Technical documentation for ButlerRobot ROS Node

Task: The ButlerRobot is designed to navigate between a home, kitchen and different tables.

Code Documentation:

# Import Libraries

```
import rospy
from geometry_msgs.msg import Twist
from std_msgs.msg import String
from enum import Enum
```

# Robot state enumeration: 7 states : .

```
class RobotState(Enum):
    HOME = 1
    TO_KITCHEN = 2
    WAITING_AT_KITCHEN = 3
    TO_TABLE = 4
    WAITING_AT_TABLE = 5
    RETURNING_HOME = 6
    CANCELLED = 7
```

Purpose: Defines various states of the robot to manage its behavior and transitions effectively.

# ButlerRobot Class

Constructor

```
class ButlerRobot:
    def __init__(self):
        rospy.init_node('butler_robot', anonymous=True)
        self.pub_cmd = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
        self.sub_order = rospy.Subscriber('/order', String, self.order_callback)
        self.sub_confirm = rospy.Subscriber('/confirm', String, self.confirm_callback)

        self.rate = rospy.Rate(10)
        self.state = RobotState.HOME
        self.current_task = None
        self.task_queue = []
        self.timeout_duration = rospy.get_param('~timeout_duration', 10)

        # Load positions (example coordinates)
        self.home_position = rospy.get_param('~home_position', [0, 0])
        self.kitchen_position = rospy.get_param('~kitchen_position', [5, 0])
        self.table_positions = {
            'table1': rospy.get_param('~table1_position', [10, 0]),
            'table2': rospy.get_param('~table2_position', [10, 5]),
            'table3': rospy.get_param('~table3_position', [5, 10])
        }
```
Purpose: Initializes ROS node, publishers, and subscribers. Sets up parameters for robot's positions and task queue.

# Order Callback

```
def order_callback(self, msg):
        order = msg.data
        self.task_queue.append(order.split(','))
        rospy.loginfo(f"Received order: {order}")
        if self.state == RobotState.HOME:
            self.process_next_task()
```

Purpose: Processes incoming orders, adds them to the task queue, and starts task processing if the robot is at home.

# Confirmation Callback

```
def confirm_callback(self, msg):
        confirmation = msg.data
        rospy.loginfo(f"Received confirmation: {confirmation}")
        if self.state == RobotState.WAITING_AT_KITCHEN:
            self.state = RobotState.TO_TABLE
            self.navigate_to_position(self.current_task[0])
        elif self.state == RobotState.WAITING_AT_TABLE:
            self.process_next_task()
```

Purpose: Handles confirmations received from the `/confirm` topic and transitions states based on the current task.

# Process Next Task

```
def process_next_task(self):
        if not self.task_queue:
            self.state = RobotState.RETURNING_HOME
            self.navigate_to_position(self.home_position)
            return

        self.current_task = self.task_queue.pop(0)
        self.state = RobotState.TO_KITCHEN
        self.navigate_to_position(self.kitchen_position)
```

Purpose: Manages and processes the next task from the queue, including navigating to the kitchen or returning home if the queue is empty.

# Navigate to Position

```
def navigate_to_position(self, position):
        rospy.loginfo(f"Navigating to position: {position}")
        move_cmd = Twist()
        # Implement actual navigation logic here
        self.pub_cmd.publish(move_cmd)
        rospy.sleep(1)  # Simulate navigation time

        rospy.loginfo(f"Current state before transition: {self.state}")

        if self.state == RobotState.TO_KITCHEN:
        self.state = RobotState.WAITING_AT_KITCHEN
        rospy.loginfo("Robot State: Changed to WAITING_AT_KITCHEN")
        self.wait_for_confirmation()
        elif self.state == RobotState.TO_TABLE:
        self.state = RobotState.WAITING_AT_TABLE
        rospy.loginfo("Robot State: Changed to WAITING_AT_TABLE")
        self.wait_for_confirmation()
        elif self.state == RobotState.RETURNING_HOME:
        self.state = RobotState.HOME
        rospy.loginfo("Robot State: Changed to HOME")
```

Purpose: Simulates navigation to the specified position, logs the state change, and waits for confirmation if necessary.

# Wait for Confirmation

```
def wait_for_confirmation(self):
    start_time = rospy.get_time()
    while rospy.get_time() - start_time < self.timeout_duration:
      rospy.sleep(1)
      if self.state not in [RobotState.WAITING_AT_KITCHEN, RobotState.WAITING_AT_TABLE]:
        return

    rospy.loginfo("Timeout reached, handling scenario.")
    if self.state == RobotState.WAITING_AT_KITCHEN:
      self.state = RobotState.RETURNING_HOME
      self.navigate_to_position(self.home_position)
    elif self.state == RobotState.WAITING_AT_TABLE:
      self.state = RobotState.TO_KITCHEN
      self.navigate_to_position(self.kitchen_position)
```

Purpose: Waits for a confirmation message within a timeout period and handles timeout scenarios by changing states as needed.

# Run Method

```
def run(self):
    rospy.spin()
```

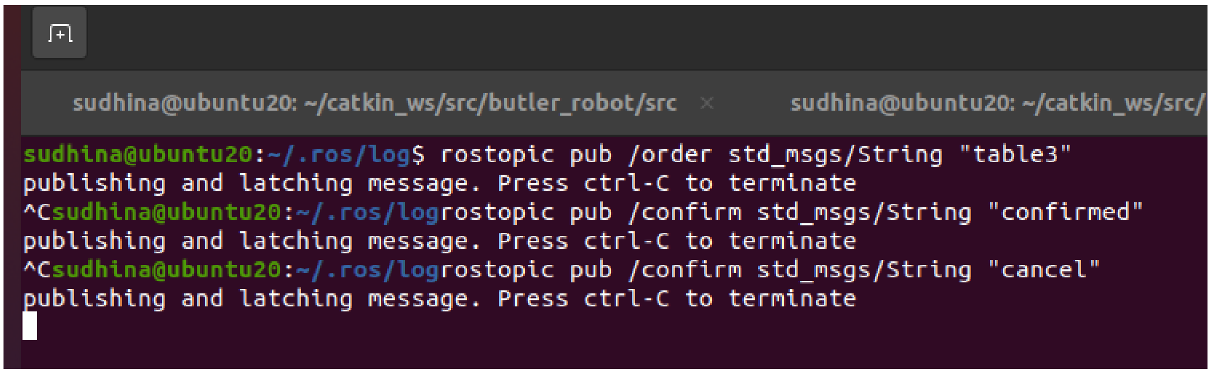
Purpose: Keeps the ROS node running, allowing it to continuously process messages and callbacks.

Main Execution Block

```
if __name__ == '__main__':
    try:
        robot = ButlerRobot()
        robot.run()
    except rospy.ROSInterruptException:
        pass
```

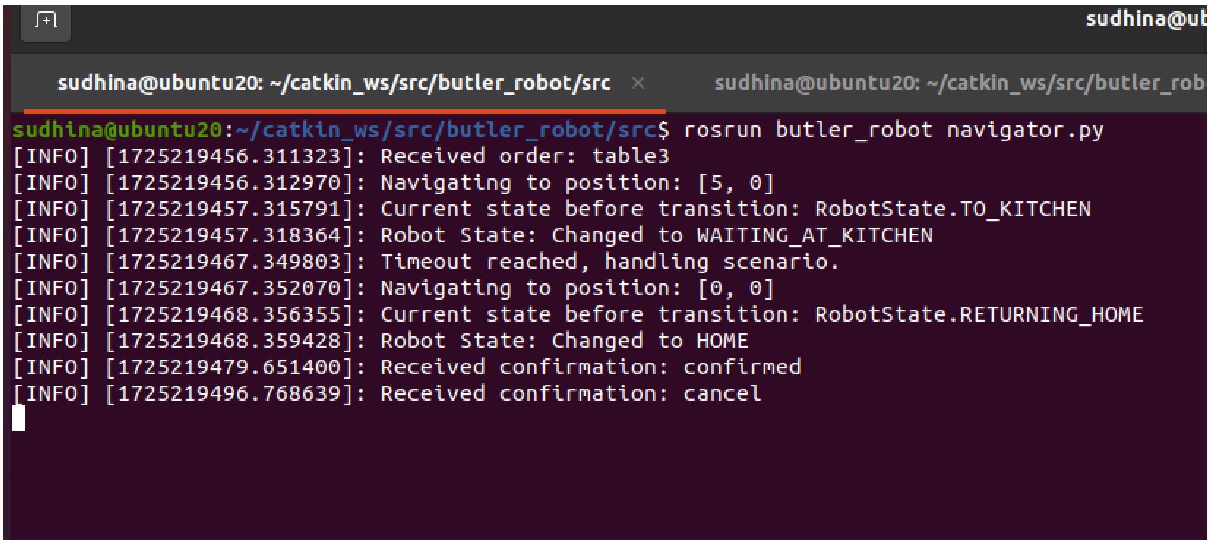
Purpose: Creates an instance of `ButlerRobot` and starts the ROS node. Handles interruptions gracefully.

User Input:


sudhina@ubuntu20: ~/catkin_ws/src/butler_robot/src
sudhina@ubuntu20: ~/catkin_ws/src/

```
sudhina@ubuntu20:~/.ros/log$ rostopic pub /order std_msgs/String "table3"
publishing and latching message. Press ctrl-C to terminate
^Csudhina@ubuntu20:~/.ros/logrostopic pub /confirm std_msgs/String "confirmed"
publishing and latching message. Press ctrl-C to terminate
^Csudhina@ubuntu20:~/.ros/logrostopic pub /confirm std_msgs/String "cancel"
publishing and latching message. Press ctrl-C to terminate
```

Output Obtained:


sudhina@ub
sudhina@ubuntu20: ~/catkin_ws/src/butler_robot/src
sudhina@ubuntu20: ~/catkin_ws/src/butler_rob

```
sudhina@ubuntu20:~/catkin_ws/src/butler_robot/src$ rosrun butler_robot navigator.py
[INFO] [1725219456.311323]: Received order: table3
[INFO] [1725219456.312970]: Navigating to position: [5, 0]
[INFO] [1725219457.315791]: Current state before transition: RobotState.TO_KITCHEN
[INFO] [1725219457.318364]: Robot State: Changed to WAITING_AT_KITCHEN
[INFO] [1725219467.349803]: Timeout reached, handling scenario.
[INFO] [1725219467.352070]: Navigating to position: [0, 0]
[INFO] [1725219468.356355]: Current state before transition: RobotState.RETURNING_HOME
[INFO] [1725219468.359428]: Robot State: Changed to HOME
[INFO] [1725219479.651400]: Received confirmation: confirmed
[INFO] [1725219496.768639]: Received confirmation: cancel
```