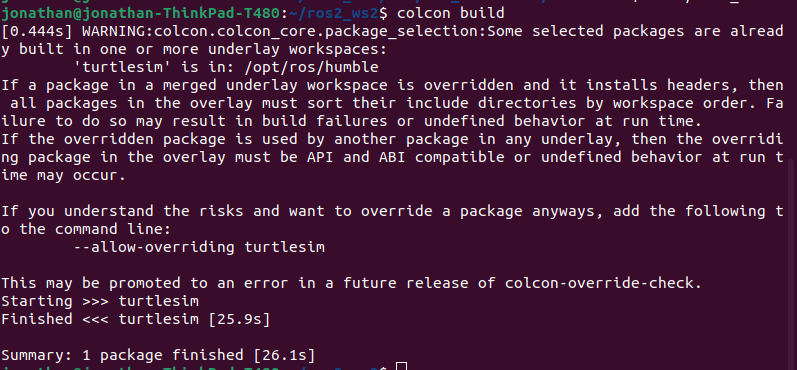
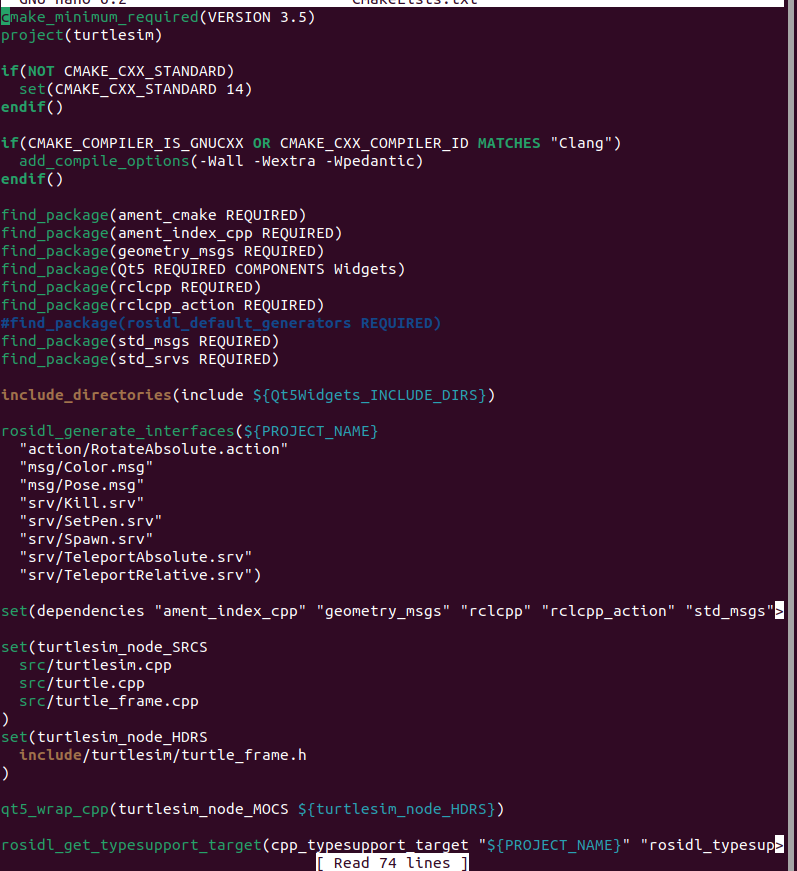
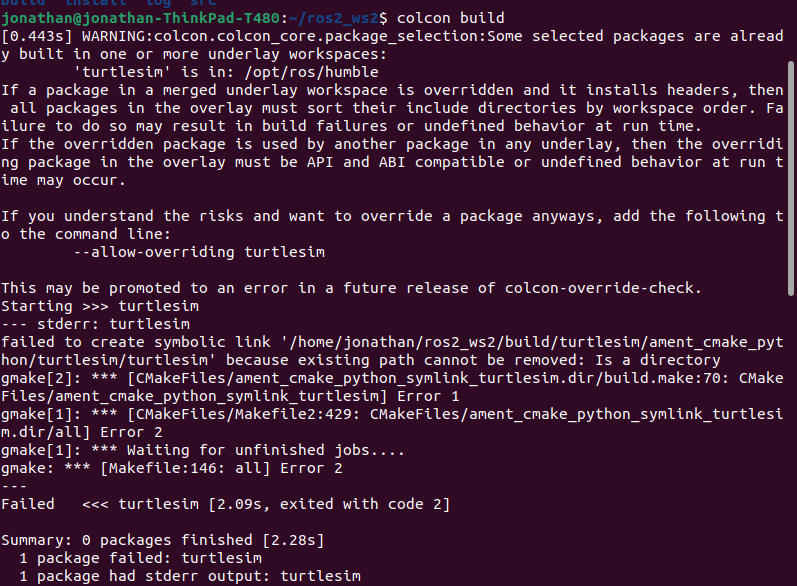
Ros 2 humble

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

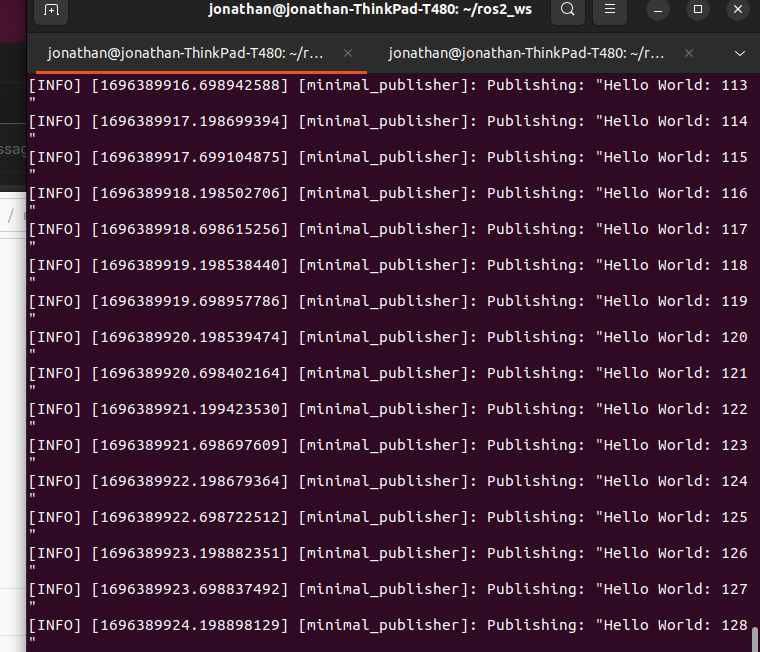


I have to comment out #find\_package(rosidl\_default\_generators REQUIRED) in order for me to be able to do it.



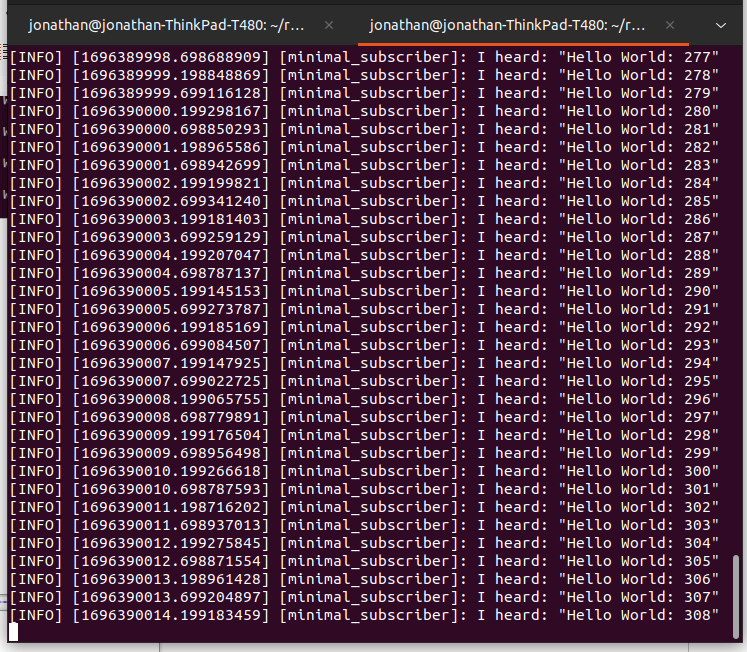
Above is the picture of the colcon build that failed because of the cmake list, for some events, the ram was overwhelmed with the

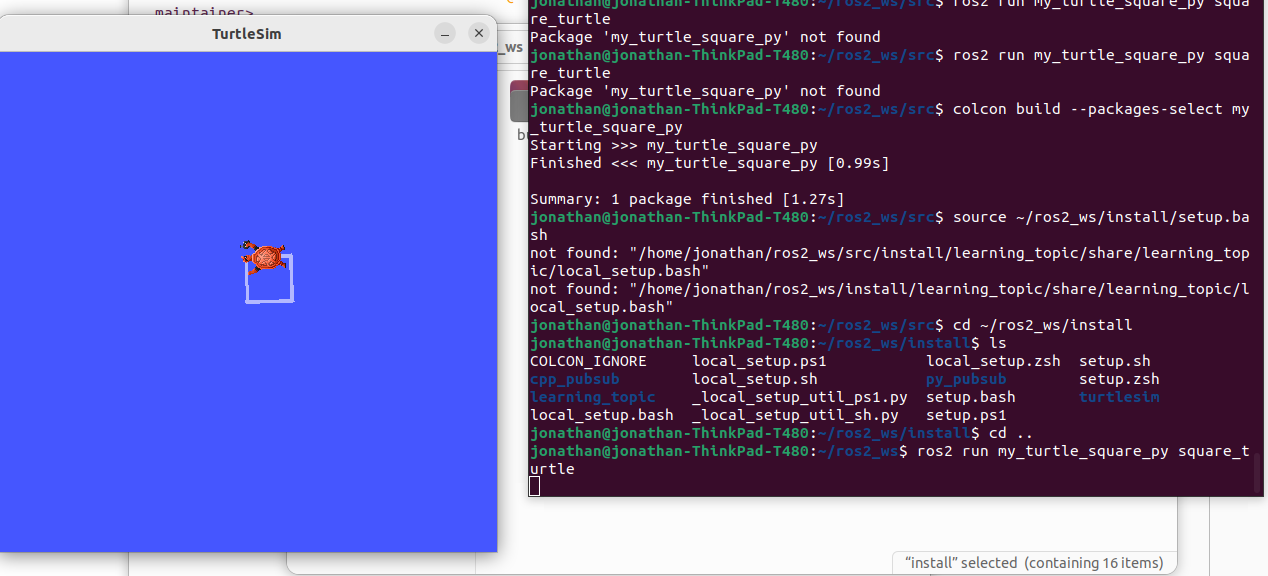
Successfully build the publisher and listener node using python:



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.

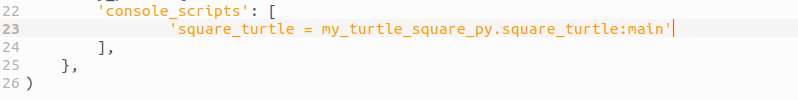
I have to change several stuffs such as:

<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:



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)

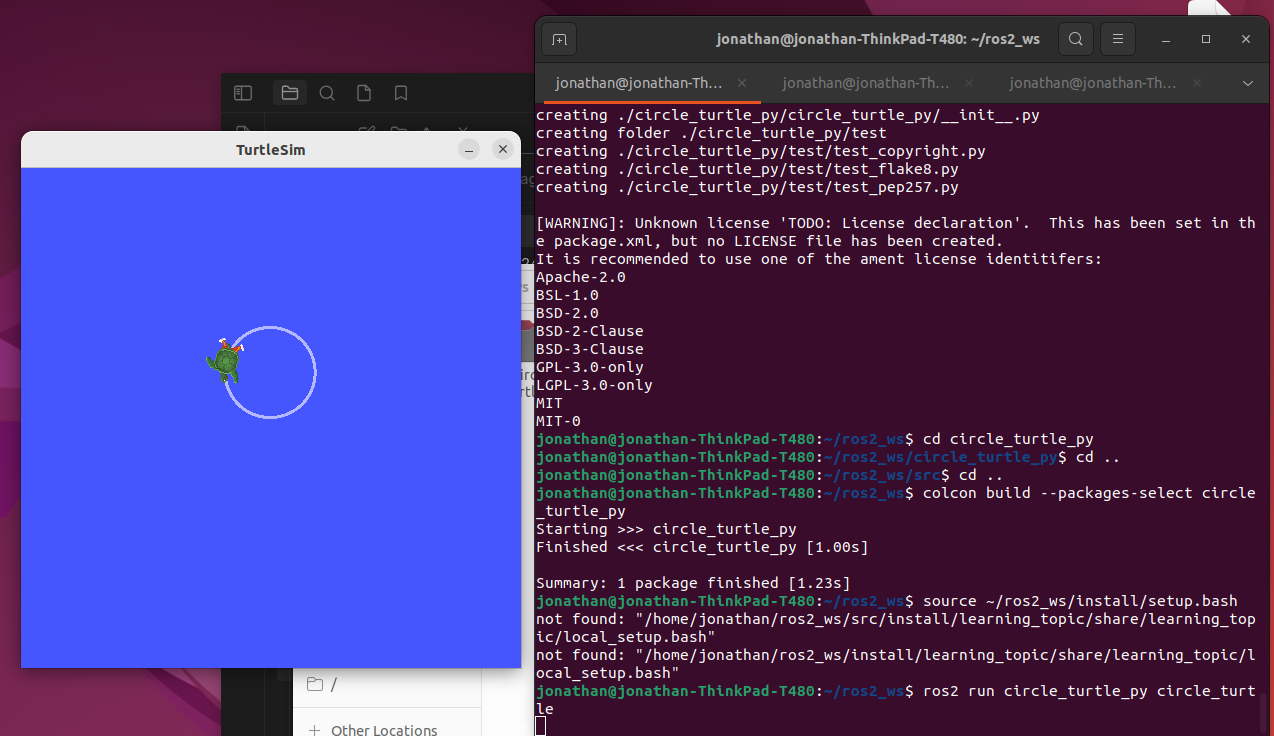
square\_turtle.destroy\_node()

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()