

## Dokumentasi Lecture 4

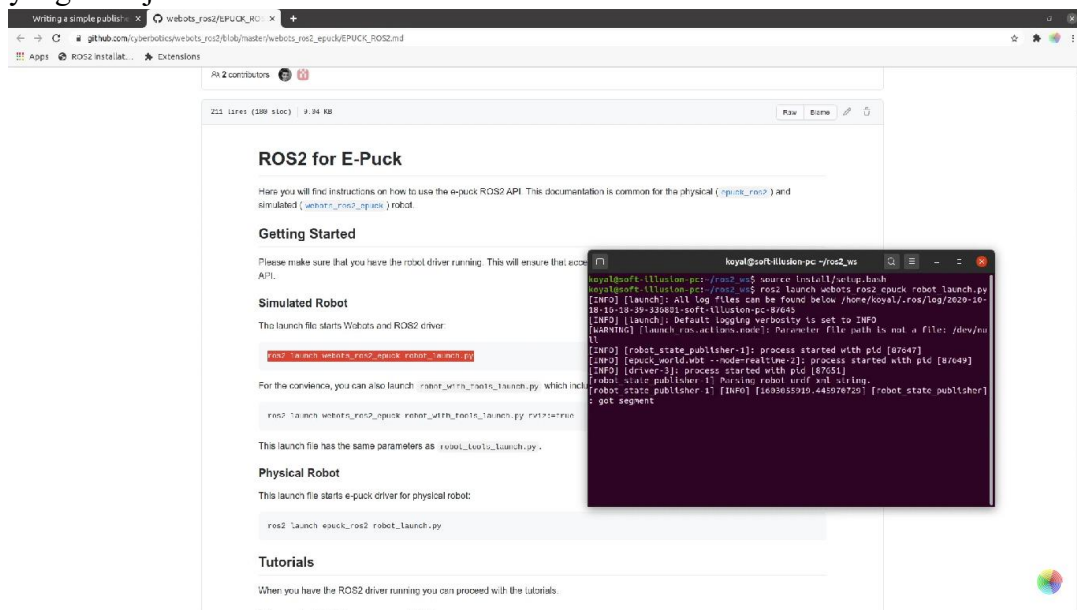
### Playlist Video:

[https://www.youtube.com/watch?v=mUmOwr-U\\_68&list=PLt69C9MnPchkP0ZXZOqmlIGRTOch8o9GiQ&index=6](https://www.youtube.com/watch?v=mUmOwr-U_68&list=PLt69C9MnPchkP0ZXZOqmlIGRTOch8o9GiQ&index=6)

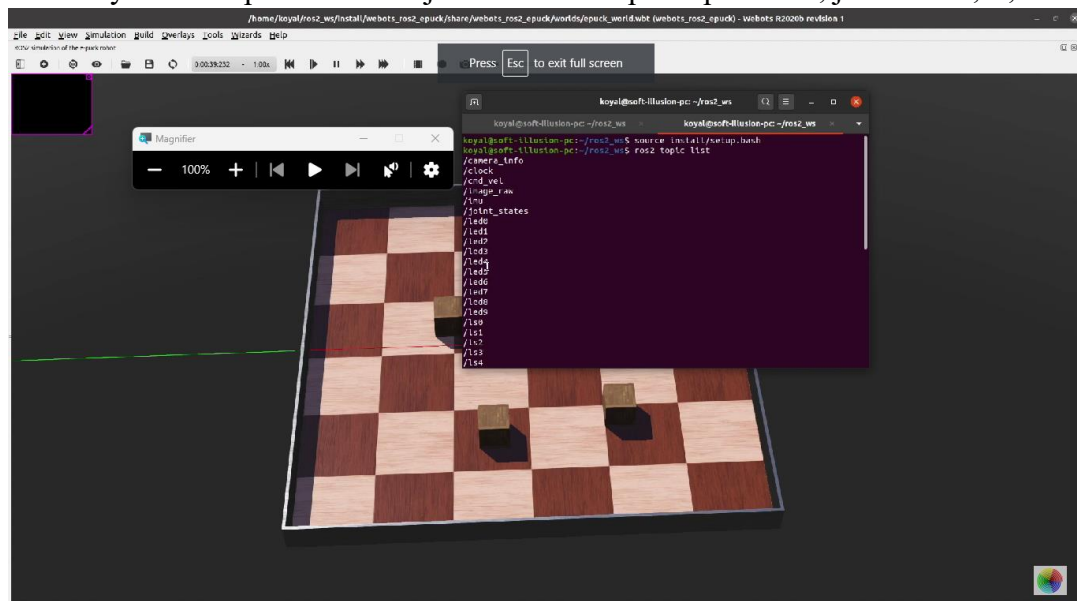
## Video 4 - Different examples in Webots with ROS2 | Webots ROS2 tutorial series | [Tutorial 2]

Video 4 terbagi dalam beberapa sub, antara lain:

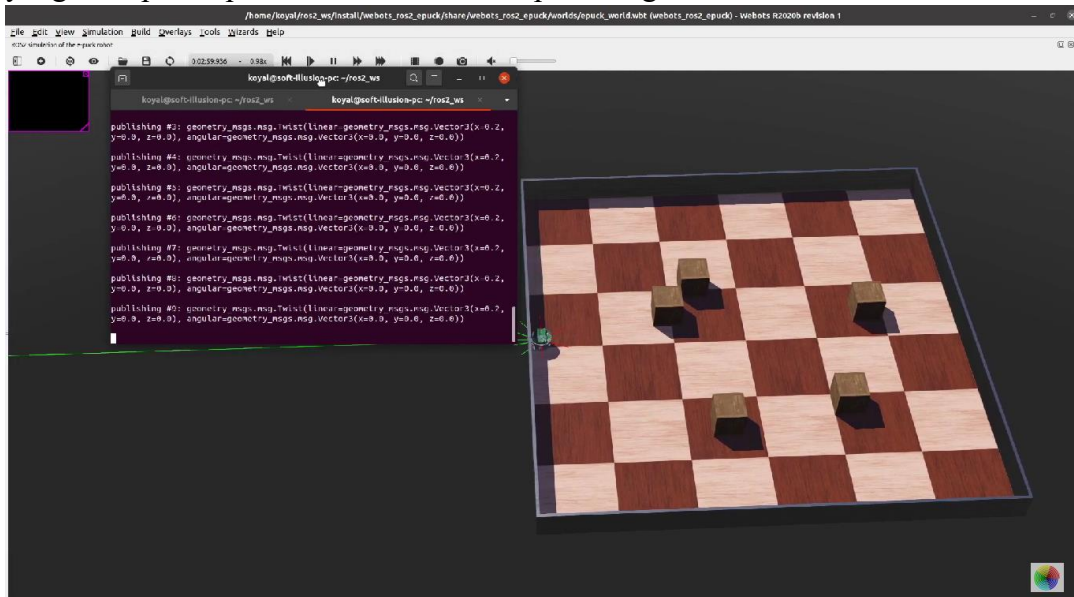
- *Publisher with Terminal*, Sebuah contoh digunakan untuk menjelaskan ini dari tautan ini: <https://github.com/cyberbotics/webots> Instruksi pada tautan ini dapat diikuti seperti yang ditunjukkan dalam video. Tetapi sebelum melanjutkan, pastikan repo ROS2 diatur seperti yang ditunjukkan dalam Video Tutorial 1.



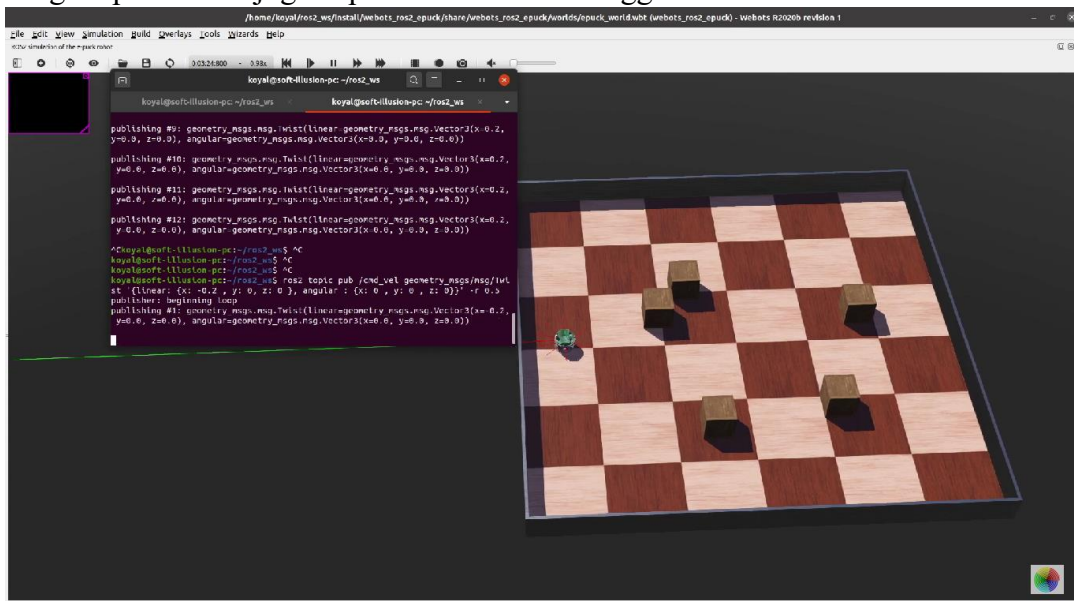
Scene robot E-Puck diluncurkan dengan file peluncuran dengan beberapa rintangan di sekitarnya. ros2 topic list menunjukkan daftar topik seperti /led, joint states, tf, dll.



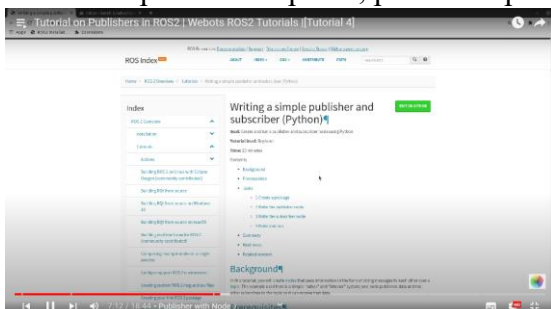
Sebuah contoh ditunjukkan dengan memublikasikan kecepatan ke topik `/cmd_vel` menggunakan `ros2 topic pub /cmd_vel.....` menggunakan kamus dalam arah x, y, dan z yang merupakan pesan twist beserta kecepatan angular.



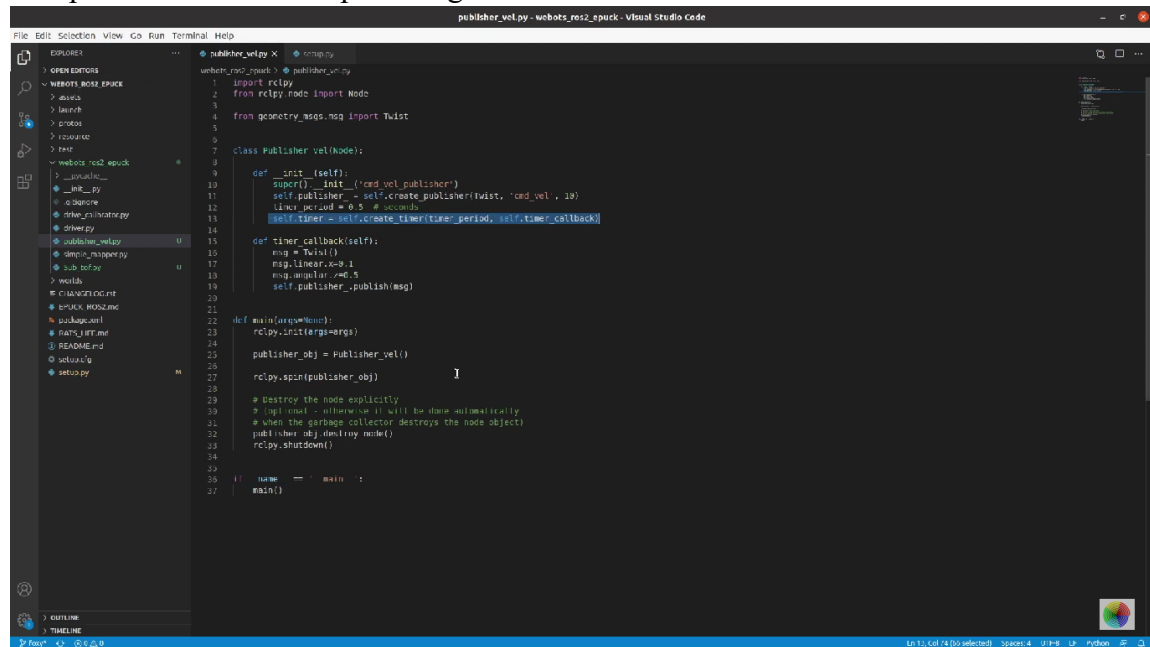
Tingkat penerbitan juga dapat dimodifikasi menggunakan ekstensi `-r`.



- *Publisher with Node* (script python), Dokumen ini dapat digunakan untuk memahami cara menulis sebuah node sederhana: <https://index.ros.org/doc/ros2/Tutorial>. Dokumen ini memandu pembuatan paket, penulisan penerbit, dan menjalankannya.



Namun, saat ini dokumentasi ini sudah tidak ada. Video menjelaskan kode mulai dari pernyataan impor hingga fungsi kelas. create\_publisher adalah fungsi di dalam rclpy yang memungkinkan pembuatan penerbit. Ini mengambil tipe pesan, nama topik, dan ukuran buffer. Video juga menjelaskan cara menggunakan timer. Dan berdasarkan timer, video menunjukkan cara terus-menerus memublikasikan pesan Twist yang memublikasikan kecepatan linear dan kecepatan angular.

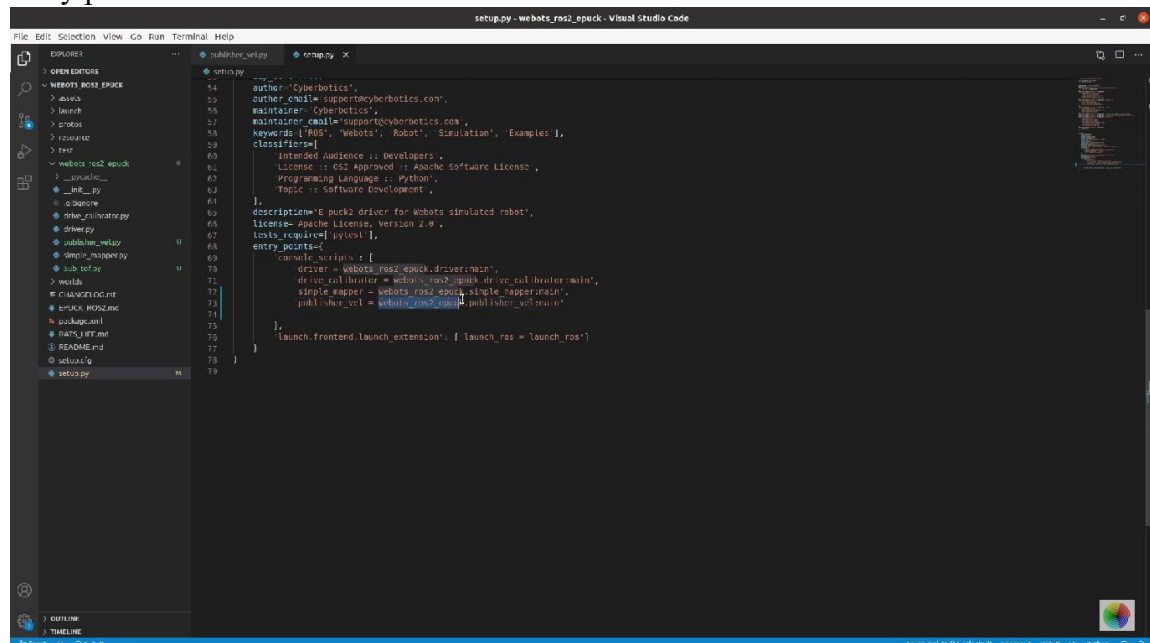


```

1  import rclpy
2  from rclpy.node import Node
3
4  from geometry_msgs.msg import Twist
5
6
7  class PublisherVel(Node):
8
9      def __init__(self):
10         super().__init__('cmd_vel_publisher')
11         self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
12         timer_period = 0.5 # seconds
13         self.timer = self.create_timer(timer_period, self.timer_callback)
14
15     def timer_callback(self):
16         msg = Twist()
17         msg.linear.x = 1
18         msg.angular.z = 0.5
19         self.publisher_.publish(msg)
20
21
22 def main(args=None):
23     rclpy.init(args=args)
24
25     publisher_obj = PublisherVel()
26
27     rclpy.spin(publisher_obj)
28
29     # Destroy the node explicitly
30     # (optional - otherwise it will be done automatically
31     # when the garbage collector destroys the node object)
32     publisher_obj.destroy_node()
33     rclpy.shutdown()
34
35
36 if __name__ == '__main__':
37     main()

```

Berkas dengan nama node dan struktur folder kemudian dimasukkan dalam setup.py pada entry point.

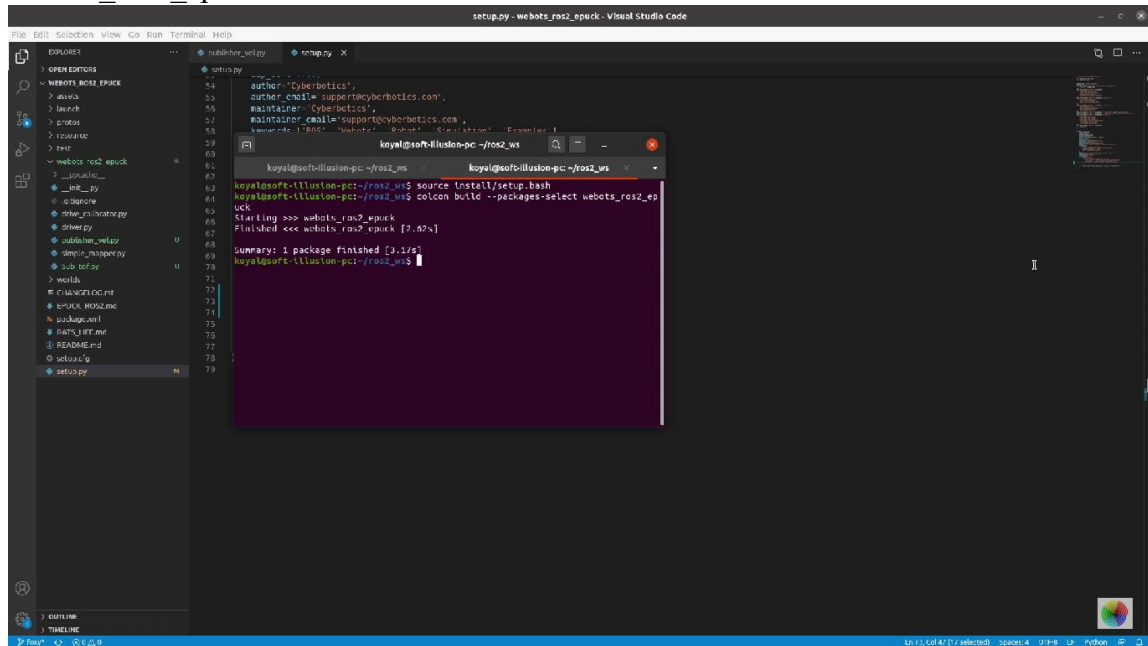


```

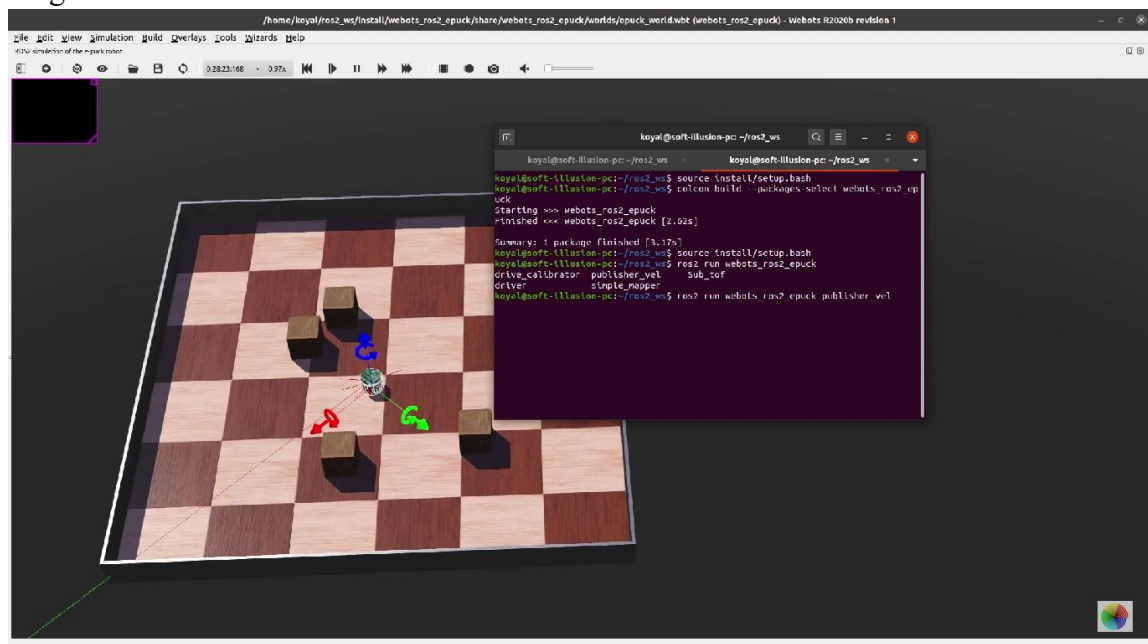
54  """
55  author: 'Cyberbotics',
56  author_email: 'support@cyberbotics.com',
57  maintainer: 'Cyberbotics',
58  maintainer_email: 'support@cyberbotics.com',
59  keywords: ['ROS', 'Webots', 'Robot', 'Simulation', 'Examples'],
60  classifiers=[
61      'Intended Audience :: Developers',
62      'License :: OSI Approved :: Apache Software License',
63      'Programming Language :: Python',
64      'Topic :: Software Development',
65  ],
66  description='E_puck2 driver for Webots simulated robot',
67  license='Apache License, Version 2.0',
68  tests_require=['pytest'],
69  entry_points={
70      'console_scripts': [
71          'driver = webots_ros2_epuck.driver:main',
72          'simple_mapper = webots_ros2_epuck.simple_mapper:main',
73          'publisher_vel = webots_ros2_epuck.publisher_vel:main'
74      ],
75      'launch.frontend.launch_extension': ['launch_ros = 'launch_ros']
76  }
77
78
79

```

Kemudian dibangun paket ini menggunakan `colcon build --packages-select webots_ros2_epuck`.



Setelah sumber ketika menjalankan paket lagi dengan nama node, dapat terlihat robot bergerak.





- Perbedaan ROS1 dan ROS2 *Publisher*

ROS Publisher	VS	ROS 2 Publisher
The duration and time types are defined in the client libraries, they are in C++ and Python.		In ROS2 these types are defined as messages and therefore are consistent across languages.
Nodes cannot survive without roscore.		Nodes can survive without roscore. As it is end to end connection.
ROS uses a centralised discovery.(Single point of failure )		ROS 2 is fully distributed including discovery.
ROS uses message definition and serialization.(DSS-RPC)		ROS2 is DDS vendor independent. (e.g default FastRTPS can be changed)
ROS can not fulfill real-time requirements so ROS is not mostly used in production scenario.		We can get Real-time system because of we can tweak DDS system configurations (QoS) according to the application.
ROS can only use old user-facing APIs (eg 0.4 "Mango Tango") which was released in 2009		ROS2 is build by considering latest API's.

- Aplikasi dari *publishers* di ROS2

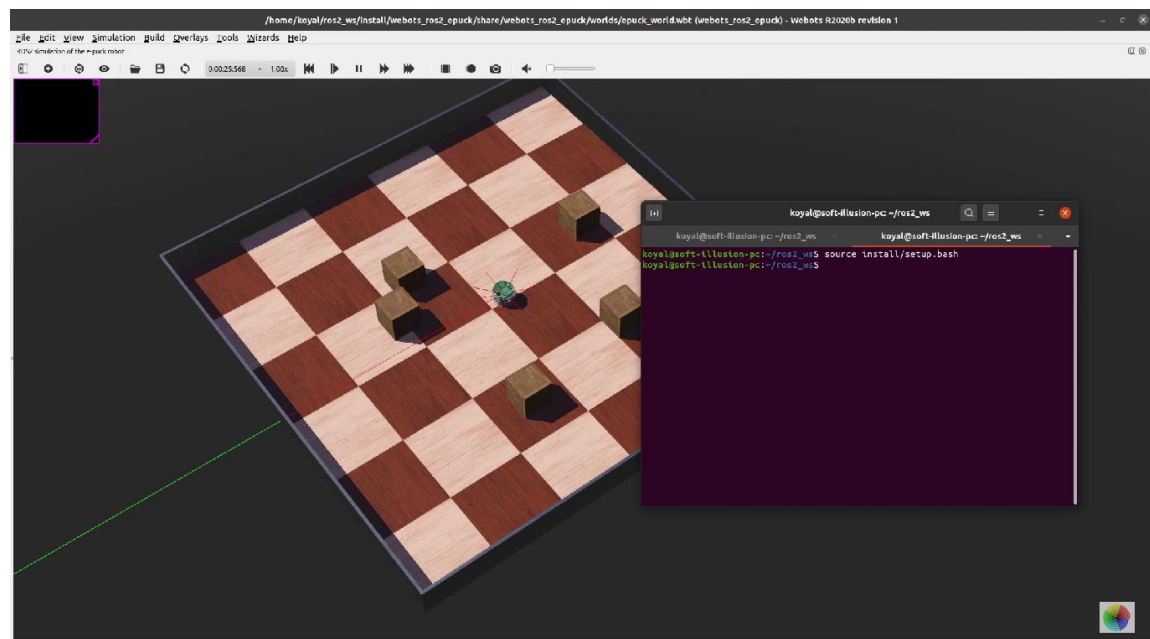
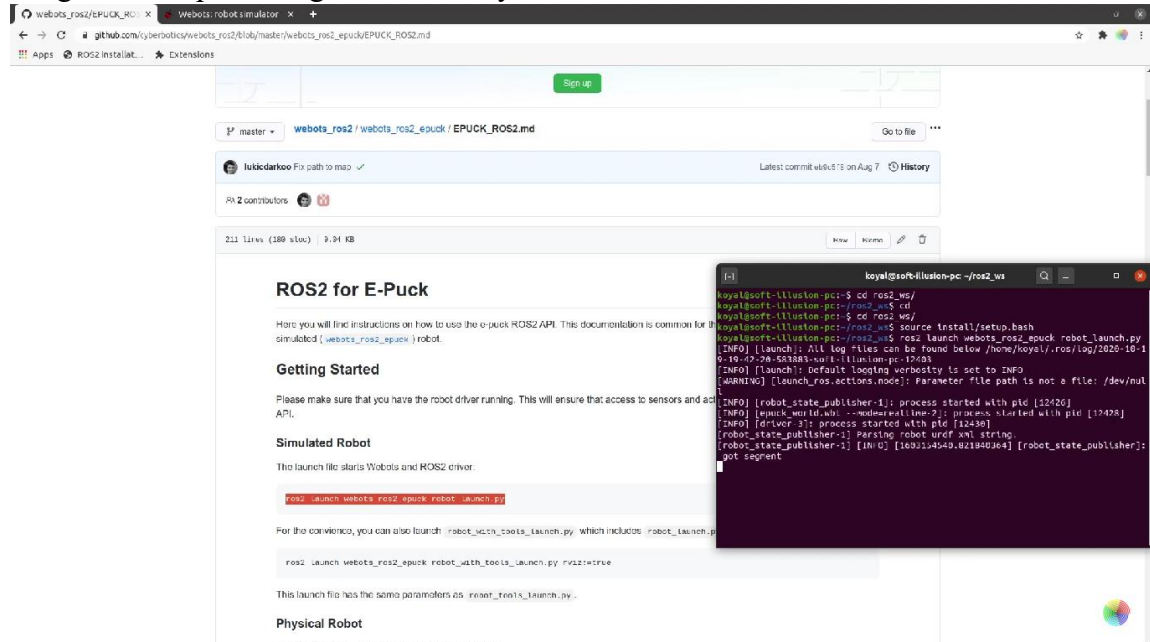
## Applications of publisher in ROS 2

- To control the system real time.
- To inform system about updated environment.
- Internal communication of the system .
- To control the actuators position and velocity for their motion.
- To perform action in sequence.
- To handle emergency stopping situation.

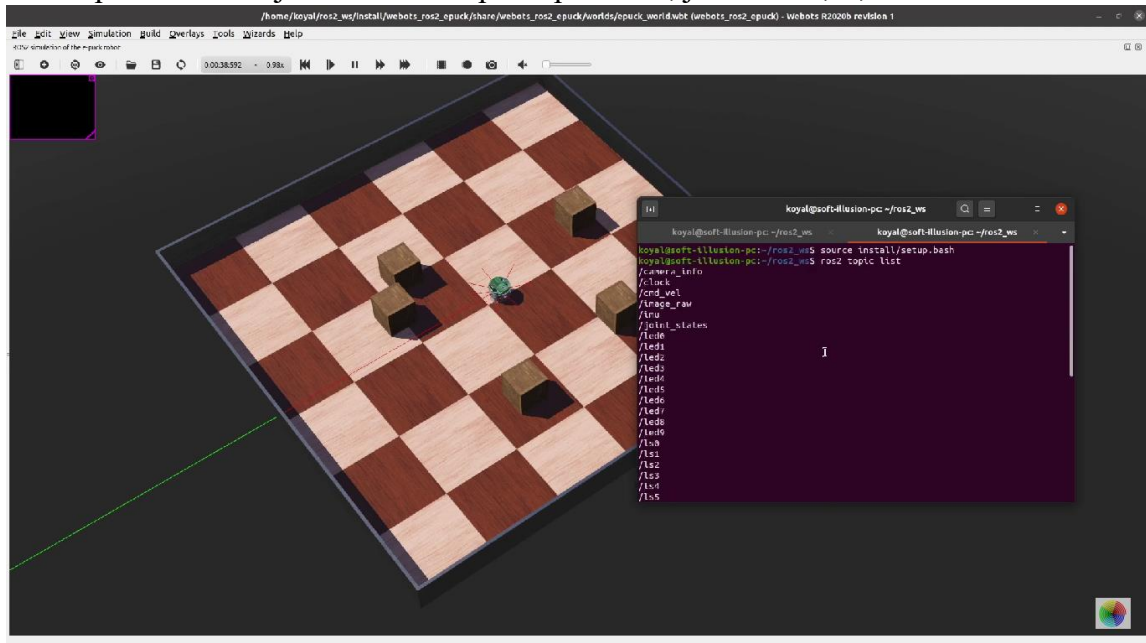
## Video 5 - Using ROS2 Services to interact with Webots | Webots ROS2 tutorial series [Tutorial 3]

Video 5 terbagi dalam beberapa sub, antara lain:

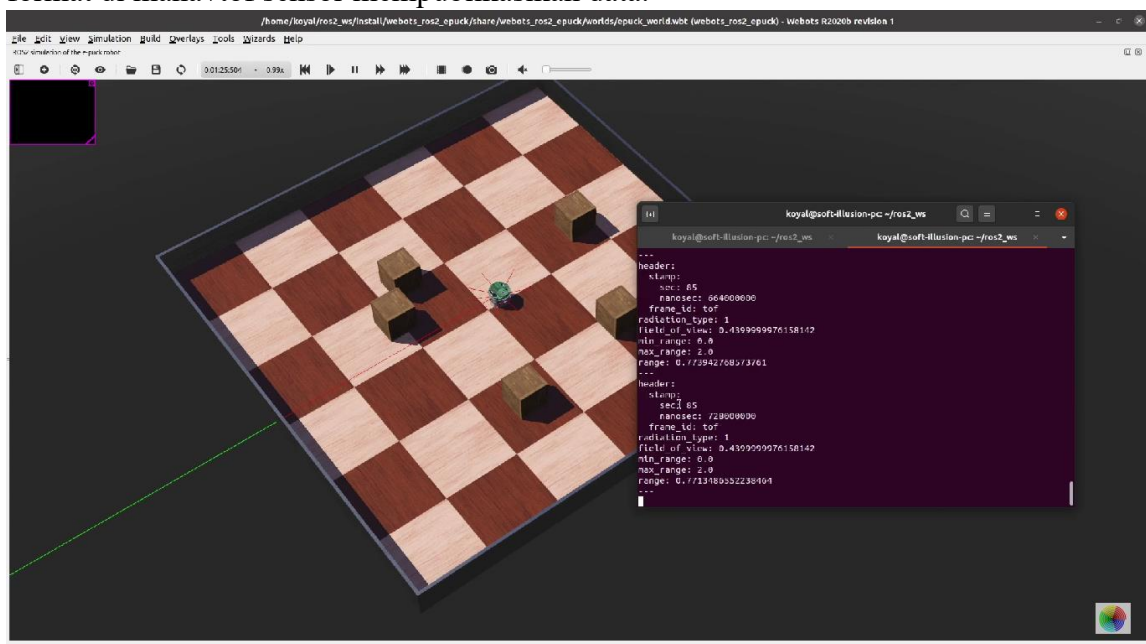
- *Subscriber with Terminal*, video pertama robot E-Puck diluncurkan dengan file peluncuran dengan beberapa rintangan di sekitarnya.



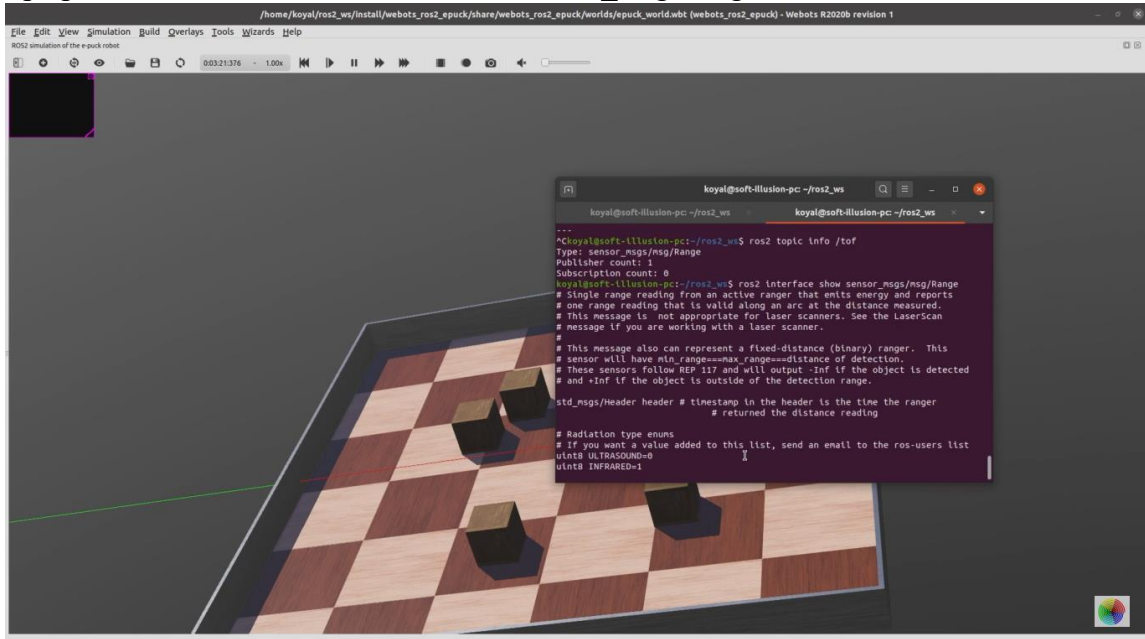
ros2 topic list menunjukkan daftar topik seperti /led, joint states, tf, dll.



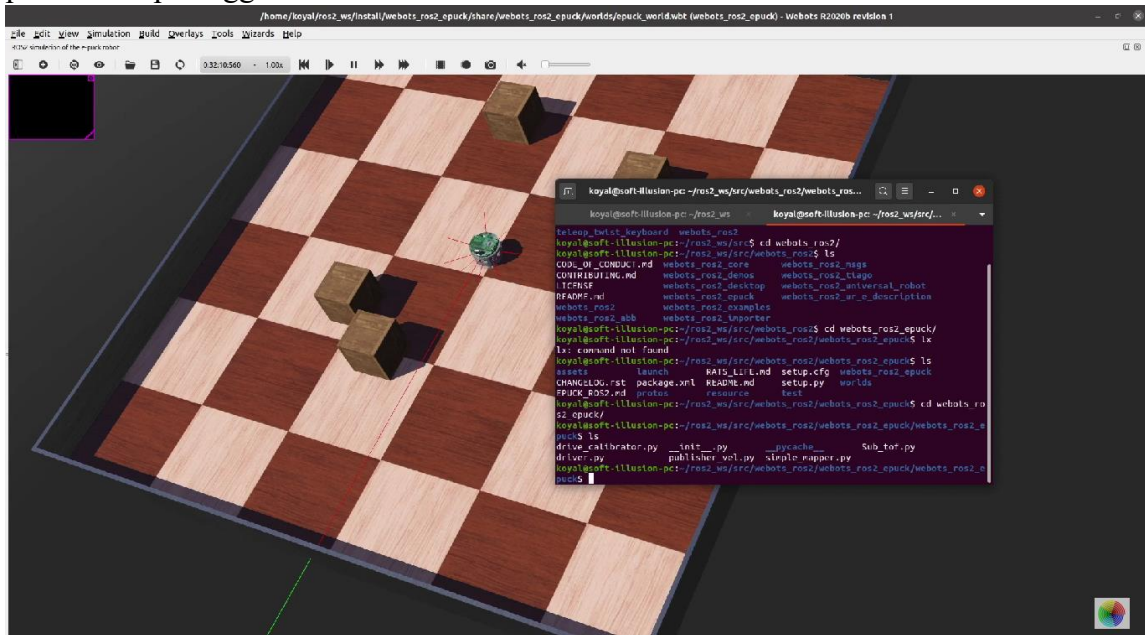
Contoh ditunjukkan dengan berlangganan ke topik sensor /tof yang merupakan garis jarak jauh seperti yang terlihat di dunia. `ros2 topic echo /tof` menunjukkan nilai dari balok dalam format di mana /tof sensor mempublikasikan data.



Tipe pesan dapat diketahui menggunakan `ros2 topic info /tof`. Untuk melihat seperti apa tipe pesan tersebut: `ros2 interface show sensor_msgs/msg`.

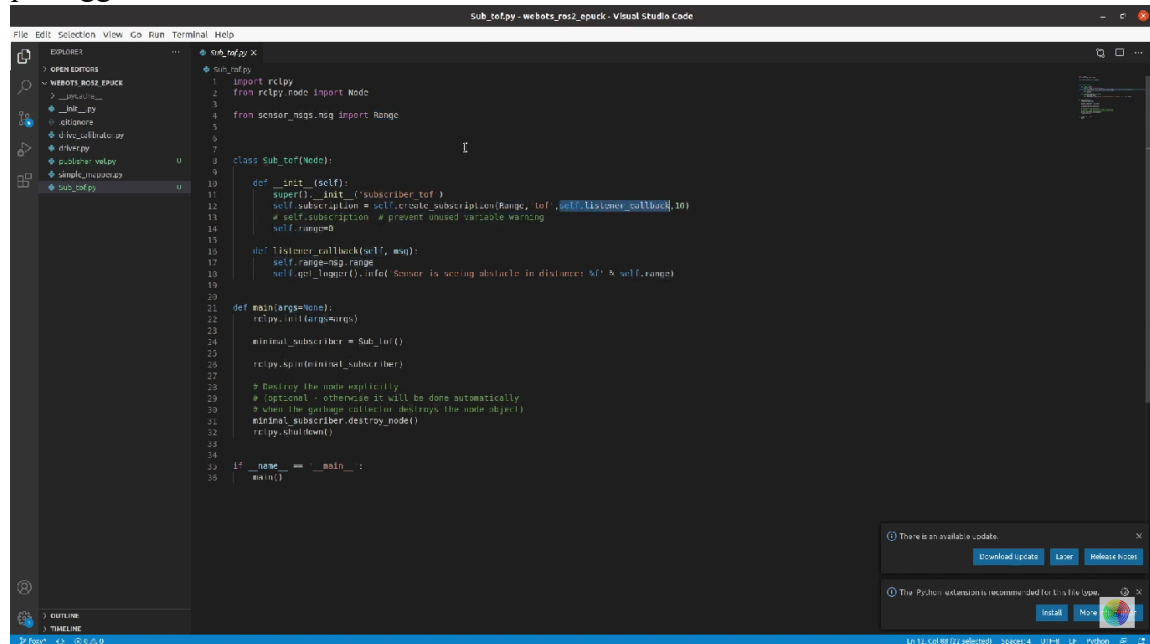


- *Subscribe with Node*, Dokumen ini memandu pembuatan paket, penulisan pelanggan, dan menjalankannya. "create\_subscription" adalah fungsi di dalam `rcpp` yang memungkinkan pembuatan pelanggan.





Ini mengambil tipe pesan, nama topik, fungsi panggilan kembali (callback function), dan ukuran buffer. Fungsi panggilan kembali dipanggil setiap kali ada pesan di topik pelanggan.

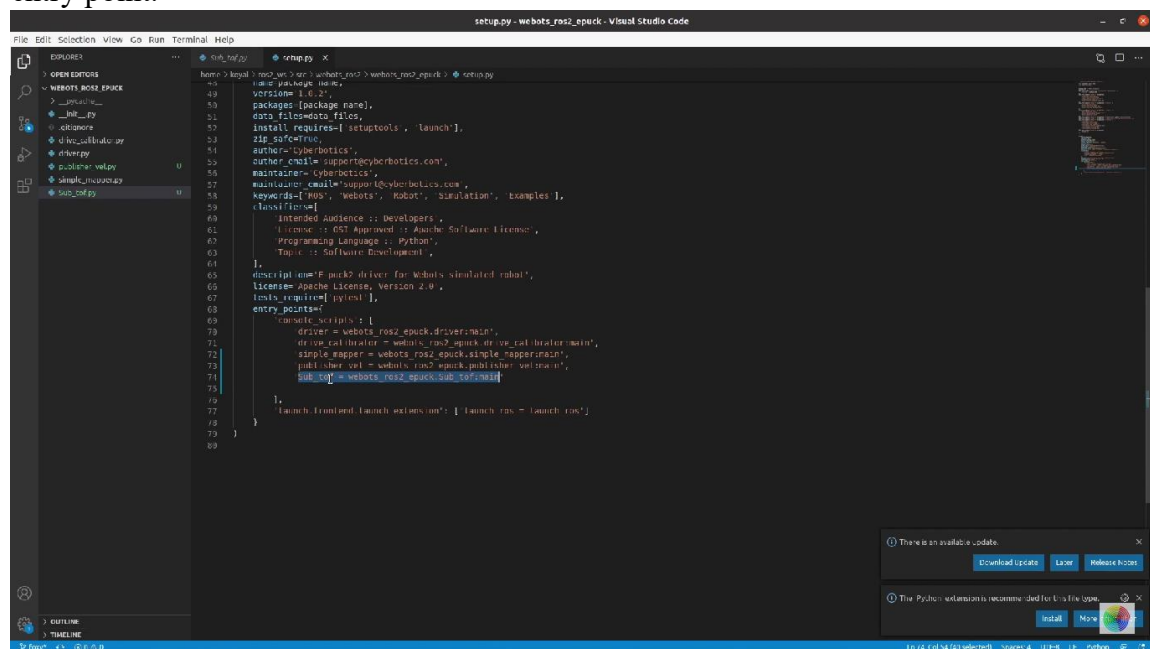


```

1 import rclpy
2 from rclpy.node import Node
3
4 from sensor_msgs.msg import Range
5
6
7
8
9
10
11 class Sub_tf(Node):
12     def __init__(self):
13         super().__init__('subscriber_tf')
14         self.subscription = self.create_subscription(Range, 'tf', self.listener_callback, 10)
15         # self.subscription # prevent unused variable warning
16         self.range=0
17
18     def listener_callback(self, msg):
19         self.range=msg.range
20         self.get_logger().info('Sensor is seeing obstacle in distance: %d' % self.range)
21
22
23
24 def main(args=None):
25     rclpy.init(args=args)
26
27     minimal_subscriber = Sub_tf()
28
29     rclpy.spin(minimal_subscriber)
30
31     # Destroy the node explicitly
32     # (optional - otherwise it will be done automatically
33     # when the garbage collection destroys the node object)
34     minimal_subscriber.destroy_node()
35     rclpy.shutdown()
36
37
38 if __name__ == '__main__':
39     main()

```

Berkas dengan nama node dan struktur folder kemudian dimasukkan dalam setup.py pada entry point.

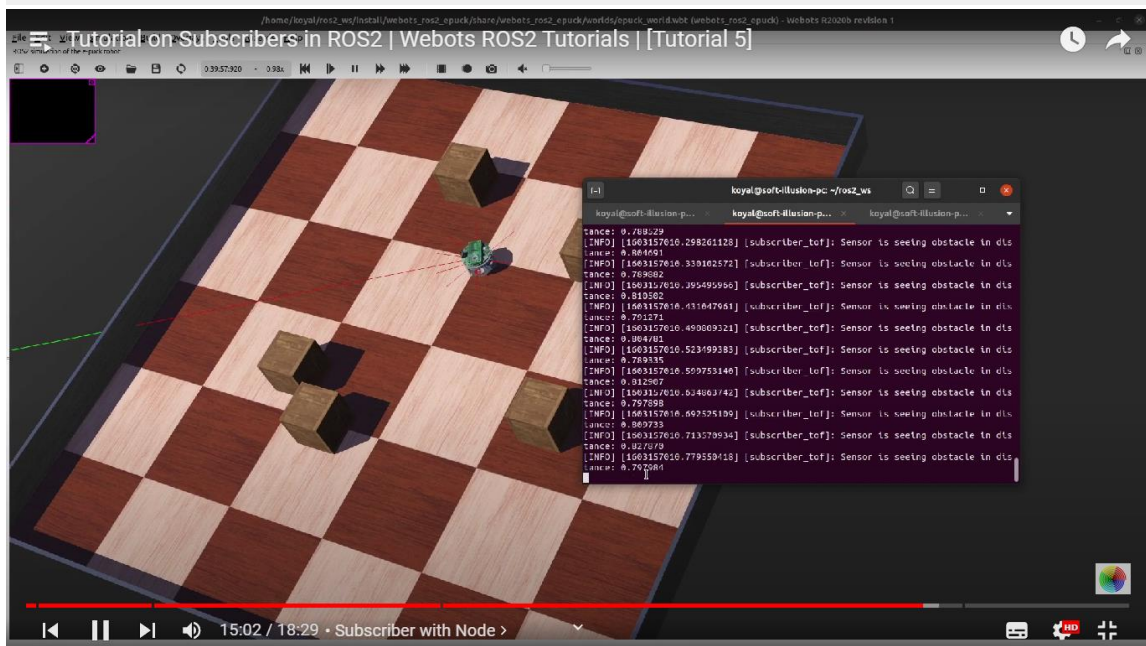
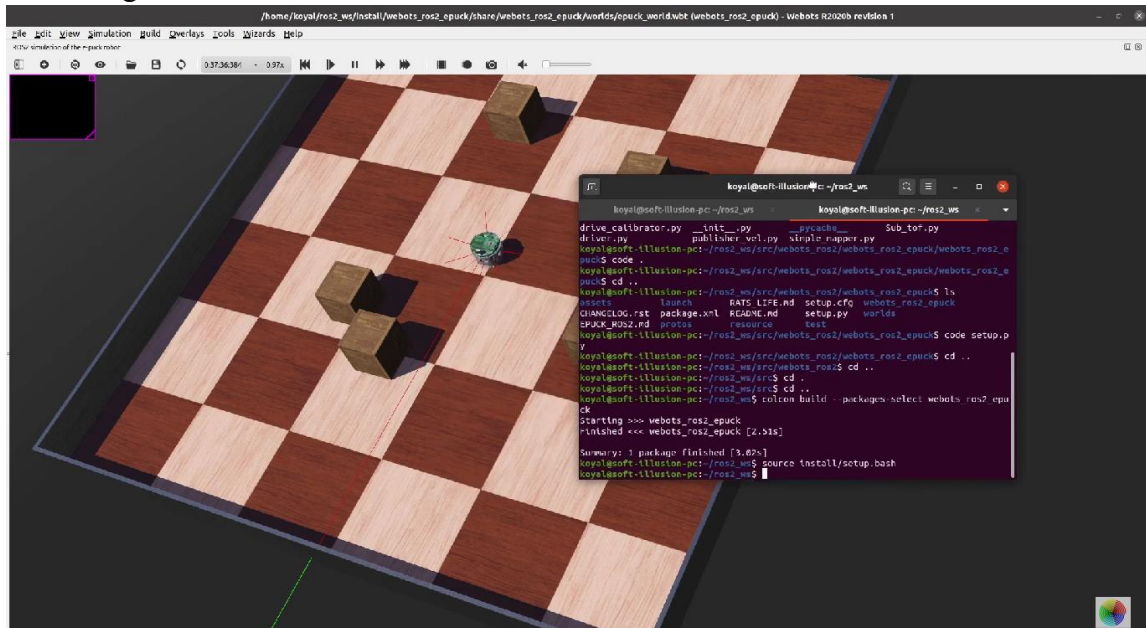


```

45 from setuptools import setup
46
47 packages=[package_name],
48 version='1.0.0',
49 data_files=[('data_files', launch)],
50 zip_safe=False,
51 author='Cyberbotics',
52 author_email='support@cyberbotics.com',
53 maintainer='Cyberbotics',
54 maintainer_email='support@cyberbotics.com',
55 keywords=['ROS', 'Robot', 'Simulation', 'Examples'],
56 classifiers=[
57     'Intended Audience :: Developers',
58     'License :: OSI Approved :: Apache Software License',
59     'Programming Language :: Python',
60     'Topic :: Software Development',
61 ],
62 description='E rack2 driver for Webots simulated robot',
63 license='Apache License, Version 2.0',
64 tests_require=['pytest'],
65 entry_points={
66     'console_scripts': [
67         'driver = webots_ros2_epuck.driver.main',
68         'simple_mapper = webots_ros2_epuck.simple_mapper.main',
69         'sub_tf = webots_ros2_epuck.sub_tf.main'
70     ],
71 },
72 launch_files=[launch_files]
73 }

```

Kemudian membangun paket ini menggunakan colcon build --packages-select webots\_ros2\_epuck. Setelah itu dapat menjalankan paket lagi dengan nama node, terlihat robot bergerak.



- Perbedaan antara ROS dan ROS2 *subscribers*

ROS Subscriber	VS	ROS 2 Subscriber
ROS 1 uses TCP as the underlying transport, which is unsuitable for lossy networks such as wireless links.		With ROS 2 relying on DDS which uses UDP as its transport, we can give control over the level of reliability a node can expect and act accordingly.
No option of setting QoS policy.		We can set policy like history : [keep last , keep all]
It does not allow nodes to be restarted online.		It will also allow nodes to be restarted or replaced on-line.
ROS offers a message passing interface that provides inter-process communication and is commonly referred to as a middleware.		ROS2 uses a DDS (Data Distribution Service). The main advantage is transmission performances will be improved. 