



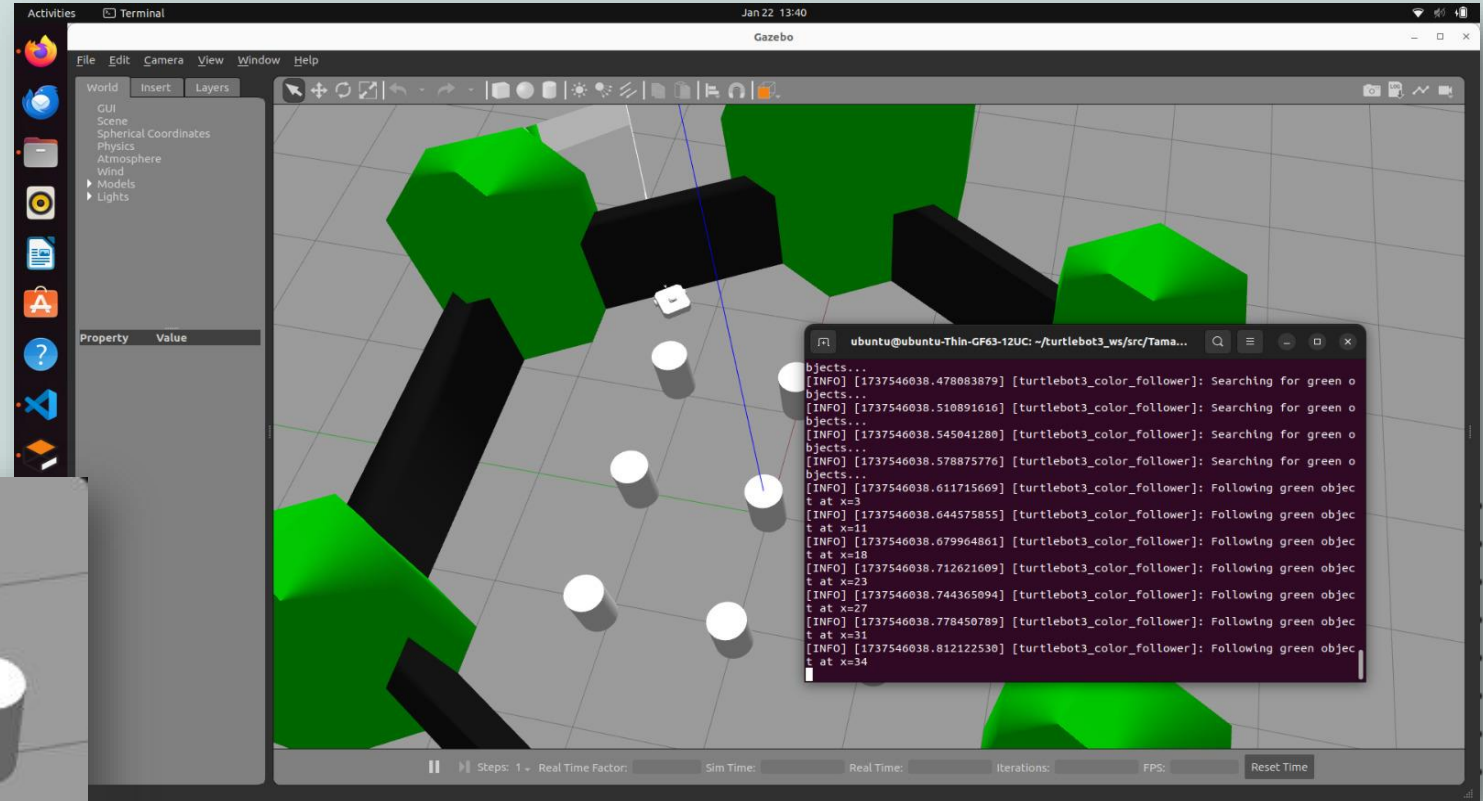
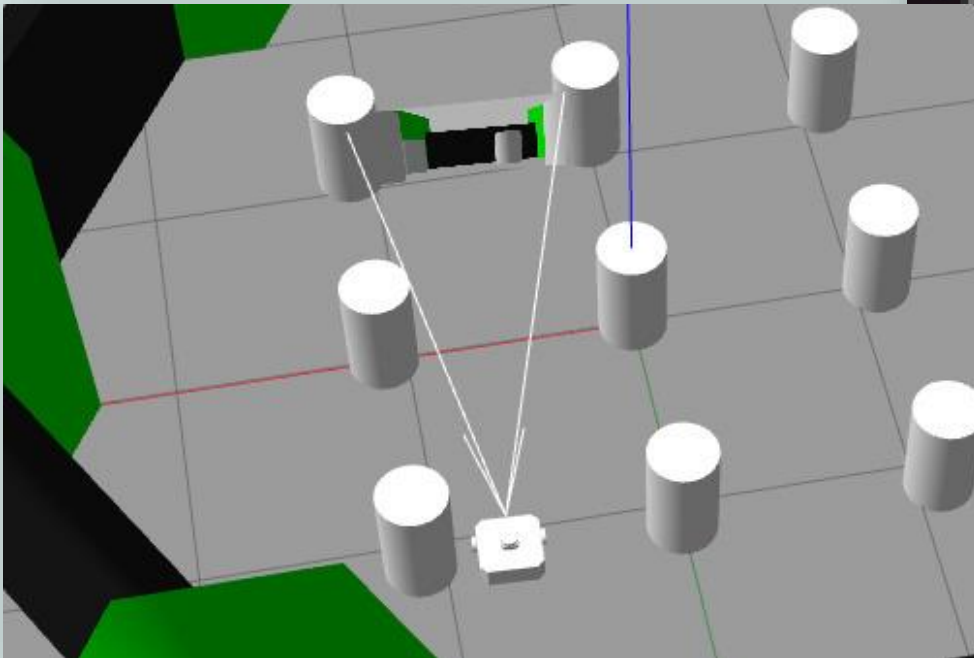
# PROIECT SINCRETIC TAMAGOTCHI

VERNICHESCU CRISTIAN

MIHAESCU CRISTINA -  
MARIA

# OBIECTIVELE PROIECTULUI

Dezvoltarea unui robot care detecteaza  
si se deplaseaza catre culoarea verde,  
si fuge de culoarea rosie



Screenshot din Gazebo cu robotul care se  
indreapta catre culoarea verde

# INSTRUMENTE ȘI TEHNOLOGII

## *Hardware:*

Robot cu senzor/camera

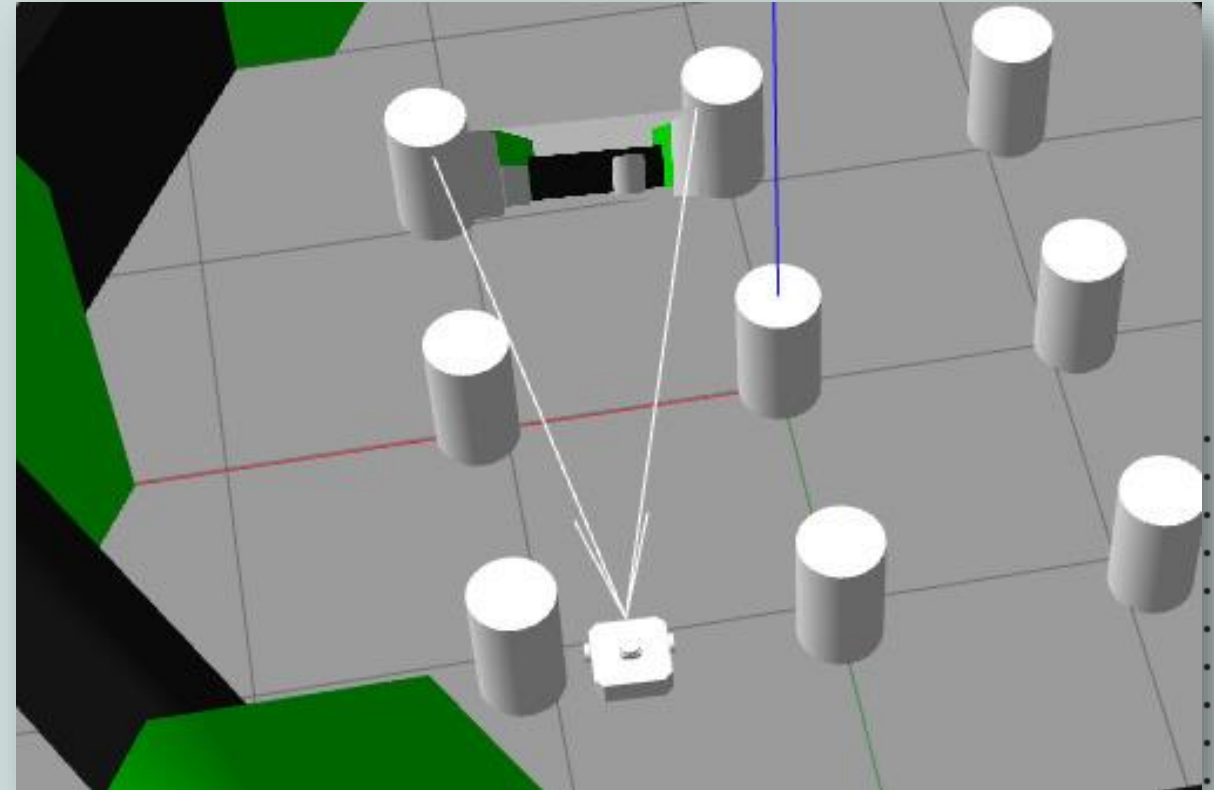
Platforma Ubuntu

## *Software:*

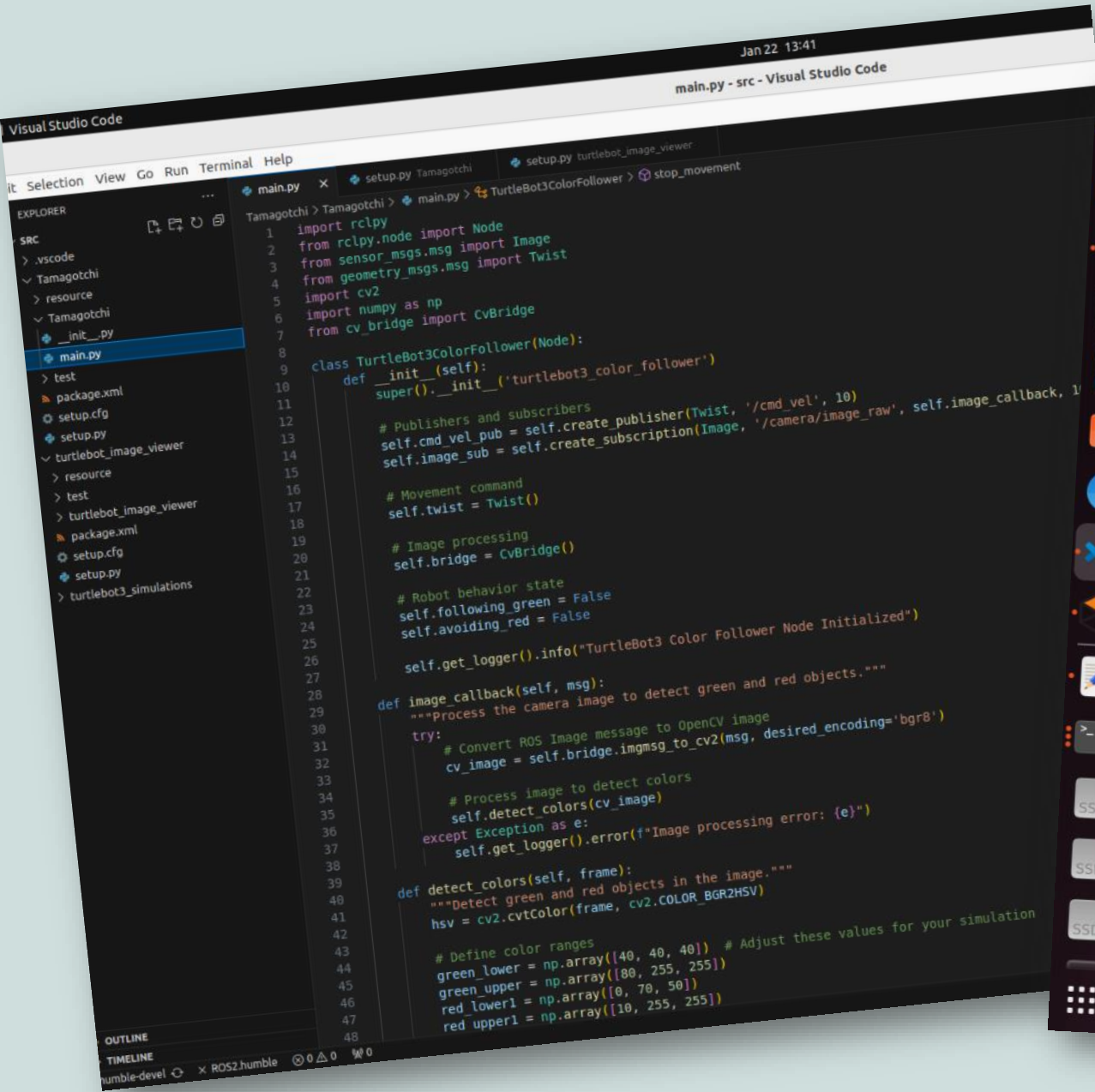
ROS2: comunicatia intre senzori si controlul miscarii

OpenCV: procesarea imaginilor

Python: limbajul principal de programare



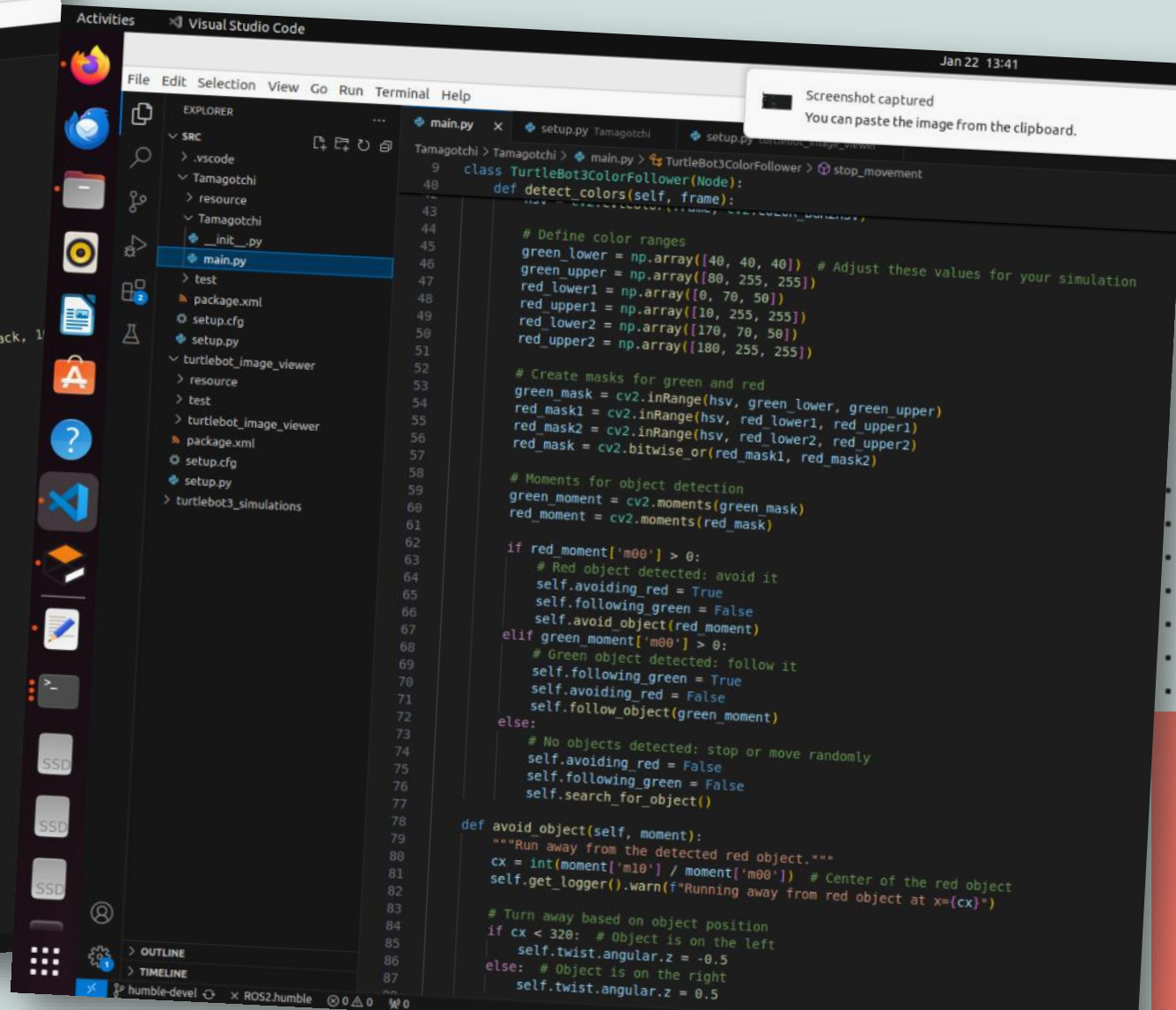
# FRAGMENTE DE COD RELEVANTE



```
Jan 22 13:41
main.py - src - Visual Studio Code

EXPLORER
SRC
> .vscode
> Tamagotchi
> resource
> Tamagotchi
  > _init_.py
  > main.py
  > test
  > package.xml
  > setup.cfg
  > setup.py
  > turtlebot_image_viewer
  > resource
  > test
  > turtlebot_image_viewer
  > package.xml
  > setup.cfg
  > setup.py
  > turtlebot3_simulations

main.py
1 import rclpy
2 from rclpy.node import Node
3 from sensor_msgs.msg import Image
4 from geometry_msgs.msg import Twist
5 import cv2
6 import numpy as np
7 from cv_bridge import CvBridge
8
9 class TurtleBot3ColorFollower(Node):
10     def __init__(self):
11         super().__init__('turtlebot3_color_follower')
12
13         # Publishers and subscribers
14         self.cmd_vel_pub = self.create_publisher(Twist, '/cmd_vel', 10)
15         self.image_sub = self.create_subscription(Image, '/camera/image_raw', self.image_callback, 1)
16
17         # Movement command
18         self.twist = Twist()
19
20         # Image processing
21         self.bridge = CvBridge()
22
23         # Robot behavior state
24         self.following_green = False
25         self.avoiding_red = False
26
27         self.get_logger().info("TurtleBot3 Color Follower Node Initialized")
28
29     def image_callback(self, msg):
30         """Process the camera image to detect green and red objects."""
31         try:
32             # Convert ROS Image message to OpenCV image
33             cv_image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
34
35             # Process image to detect colors
36             self.detect_colors(cv_image)
37         except Exception as e:
38             self.get_logger().error(f"Image processing error: {e}")
39
40     def detect_colors(self, frame):
41         """Detect green and red objects in the image."""
42         hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
43
44         # Define color ranges
45         green_lower = np.array([40, 40, 40]) # Adjust these values for your simulation
46         green_upper = np.array([80, 255, 255])
47         red_lower1 = np.array([0, 70, 50])
48         red_upper1 = np.array([10, 255, 255])
49         red_lower2 = np.array([170, 70, 50])
50         red_upper2 = np.array([180, 255, 255])
51
52         # Create masks for green and red
53         green_mask = cv2.inRange(hsv, green_lower, green_upper)
54         red_mask1 = cv2.inRange(hsv, red_lower1, red_upper1)
55         red_mask2 = cv2.inRange(hsv, red_lower2, red_upper2)
56         red_mask = cv2.bitwise_or(red_mask1, red_mask2)
57
58         # Moments for object detection
59         green_moment = cv2.moments(green_mask)
60         red_moment = cv2.moments(red_mask)
61
62         if red_moment['m00'] > 0:
63             # Red object detected: avoid it
64             self.avoiding_red = True
65             self.following_green = False
66             self.avoid_object(red_moment)
67         elif green_moment['m00'] > 0:
68             # Green object detected: follow it
69             self.following_green = True
70             self.avoiding_red = False
71             self.follow_object(green_moment)
72         else:
73             # No objects detected: stop or move randomly
74             self.avoiding_red = False
75             self.following_green = False
76             self.search_for_object()
77
78     def avoid_object(self, moment):
79         """Run away from the detected red object."""
80         cx = int(moment['m10'] / moment['m00']) # Center of the red object
81         self.get_logger().warn(f"Running away from red object at x={cx}")
82
83         # Turn away based on object position
84         if cx < 320: # Object is on the left
85             self.twist.angular.z = -0.5
86         else: # Object is on the right
87             self.twist.angular.z = 0.5
```



```
Jan 22 13:41
main.py - src - Visual Studio Code

EXPLORER
SRC
> .vscode
> Tamagotchi
> resource
> Tamagotchi
  > _init_.py
  > main.py
  > test
  > package.xml
  > setup.cfg
  > setup.py
  > turtlebot_image_viewer
  > resource
  > test
  > turtlebot_image_viewer
  > package.xml
  > setup.cfg
  > setup.py
  > turtlebot3_simulations

main.py
9 class TurtleBot3ColorFollower(Node):
10     def detect_colors(self, frame):
11         # Define color ranges
12         green_lower = np.array([40, 40, 40]) # Adjust these values for your simulation
13         green_upper = np.array([80, 255, 255])
14         red_lower1 = np.array([0, 70, 50])
15         red_upper1 = np.array([10, 255, 255])
16         red_lower2 = np.array([170, 70, 50])
17         red_upper2 = np.array([180, 255, 255])
18
19         # Create masks for green and red
20         green_mask = cv2.inRange(hsv, green_lower, green_upper)
21         red_mask1 = cv2.inRange(hsv, red_lower1, red_upper1)
22         red_mask2 = cv2.inRange(hsv, red_lower2, red_upper2)
23         red_mask = cv2.bitwise_or(red_mask1, red_mask2)
24
25         # Moments for object detection
26         green_moment = cv2.moments(green_mask)
27         red_moment = cv2.moments(red_mask)
28
29         if red_moment['m00'] > 0:
30             # Red object detected: avoid it
31             self.avoiding_red = True
32             self.following_green = False
33             self.avoid_object(red_moment)
34         elif green_moment['m00'] > 0:
35             # Green object detected: follow it
36             self.following_green = True
37             self.avoiding_red = False
38             self.follow_object(green_moment)
39         else:
40             # No objects detected: stop or move randomly
41             self.avoiding_red = False
42             self.following_green = False
43             self.search_for_object()
44
45     def avoid_object(self, moment):
46         """Run away from the detected red object."""
47         cx = int(moment['m10'] / moment['m00']) # Center of the red object
48         self.get_logger().warn(f"Running away from red object at x={cx}")
49
50         # Turn away based on object position
51         if cx < 320: # Object is on the left
52             self.twist.angular.z = -0.5
53         else: # Object is on the right
54             self.twist.angular.z = 0.5
```

Screenshot captured  
You can paste the image from the clipboard.



Jan 22 13:41

Screenshot captured  
You can paste the image from the clipboard.

```
Run Terminal Help
main.py x setup.py Tamagotchi setup.py turtlebot3_image_viewer stop_movement

Tamagotchi > Tamagotchi > main.py > TurtleBot3ColorFollower > stop_movement
9 class TurtleBot3ColorFollower(Node):
78     def avoid_object(self, moment):
87         self.twist.angular.z = 0.0
88
89     # Move backward
90     self.twist.linear.x = -0.2
91     self.cmd_vel_pub.publish(self.twist)
92
93     def follow_object(self, moment):
94         """Follow the detected green object."""
95         cx = int(moment['m10'] / moment['m00']) # Center of the green object
96         self.get_logger().info(f"Following green object at x={cx}")
97
98         # Adjust direction based on object's position
99         if cx < 320: # Object is on the left
100             self.twist.angular.z = 0.3
101         elif cx > 320: # Object is on the right
102             self.twist.angular.z = -0.3
103         else:
104             self.twist.angular.z = 0.0
105
106     # Move forward
107     self.twist.linear.x = 0.2
108     self.cmd_vel_pub.publish(self.twist)
109
110     def search_for_object(self):
111         """Move randomly when no objects are detected."""
112         self.get_logger().info("Searching for green objects...")
113         self.twist.linear.x = 0.1
114         self.twist.angular.z = 0.2
115         self.cmd_vel_pub.publish(self.twist)
116
117     def stop_movement(self):
118         """Stop the robot."""
119         self.twist.linear.x = 0.0
120         self.twist.angular.z = 0.0
121         self.cmd_vel_pub.publish(self.twist)
122
123     def main(args=None):
124         rclpy.init(args=args)
125         color_follower = TurtleBot3ColorFollower()
126
127         try:
128             rclpy.spin(color_follower)
129         except KeyboardInterrupt:
130             color_follower.get_logger().info("Shutting down Color Follower Node")
131             color_follower.stop_movement()
132         finally:
133             color_follower.destroy_node()
134             rclpy.shutdown()
```

Jan 22 13:41

Screenshot captured  
You can paste the image from the clipboard.

Visual Studio Code

Selection View Go Run Terminal Help

main.py x setup.py Tamagotchi setup.py turtlebot3\_image\_viewer stop\_movement

```
main.py x setup.py Tamagotchi setup.py turtlebot3_image_viewer stop_movement
9 class TurtleBot3ColorFollower(Node):
93     def follow_object(self, moment):
101         self.twist.angular.z = 0.3
102         elif cx > 320: # Object is on the right
103             self.twist.angular.z = -0.3
104         else:
105             self.twist.angular.z = 0.0
106
107     # Move forward
108     self.twist.linear.x = 0.2
109     self.cmd_vel_pub.publish(self.twist)
110
111     def search_for_object(self):
112         """Move randomly when no objects are detected."""
113         self.get_logger().info("Searching for green objects...")
114         self.twist.linear.x = 0.1
115         self.twist.angular.z = 0.2
116         self.cmd_vel_pub.publish(self.twist)
117
118     def stop_movement(self):
119         """Stop the robot."""
120         self.twist.linear.x = 0.0
121         self.twist.angular.z = 0.0
122         self.cmd_vel_pub.publish(self.twist)
123
124     def main(args=None):
125         rclpy.init(args=args)
126         color_follower = TurtleBot3ColorFollower()
127
128         try:
129             rclpy.spin(color_follower)
130         except KeyboardInterrupt:
131             color_follower.get_logger().info("Shutting down Color Follower Node")
132             color_follower.stop_movement()
133         finally:
134             color_follower.destroy_node()
135             rclpy.shutdown()
136
137 if __name__ == '__main__':
138     main()
```

# ROLUL ROS2

*Robot Operating System 2* (ROS2) este cadrul software care gestioneaza comunicarea intre componentele robotului. Este esential pentru coordonarea senzorilor, procesarea datelor si controlul miscarii robotului.

ROS2 faciliteaza schimbul de informatii intre nodurile care gestioneaza functii specifice, cum ar fi:  
nodul pentru capturarea imaginii de la camera  
nodul care proceseaza imaginea pentru detectarea verdelui si rosului  
nodul care trimite comenzi de miscare catre motoarele robotului

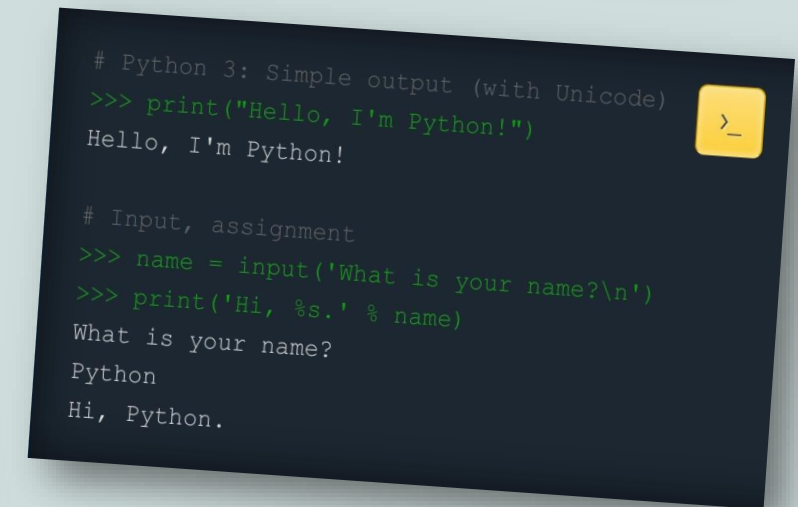


# ROLUL PYTHON

*Python* este limbajul de programare folosit pentru a scrie nodurile ROS2 si pentru a implementa logica necesara proiectului. Este preferat in robotica datorita simplitatii sale si a suportului extins pentru biblioteci relevante.

Python implementează algoritmul care decide cum să se deplaseze robotul.

Python folosește biblioteca ROS2 pentru a comunica cu restul sistemului.



# *Concluzii*

***Lectii Invatate:*** Importanta procesarii imaginilor în robotica. Integrarea hardware-software

Realizarea acestui proiect a demonstrat importanța integrării unor tehnologii avansate precum ROS2, Ubuntu și Python pentru dezvoltarea unui robot autonom. Fiecare componentă a avut un rol esențial în funcționarea corectă și eficientă a sistemului.

Această combinație de tehnologii a permis dezvoltarea unui sistem autonom capabil să detecteze culoarea verde, să se deplaseze către ea și să evite culoarea roșie. Proiectul evidențiază puterea integrării software-hardware pentru rezolvarea unor sarcini complexe și demonstrează cât de esențial este un ecosistem bine definit în robotică.