_MID, self.GS_LEFT = 0, 0, 0
36     self.DeltaS = 0
37     self.cmd = Twist()
38     self.stop = False
39     self.count = 0
40     self.count_threshold = 10
41
42     def LineFollowingModule(self):
43         # Call back to update sensor reading variables
44         self.rightIR_cb(self, msg)
45         self.leftIR_cb(self, msg)
46         self.midIR_cb(self, msg)
47
48     def mainargophone():
49         if __name__ == '__main__':
50             main()
```

master.py

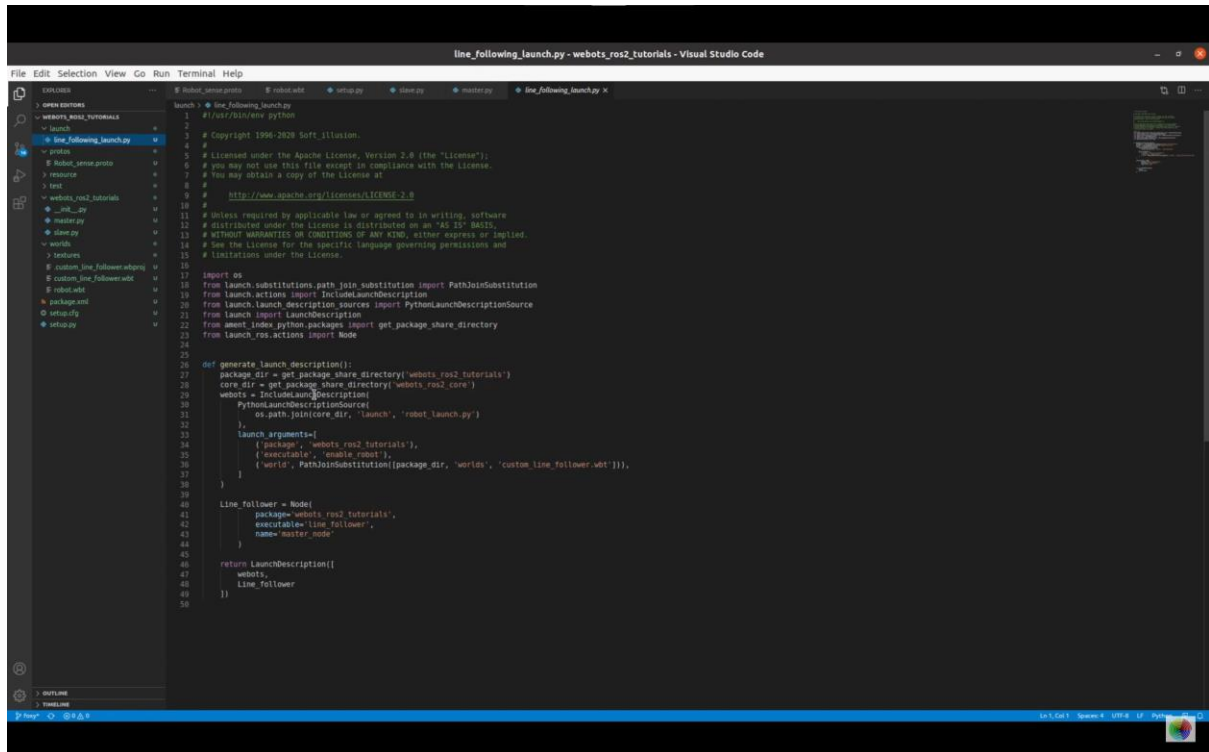


```
51
52
53     def LineFollowingModule(self):
54         # Constant velocity
55         self.cmd.linear.x = self.speed
56
57         # Correction parameters
58         self.GS_RIGHT = self.GS_RIGHT - self.GS_LEFT
59         self.GS_angular.z = self.angle_correction * self.DeltaS
60
61         # Logic for stop if black line not seen
62         if self.GS_RIGHT > 500 and self.GS_LEFT > 500 and self.GS_MID > 500:
63             self.count += 1
64         else:
65             self.count = 0
66
67         if self.count > self.count_threshold:
68             self.stop = True
69
70         if self.stop:
71             self.cmd.linear.x = 0.0
72             self.cmd.angular.z = 0.0
73
74         # Publish cmd vel
75         self.pubs_cmdvel.publish(self.cmd)
76         self.stop = False
77
78     # Call back to update sensor reading variables
79     def rightIR_cb(self, msg):
80         self.GS_RIGHT = msg.data
81         self.LineFollowingModule()
82
83     def leftIR_cb(self, msg):
84         self.GS_LEFT = msg.data
85         self.LineFollowingModule()
86
87     def midIR_cb(self, msg):
88         self.GS_MID = msg.data
89         self.LineFollowingModule()
```

Setup Project

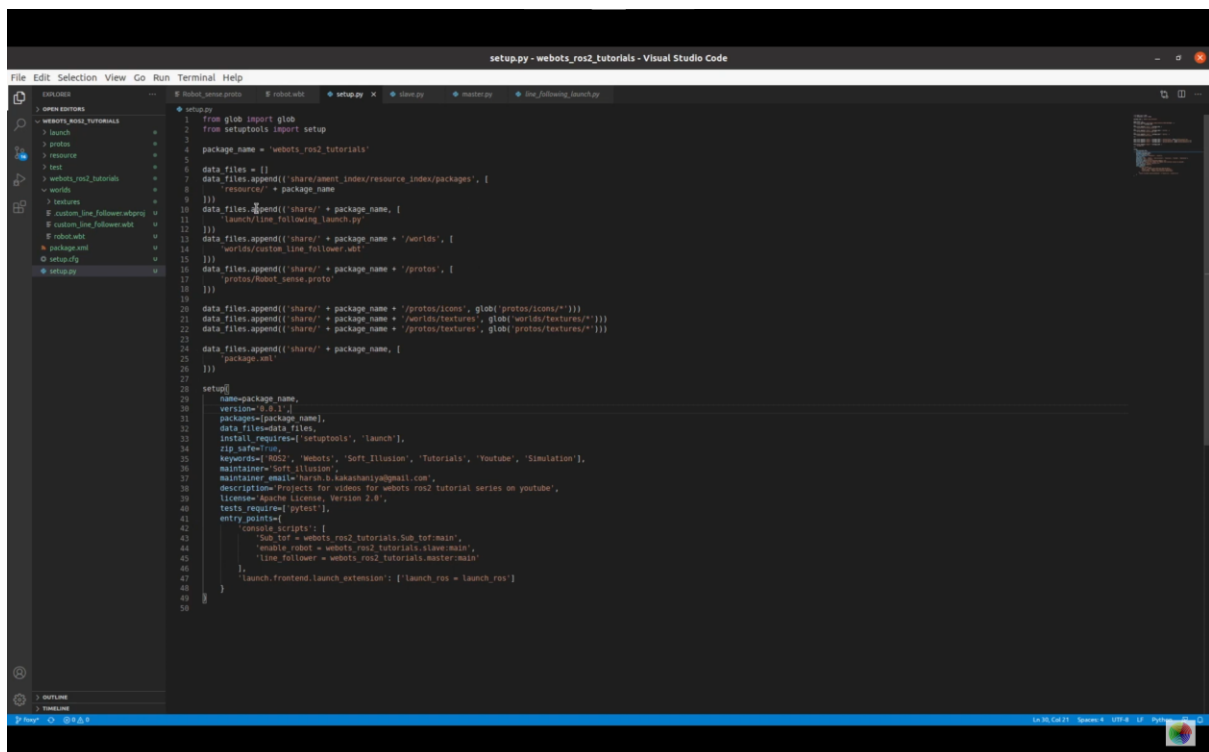
Pengaturan proyek ROS2, termasuk pembuatan package, node, dan konfigurasi komunikasi antar node.

line_following_launch.py



```
1 #!/usr/bin/env python
2
3 # Copyright 1996-2020 Soft Illusion.
4
5 # Licensed under the Apache License, Version 2.0 (the "License");
6 # you may not use this file except in compliance with the License.
7 # You may obtain a copy of the License at
8 #
9 # http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 # See the License for the specific language governing permissions and
15 # limitations under the License.
16
17 import os
18 from launch.substitutions.path_join_substitution import PathJoinSubstitution
19 from launch.actions import IncludeLaunchDescription
20 from launch.launch_description_sources import PythonLaunchDescriptionSource
21 from launch import LaunchDescription
22 from ament_index_python.packages import get_package_share_directory
23 from launch_ros.actions import Node
24
25
26 def generate_launch_description():
27     package_dir = get_package_share_directory('webots_ros2_tutorials')
28     core_dir = get_package_share_directory('webots_ros2_core')
29     webots = IncludeLaunchDescription(
30         PythonLaunchDescriptionSource(
31             os.path.join(core_dir, 'launch', 'robot_launch.py')
32         ),
33         launch_arguments=[
34             ('package', 'webots_ros2_tutorials'),
35             ('executable', 'enable_robot'),
36             ('world', PathJoinSubstitution([package_dir, 'worlds', 'custom_line_follower.world'])),
37         ],
38     )
39     Line_follower = Node(
40         package='webots_ros2_tutorials',
41         executable='line_follower',
42         name='master_node'
43     )
44
45     return LaunchDescription([
46         webots,
47         Line_follower
48     ])
49
50
```

setup.py



```
1 from glob import glob
2 from setuptools import setup
3
4 package_name = 'webots_ros2_tutorials'
5
6 data_files = []
7 data_files.append([share/ament_index/resource_index/packages', [
8     'resource' + package_name
9 ]])
10 data_files.append([share/' + package_name, [
11     'launch/line_following_launch.py'
12 ]])
13 data_files.append([share/' + package_name + '/worlds', [
14     'worlds/custom_line_follower.world'
15 ]])
16 data_files.append([share/' + package_name + '/protos', [
17     'protos/robot_sense.proto'
18 ]])
19
20 data_files.append([share/' + package_name + '/protos/icons', glob('protos/icons/*')])
21 data_files.append([share/' + package_name + '/worlds/textures', glob('worlds/textures/*')])
22 data_files.append([share/' + package_name + '/protos/textures', glob('protos/textures/*')])
23
24 data_files.append([share/' + package_name, [
25     'package.xml'
26 ]])
27
28 setup(
29     name=package_name,
30     version='0.0.1',
31     packages=[package_name],
32     data_files=data_files,
33     install_requires=['setuptools', 'launch'],
34     zip_safe=False,
35     keywords=['ROS2', 'Webots', 'Soft Illusion', 'Tutorials', 'Youtube', 'Simulation'],
36     maintainer='Soft Illusion',
37     maintainer_email='harsh.b.kanashetty@gmail.com',
38     description='Projects for videos for webots ros2 tutorial series on youtube',
39     license='Apache License, Version 2.0',
40     tests_require=['pytest'],
41     entry_points={
42         'console_scripts': [
43             'line_follower = webots_ros2_tutorials.sub_tof:main',
44             'enable_robot = webots_ros2_tutorials.slave:main',
45             'line_follower = webots_ros2_tutorials.master:main'
46         ],
47         'launch.frontend.launch_extension': ['launch_ros = launch_ros']
48     },
49 )
50
```

Build and Demo

Membangun proyek dan menjalankan simulasi robot. Demonstrasi cara mengontrol robot mengikuti garis menggunakan keyboard.

