# Modul Tambahan

## ROS 2 : Setup

Sebelum dapat menggunakan ROS2 dalam terminal, directory setup.bash dari ROS2 harus di source di dalam terminal dari workspace. Untuk men-source setup.bash, dapat dilakukan command :

source /opt/ros/humble/setup.bash

Untuk menghindari harus melakukan command tersebut setiap kali terminal dibuka, dapat mengubah bashrc (file config yang dijalankan setiap kali user menyalakan terminal) dengan menggunakan command:

gedit ~/.bashrc

dan menambahkan exerp code ini kepada akhir file bash:

source /opt/ros/humble/setup.bash

ROS2 memiliki fitur untuk autocomplete command yang Anda sedang ketik, namun fitur ini menggunakan extension ‘argcomplete’ untuk dapat dilakukan. Command yang dapat dilakukan untuk download extension tersebut :

sudo apt install python3-argcomplete

ROS2 seringkali harus menggunakan 2 atau lebih terminal, untuk memudahkan user dalam me-manage sejumlah terminal ini, dapat digunakan aplikasi external terminal yang bernama “Terminator”. Command yang dapat dijalankan untuk meng-install Terminator adalah:

sudo apt install terminator

Untuk membuat program ROS2 sendiri, kita perlu alat untuk mem-build program kita. Alat yang dapat digunakan untuk build ini adalah “colcon”. Untuk meng-install colcon, dapat digunakan command ini:

sudo apt install colcon

Colcon sendiri juga memiliki fitur untuk auto complete seperti ROS2, dan harus di-initialize setiap kali terminal baru dipanggil. Maka, sebaiknya fitur ini di source langsung di bashrc. Lakukan command:

gedit ~/.bashrc

dan tambahkan:

source /usr/share/colcon\_argcomplete/hook/ colcon\_argcomplete.bash

## ROS 2 : Node dan Packages

Memisah-misahkan masalah yang kompleks menjadi masalah yang kecil adalah aspek yang penting dalam penyelesaian masalah dalam robotika, tidak hanya masalah menjadi lebih mudah diselesaikan, namun pembagian tugas antar anggota semakin mudah. Salah satu cara ROS dapat memudahkan kita melakukan pemecahan masalah itu adalah dengan memudahkan kita untuk membuat “Package” yang dapat menyelesaikan suatu masalah tertentu. Package-package ini kemudian dapat diintegrasikan dengan bantuan ROS2.

Untuk bagian ini, modul akan dikhususkan untuk praktek. Sehingga untuk penjelasan ulang untuk materi dapat diakses pada link berikut:

1. Node : <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>
2. Topics : <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html>
3. Services : <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html>
4. Parameter : <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Parameters/Understanding-ROS2-Parameters.html>

### Langkah 1 : Membuat workspace

Pertama-tama, buatlah folder workspace dan move ke dalam folder tersebut :

mkdir [NAMA FOLDER WORKSPACE KALIAN]

cd [NAMA FOLDER WORKSPACE KALIAN]

Lalu, buatlah folder src di dalam folder workspace kalian :

mkdir src

dan build workspace kalian :

colcon build

ada baiknya kalian source workspace kalian :

gedit ~/.bashrc

source ~/[NAMA FOLDER WORKSPACE KALIAN]/install /setup.bash

### Langkah 2 : Membuat package

Langkah selanjutnya adalah untuk memindahkan diri ke folder src pada workspace kalian:

cd src

Lalu, tergantung bahasa pemrograman yang ingin dipakai dalam package, package dapat dibuat dengan command :

Untuk Python :

ros2 pkg create [NAMA PACKAGE] –build-type ament\_python –dependencies rclpy

Untuk C++ :

ros2 pkg create [NAMA PACKAGE] –build-type ament\_cmake –dependencies rclcpp

Buka VSCode (atau coding software preferensi kalian), dan navigasi ke folder src kalian. Seharusnya jika pembuatan package berjalan dengan lancar, maka akan terlihat sebuah folder dengan nama package kalian.

### Langkah 3 : Membuat node

Walaupun package sudah terbentuk, Package masih belum memiliki fungsi apapun. Untuk memberikan fungsi-fungsi kepada package, maka package perlu ditambahkan node. Node berperan dalam package sebagai subprogram yang berfungsi untuk menyediakan 1 Fungsi.

**PYTHON**

Pertama, pindah ke package kalian dan buat node :

cd [NAMA PACKAGE]/[NAMA PACKAGE]

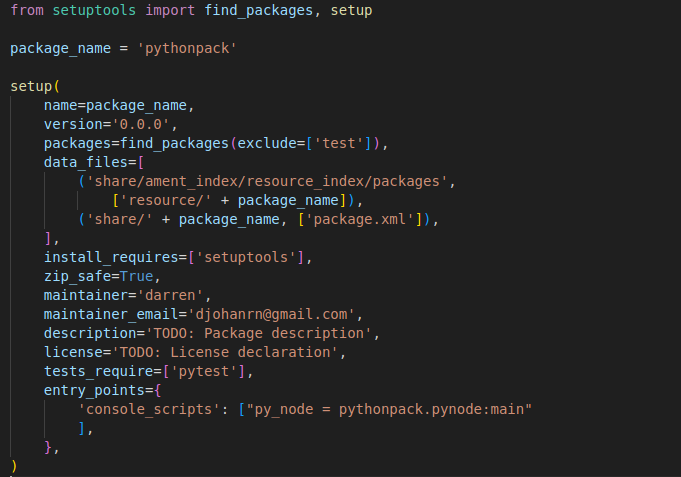
touch [Nama Node].py

Buka VSCode dan open node python yang baru kalian buat. Dalam file kosong tersebut, masukkan template kode (akan mencetak Hello World! ke terminal) berikut:

|  |
| --- |
| #!/usr/bin/env python3  import rclpy  from rclpy.node import Node  def main(args=None):  rclpy.init(args=args)  node = Node("pynode")  node.get\_logger().info("Hello World!")  rclpy.spin(node)  rclpy.shutdown()  if name == "main":  main() |

Lalu, pindah ke setup.py pada package kalian, dan tambahkan line ini ke bagian “entry\_points”:

“py\_node = [NAMA PACKAGE].[NAMA NODE(nama file python node]:main”



Bagan 1 : Setup.py setelah config

Kembali ke root workspace kalian, lalu build workspace kalian :

colcon build

Open terminal baru, source ulang bashrc, lalu run command berikut:

ros2 run [NAMA PACKAGE] py\_node

dan output dari ROS akan terlihat :



Bagan 2: output dari terminal

**C++ :**

Pertama, pindah ke package kalian dan buat node :

cd [NAMA PACKAGE]/[NAMA PACKAGE]

touch [Nama Node].cpp

Buka VSCode dan open node python yang baru kalian buat. Dalam file kosong tersebut, masukkan template kode (akan mencetak Hello World! ke terminal) berikut:

|  |
| --- |
| #include "rclcpp/rclcpp.hpp"  int main (int argc, char \*\*argv){  rclcpp::init(argc,argv);  auto node = std::make\_shared<rclcpp::Node>("cpppack\_node");  RCLCPP\_INFO(node->get\_logger(),"Hello World!");  rclcpp::spin(node);  return 0;  } |

include pada VSCode akan digolongkan error, namun tidak akan masalah bila colcon build. Untuk menghilangkan pada VSCode :

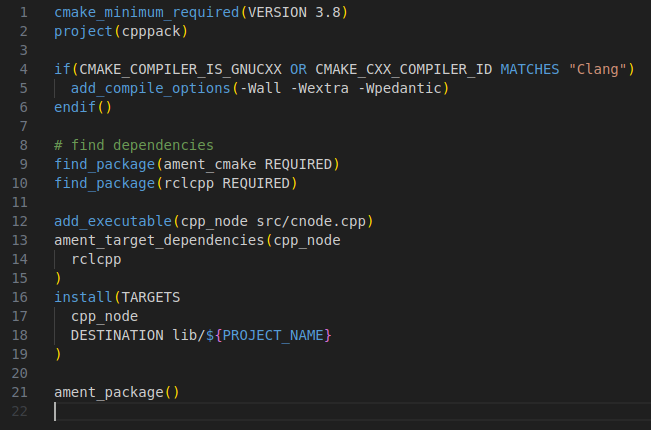
1. “Ctrl + Shift + P” untuk membuka file configuration vscode
2. Cari “C/C++ : Edit Configurations (JSON)” dan open
3. Pada JSON file yang terbuka, edit “includePath”: dan tambahkan “/opt/ros/DISTRO/include” pada include path. Berikut gambar untuk distro Foxy :



Bagan 3: Config JSON file

Sebelum program C++ dapat di-compile, CMakeList.txt harus diubah:

1. Hilangkan bagian #Default to C99, sampai bagian endif(); {4 Line dihapus}
2. Hilangkan bagian If(BUILD\_TESTING), sampai endif(); {10 Line dihapus}
3. Tambahkan bagian baru dibawah find dependencies:
   1. add\_executable(cpp\_node src/[NAMA NODE])
   2. ament\_target\_dependencies(cpp\_node rclcpp)
   3. install(TARGETS  
      cpp\_node  
      DESTINATION lib/${PROJECT\_NAME}  
      )



Bagan 4 : CmakeList.txt

Kembali ke root workspace kalian, lalu build workspace kalian :

colcon build

Open terminal baru, source ulang bashrc, lalu run command berikut:

ros2 run [NAMA PACKAGE] cpp\_node

dan output dari ROS akan terlihat :



Bagan 5 : Output dari Terminal

## ROS 2 : Tugas Opsional

Ikuti contoh yang sudah diterapkan pada modul, namun ubah output yang ditampilkan pada terminal menjadi “(NAMA ANDA)-(NIM ANDA)”.

1. Buat **1 Package saja**.
2. Pemilihan bahasa pemrograman dibebaskan.
3. File yang dikumpul berupa PDF yang berisi :
   1. Kendala yang dialami selama proses pembuatan (kalau tidak ada, cukup berikan “-”).
   2. Source code untuk node.py atau node.cpp Anda.
   3. Screenshot hasil output pada terminal.

📩 **Cara pengumpulan:**

Langsung kirimkan via DM Line ke @monetaryape450

⏰ **Deadline terakhir:**

Minggu, 1 Juni 2025, pukul 15.00 WIB

## ROS 2 : Materi lanjutan dan Sumber belajar

* Course Udemy ROS2:  
  <https://drive.google.com/drive/folders/1Ar85uglPw8Oulw8mEiXjiFhDq3EVdp3K?usp=drive_link>
* Tutorial ROS2 Humble:

<https://docs.ros.org/en/humble/Tutorials.html>