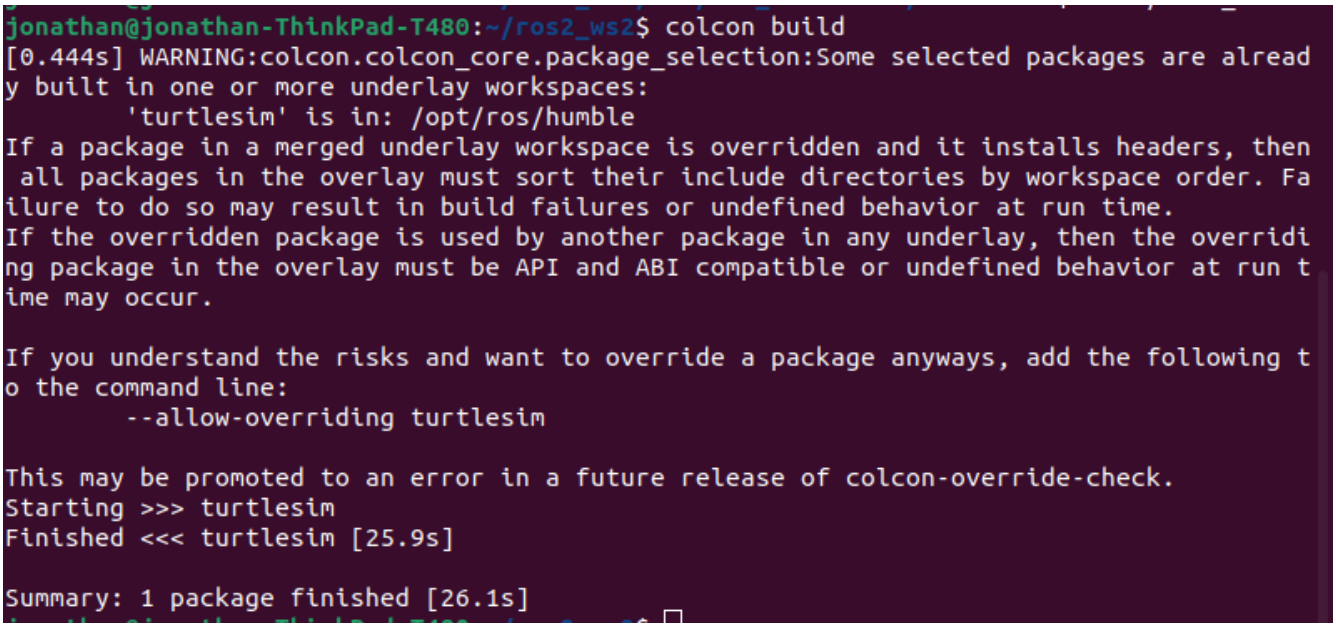
Ros 2 humble
Initially I have a problem on building the program using colcon, because of the difference between colcon and catkin. I look online for a help and they mentioned that I would have to remove one of the line in CmakeLists.txt in order for me to be able to compile this program



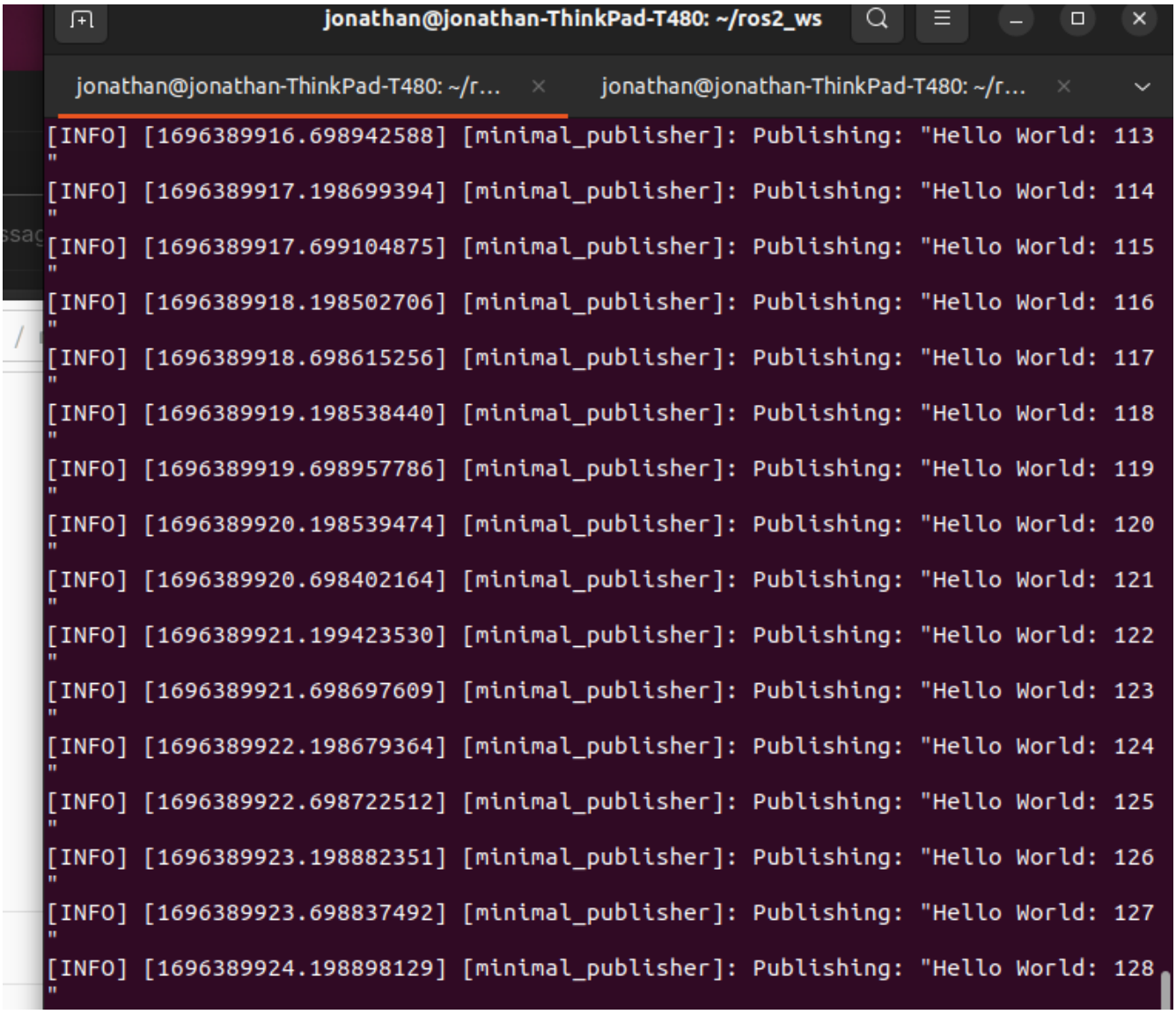
```
cmake_minimum_required(VERSION 3.5)
project(turtlesim)

if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

find_package(ament_cmake REQUIRED)
find_package(ament_index_cpp REQUIRED)
find_package(geometry_msgs REQUIRED)
find_package(Qt5 REQUIRED COMPONENTS Widgets)
find_package(rclcpp REQUIRED)
find_package(rclcpp_action REQUIRED)
#find_package(rosidl_default_generators REQUIRED)
find_package(std_msgs REQUIRED)
find_package(std_srvs REQUIRED)

include_directories(include ${Qt5Widgets_INCLUDE_DIRS})

rosidl_generate_interfaces(${PROJECT_NAME}
  "action/RotateAbsolute.action"
  "msg/Color.msg"
  "msg/Pose.msg"
  "srv/Kill.srv"
  "srv/SetPen.srv"
  "srv/Spawn.srv"
  "srv/TeleportAbsolute.srv"
  "srv/TeleportRelative.srv")

set(dependencies "ament_index_cpp" "geometry_msgs" "rclcpp" "rclcpp_action" "std_msgs">

set(turtlesim_node_SRCS
  src/turtlesim.cpp
  src/turtle.cpp
  src/turtle_frame.cpp
)
set(turtlesim_node_HDRS
  include/turtlesim/turtle_frame.h
)

qt5_wrap_cpp(turtlesim_node_MOCS ${turtlesim_node_HDRS})

rosidl_get_typesupport_target(cpp_typesupport_target "${PROJECT_NAME}" "rosidl_typesup>
                                  [ Read 74 lines ]
```

I have to comment out #find_package(rosidl_default_generators REQUIRED) in order for me to be able to do it.

```
jonathan@jonathan-ThinkPad-T480:~/ros2_ws2$ colcon build
[0.443s] WARNING:colcon.colcon_core.package_selection:Some selected packages are alread
y built in one or more underlay workspaces:
        'turtlesim' is in: /opt/ros/humble
If a package in a merged underlay workspace is overridden and it installs headers, then
 all packages in the overlay must sort their include directories by workspace order. Fa
ilure to do so may result in build failures or undefined behavior at run time.
If the overridden package is used by another package in any underlay, then the overridi
ng package in the overlay must be API and ABI compatible or undefined behavior at run t
ime may occur.

If you understand the risks and want to override a package anyways, add the following t
o the command line:
        --allow-overriding turtlesim

This may be promoted to an error in a future release of colcon-override-check.
Starting >>> turtlesim
--- stderr: turtlesim
failed to create symbolic link '/home/jonathan/ros2_ws2/build/turtlesim/ament_cmake_pyt
hon/turtlesim/turtlesim' because existing path cannot be removed: Is a directory
gmake[2]: *** [CMakeFiles/ament_cmake_python_symlink_turtlesim.dir/build.make:70: CMake
Files/ament_cmake_python_symlink_turtlesim] Error 1
gmake[1]: *** [CMakeFiles/Makefile2:429: CMakeFiles/ament_cmake_python_symlink_turtlesi
m.dir/all] Error 2
gmake[1]: *** Waiting for unfinished jobs....
gmake: *** [Makefile:146: all] Error 2
---
Failed   <<< turtlesim [2.09s, exited with code 2]

Summary: 0 packages finished [2.28s]
  1 package failed: turtlesim
  1 package had stderr output: turtlesim
```

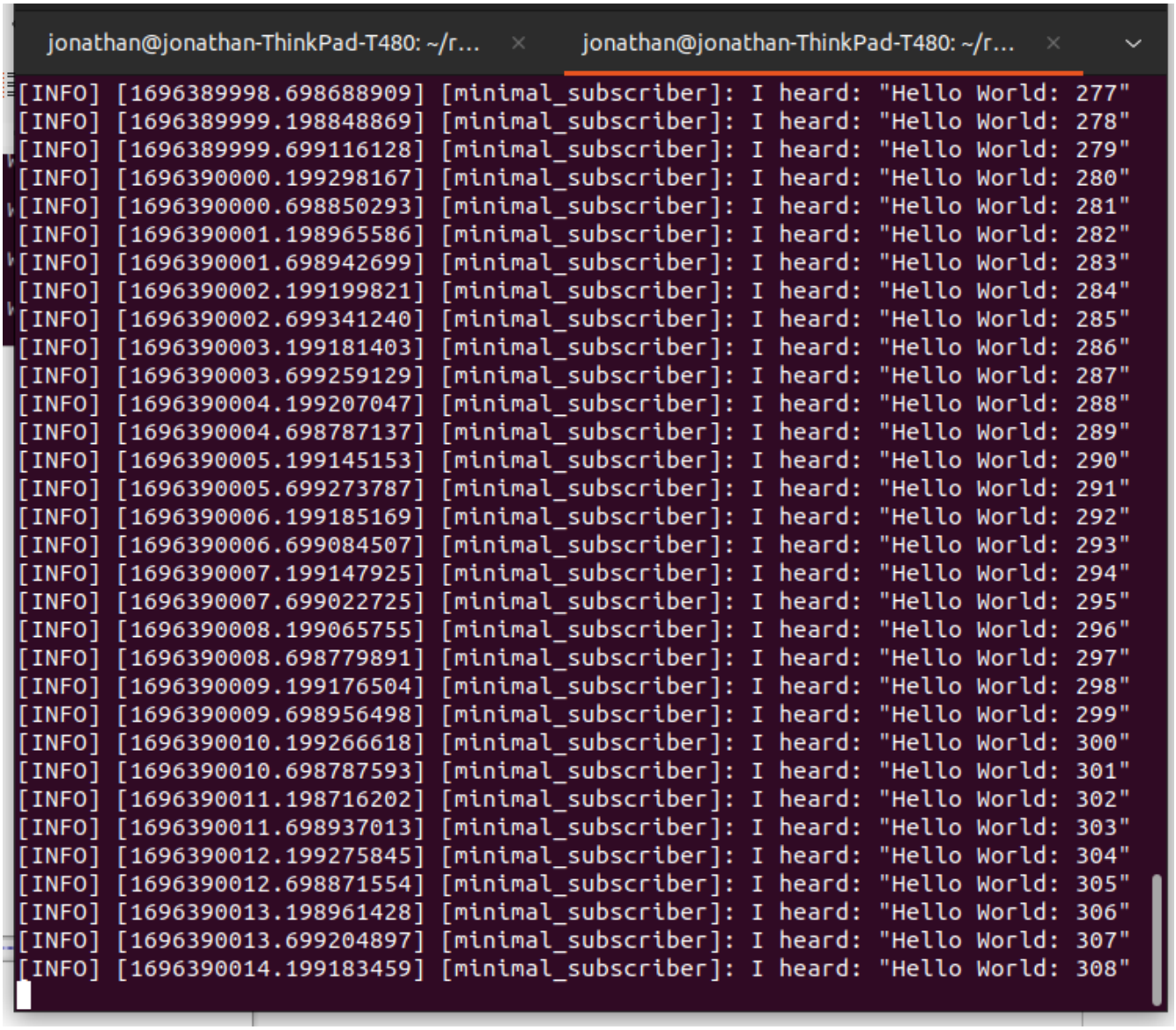
Above is the picture of the colcon build that failed because of the cmake list, for some events, the ram was overwhelmed with the

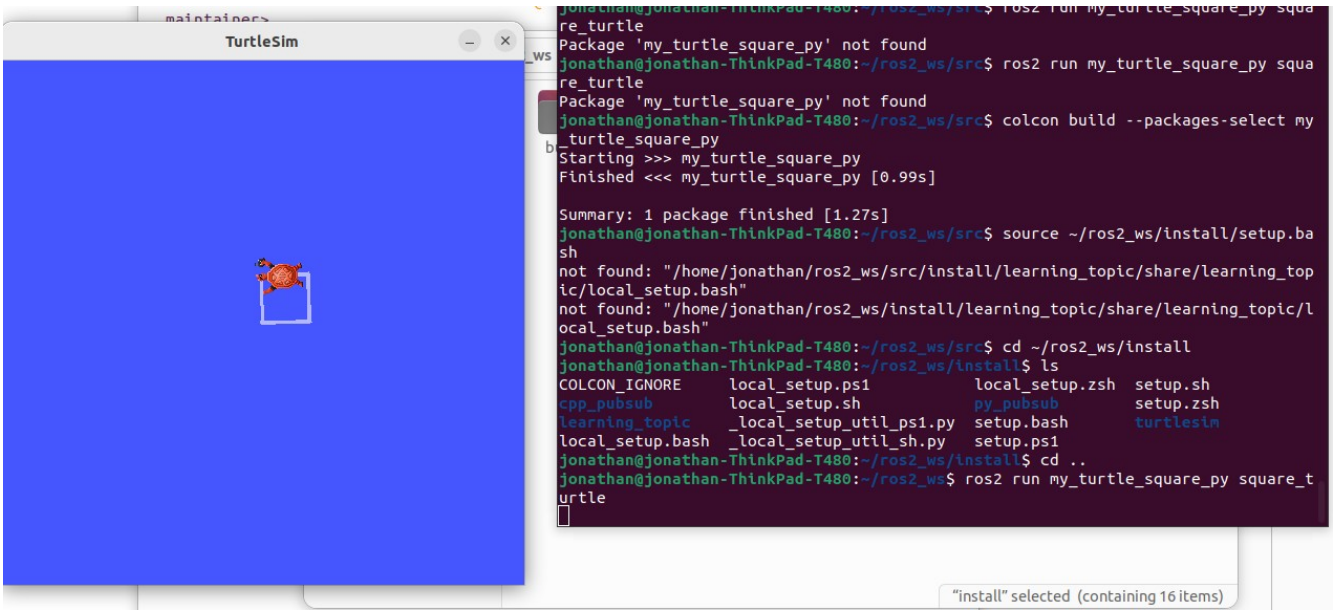
Successfully build the publisher and listener node using python:



Above is the publishing node

Listener Node





I have a success using python instead of a c++ code to make the turtle to create a square
It took a while, but I was able to create the turtle to create the square.

I have to change several stuffs such as:

```
<exec_depend>rclpy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

on package.xml

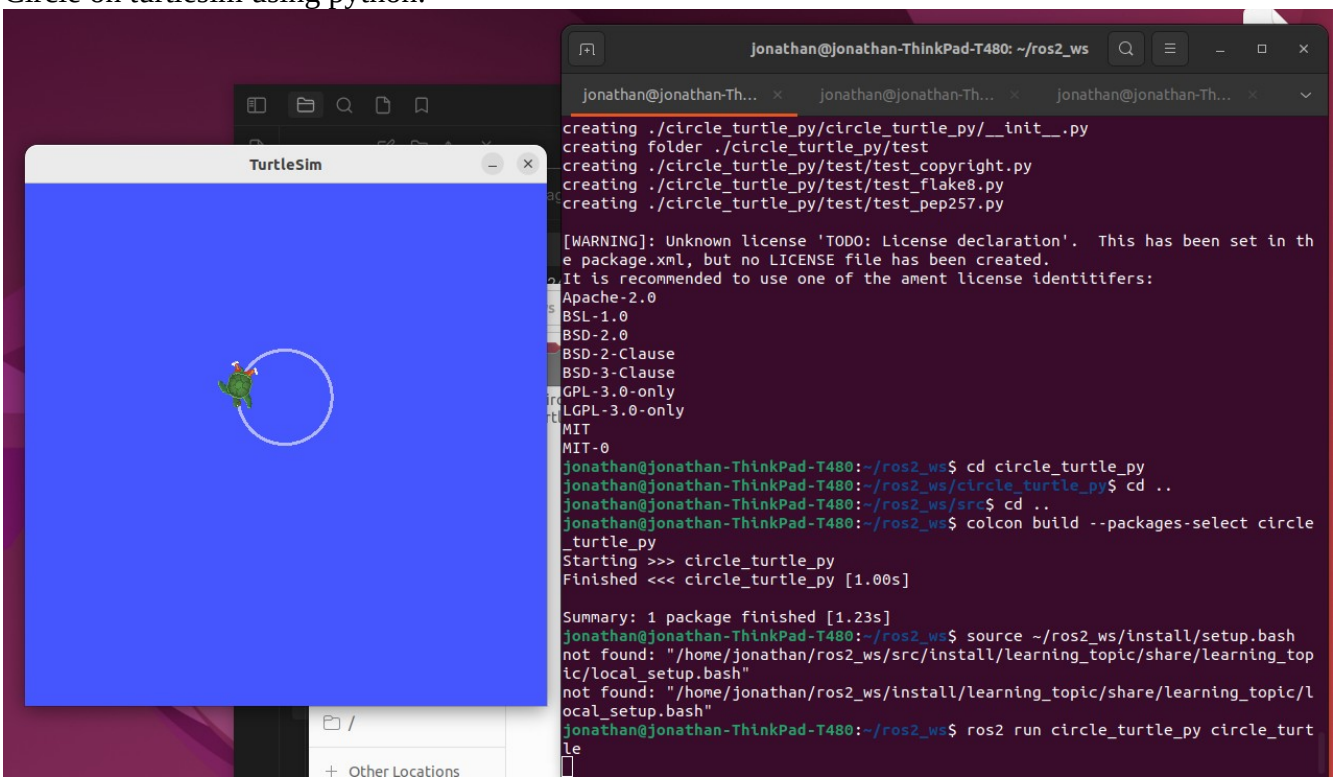
we also need to add the entry point like described below:

```
        'console_scripts': [
                'square_turtle = my_turtle_square_py.square_turtle:main'
        ],
    },
)
```

python code:

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist

class SquareTurtle(Node):
    def __init__(self):
        super().__init__('square_turtle')
        self.publisher_ = self.create_publisher(Twist, 'turtle1/cmd_vel', 10)
        self.timer = self.create_timer(2, self.publish_square)
        self.counter = 0

    def publish_square(self):
        vel_msg = Twist()
        if self.counter % 4 == 0:
            vel_msg.linear.x = 1.0
        elif self.counter % 4 == 1:
            vel_msg.angular.z = 1.5708  # 90 degrees in radians
        elif self.counter % 4 == 2:
            vel_msg.linear.x = 1.0
        elif self.counter % 4 == 3:
            vel_msg.angular.z = 1.5708  # 90 degrees in radians

        self.publisher_.publish(vel_msg)
        self.counter += 1

def main(args=None):
    rclpy.init(args=args)
    square_turtle = SquareTurtle()
    rclpy.spin(square_turtle)
    square_turtle.destroy_node()
    rclpy.shutdown()

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

Circle on turtlesim using python:



python code to make circle using python:

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
import math

class CircleTurtle(Node):
    def __init__(self):
        super().__init__('circle_turtle')
        self.publisher_ = self.create_publisher(Twist, 'turtle1/cmd_vel', 10)
        self.timer = self.create_timer(0.1, self.publish_circle)
        self.angle = 0

    def publish_circle(self):
        vel_msg = Twist()
        vel_msg.linear.x = 1.0  # Linear velocity
        vel_msg.angular.z = 1.0  # Angular velocity

        self.publisher_.publish(vel_msg)

        # Increase the angle
        self.angle += 0.1

        # Reset angle to 0 after completing the circle
        if self.angle >= 2 * math.pi:
            self.angle = 0

def main(args=None):
    rclpy.init(args=args)
    circle_turtle = CircleTurtle()
    rclpy.spin(circle_turtle)
    circle_turtle.destroy_node()
    rclpy.shutdown()

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