

Linear velocity calibration

This lesson uses the Raspberry Pi as an example. For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to [\[01. Robot Configuration and Operation Guide\] -- \[4.Enter Docker \(For JETSON Nano and RPi 5\)\]](#). For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

1. Program Description

Run the program and adjust the parameters using the dynamic parameter adjuster to calibrate the linear velocity of the car. To visually demonstrate the calibration linear velocity, give the command to move the car forward 1 meter and observe how far it actually travels and whether it falls within the error range.

2. Program Startup

2.1 Startup Command

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

Enter the Docker container (see [\[Docker course\] --- \[4. Docker Startup Script\]](#) for steps).

All the following commands must be executed within the same Docker container ([see \[Docker course\] --- \[3. Docker Submission and Multi-Terminal Access\]](#) for steps).

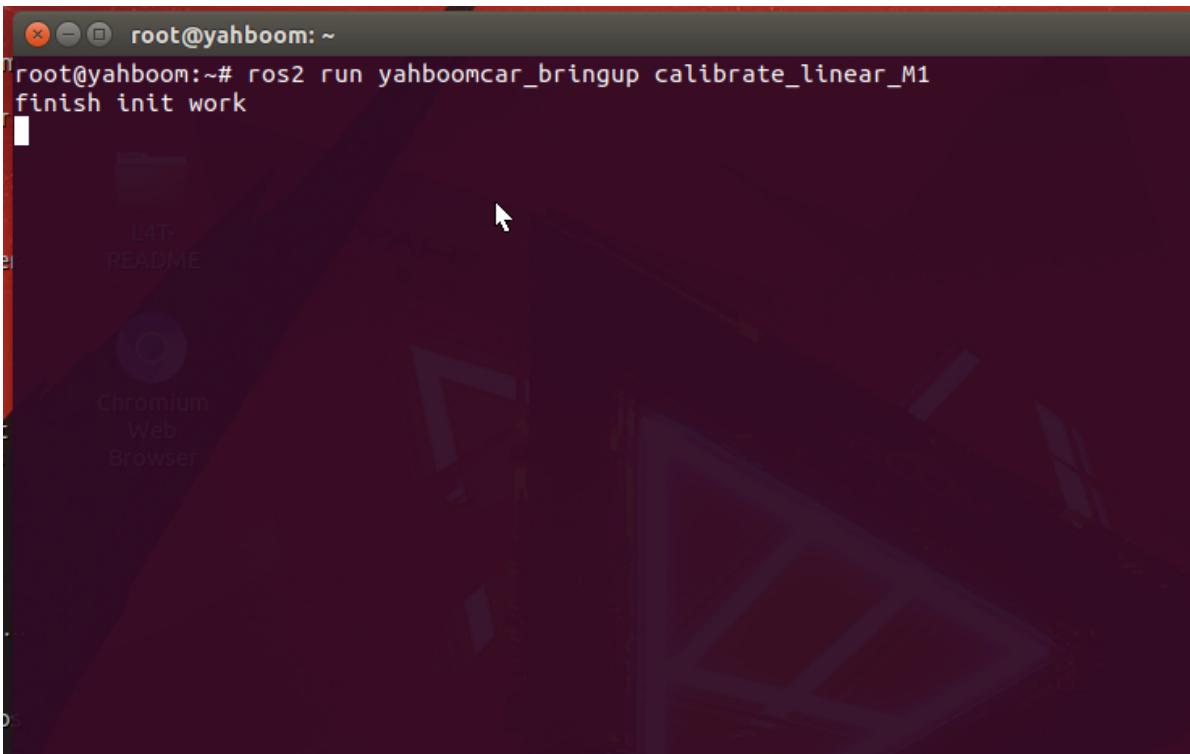
Start chassis data, input in terminal,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_M1_launch.py
```

```
root@raspberrypi:/  
File Edit Tabs Help  
got segment base_link  
[robot_state_publisher-2] [INFO] [1755157941.620678910] [robot_state_publisher]:  
got segment camera_link  
[robot_state_publisher-2] [INFO] [1755157941.620688336] [robot_state_publisher]:  
got segment imu_link  
[robot_state_publisher-2] [INFO] [1755157941.620698873] [robot_state_publisher]:  
got segment laser  
[robot_state_publisher-2] [INFO] [1755157941.620707650] [robot_state_publisher]:  
got segment left_front_wheel_joint  
[robot_state_publisher-2] [INFO] [1755157941.620717132] [robot_state_publisher]:  
got segment left_rear_wheel_hinge  
[robot_state_publisher-2] [INFO] [1755157941.620726391] [robot_state_publisher]:  
got segment left_steering_hinge_joint  
[robot_state_publisher-2] [INFO] [1755157941.620734965] [robot_state_publisher]:  
got segment right_front_wheel_joint  
[robot_state_publisher-2] [INFO] [1755157941.620745576] [robot_state_publisher]:  
got segment right_rear_wheel_hinge  
[robot_state_publisher-2] [INFO] [1755157941.620755039] [robot_state_publisher]:  
got segment right_steering_hinge_joint  
[joint_state_publisher-1] [INFO] [1755157942.125945703] [joint_state_publisher]:  
Waiting for robot_description to be published on the robot_description topic...  
[imu_filter_madgwick_node-5] [INFO] [1755157942.341251881] [imu_filter_madgwick]:  
First IMU message received.
```

```
ros2 run yahboomcar_bringup calibrate_linear_M1
```

The image below was successfully displayed.



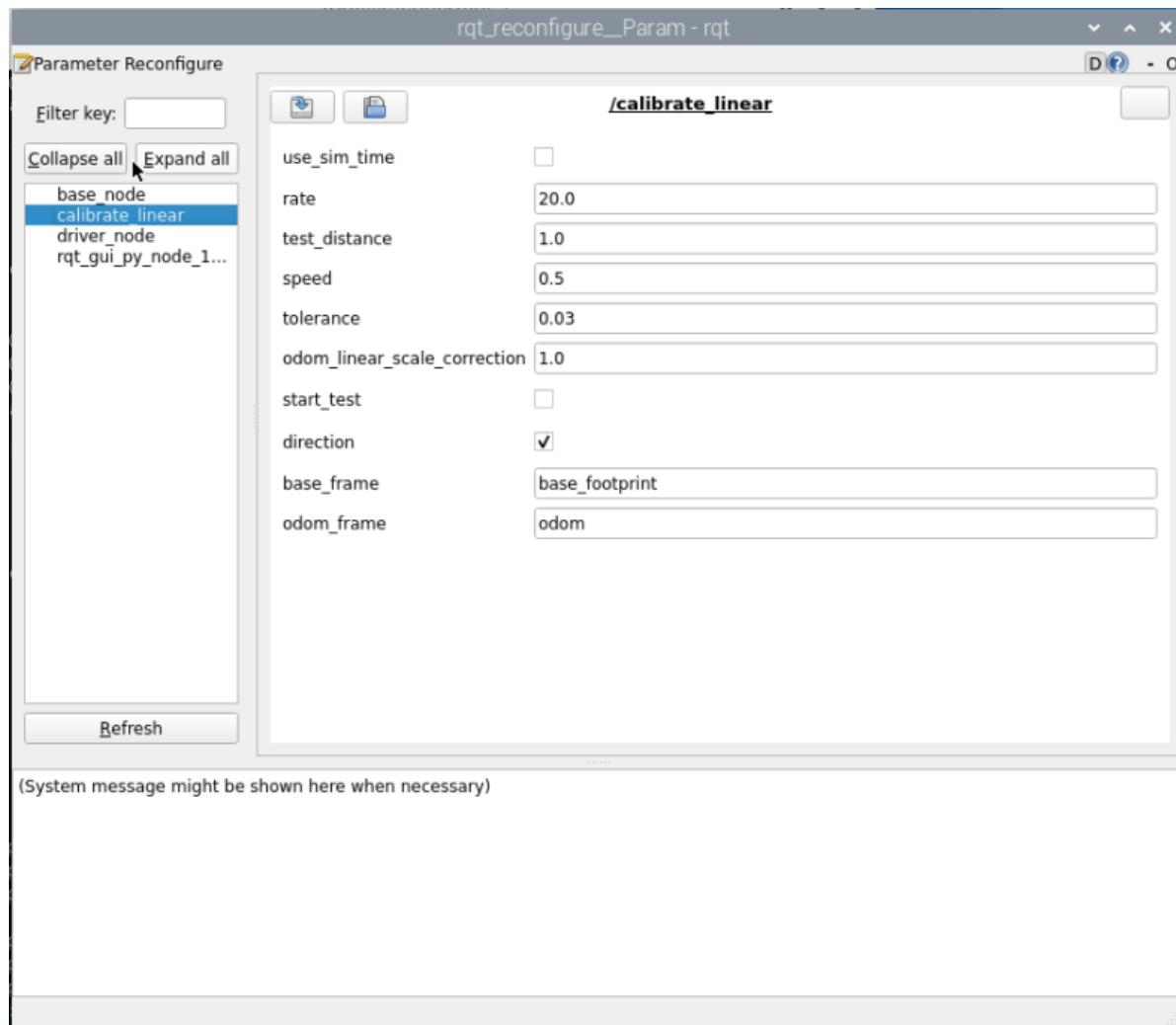
If you encounter an error message indicating no TF transformation during runtime, simply press **Ctrl+C** to exit the program and then run it again.

```
[root@raspberrypi ~]# rosrun yahboomcar_ros2_ws yahboomcar_ws/install/yahboomcar_bringup/lib/yahboomcar_bringup/calibrate_linear_A1.py
File "/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_bringup/lib/yahboomcar_bringup/calibrate_linear_A1", line 33, in <module>
    sys.exit(load_entry_point('yahboomcar-bringup==0.0.0', 'console_scripts', 'calibrate_linear_A1')())
File "/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_bringup/lib/python3.10/site-packages/yahboomcar_bringup/calibrate_linear_A1.py", line 148, in main
    rclpy.spin(class_calibratelinear)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/_init_.py", line 226, in spin
    executor.spin_once()
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/executors.py", line 751, in spin_once
    self._spin_once_implementation(timeout_sec)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/executors.py", line 748, in _spin_once_implementation
    raise handler.exception()
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/task.py", line 254, in __call__
    self._handler.send(None)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/executors.py", line 447, in handler
    await call_coroutine(entity, arg)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/executors.py", line 361, in _execute_timer
    await await_or_execute(tmr.callback)
File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/executors.py", line 107, in await_or_execute
    return callback(*args)
File "/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_bringup/lib/python3.10/site-packages/yahboomcar_bringup/calibrate_linear_A1.py", line 114, in on_timer
    self.x_start = self.get_position().transform.translation.x
File "/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_bringup/lib/python3.10/site-packages/yahboomcar_bringup/calibrate_linear_A1.py", line 136, in get_position
    trans = self.tf_buffer.lookup_transform(self.odom_frame, self.base_frame, now)
File "/opt/ros/humble/lib/python3.10/site-packages/tf2_ros/buffer.py", line 136, in lookup_transform
    return self.lookup_transform_core(target_frame, source_frame, time)
tf2.LookupException: "odom" passed to lookupTransform argument target_frame does not exist.
[ INFO] [1628948811.110000000]: Process exited with failure -1
[root@raspberrypi ~]#
```

Open the dynamic parameter tuner and run the following in the terminal:

```
ros2 run rqt_reconfigure rqt_reconfigure
```

Click the **calibrate_linear** node in the left-hand node options:



Note: The above nodes may not be visible initially. Clicking "Refresh" will display all nodes. The displayed **calibrate_linear** node is the node for calibrating linear velocity.

The RQT interface parameters are explained below:

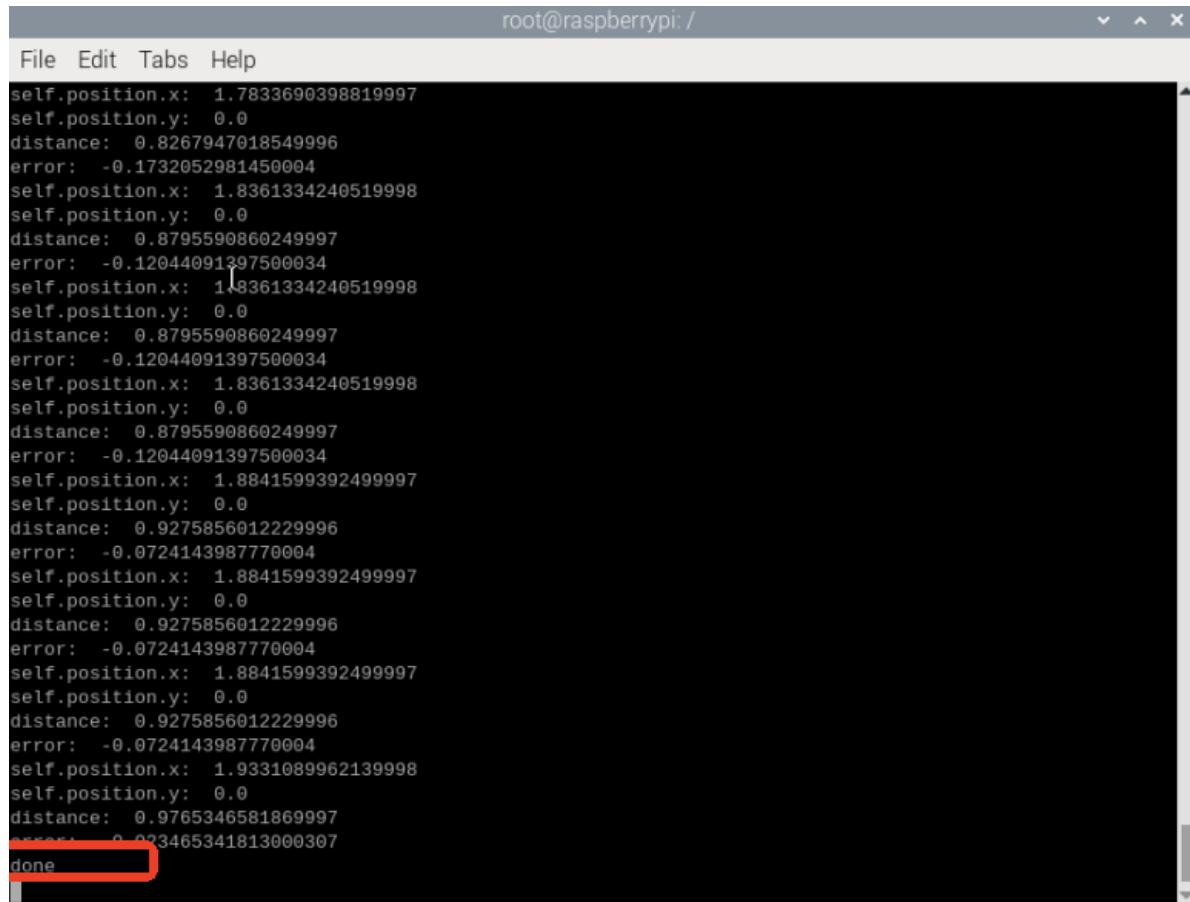
- **test_distance:** The distance for calibration testing; here, the test moves forward 1 meter.
- **speed:** The magnitude of the linear velocity.
- **tolerance:** The tolerance for error.
- **odom_linear_scale_correction:** The linear velocity scale factor. This value is modified if the test results are unsatisfactory.
- **start_test:** Test switch.
- **direction:** Can be ignored. This value is used for the McLaren wheel structure vehicle. Modifying it allows calibration of the linear velocity for left and right movement.
- **base_frame:** The name of the base coordinate system.
- **odom_frame:** The name of the odometry coordinate system.

2.2 Starting Calibration

In the rqt_reconfigure interface, select the calibration_linear node (if it's not displayed, click **Refresh**).

Select a reference of known length on the ground (measuring tape, tile, etc.); Change **test_distance** to the actual test distance. Here, we take a 1-meter test distance as an example. Check the **start_test** box to start calibration.

After clicking start_test, the car will listen to the TF transform of base_footprint and odom, calculate the theoretical distance the car should travel, and wait until the error is less than tolerance. Issue a stop command and print "done" on the terminal. If the actual distance traveled by the car is less than 1 meter, appropriately increase the **odom_linear_scale_correction** parameter. After modification, click a blank area, click start_test again to reset start_test, and then click start_test again to complete the calibration. Modifying other parameters is the same; you need to click a blank area to write the modified parameters. Record the final **odom_linear_scale_correction** parameter.



```
root@raspberrypi:/home/pi/yahboomcar_ws/src/yahboomcar_bringup/launch
```

```
File Edit Tabs Help
```

```
self.position.x: 1.7833690398819997
self.position.y: 0.0
distance: 0.8267947018549996
error: -0.1732052981450004
self.position.x: 1.8361334240519998
self.position.y: 0.0
distance: 0.8795590860249997
error: -0.12044091397500034
self.position.x: 1.8361334240519998
self.position.y: 0.0
distance: 0.8795590860249997
error: -0.12044091397500034
self.position.x: 1.8361334240519998
self.position.y: 0.0
distance: 0.8795590860249997
error: -0.12044091397500034
self.position.x: 1.8841599392499997
self.position.y: 0.0
distance: 0.9275856012229996
error: -0.0724143987770004
self.position.x: 1.8841599392499997
self.position.y: 0.0
distance: 0.9275856012229996
error: -0.0724143987770004
self.position.x: 1.8841599392499997
self.position.y: 0.0
distance: 0.9275856012229996
error: -0.0724143987770004
self.position.x: 1.9331089962139998
self.position.y: 0.0
distance: 0.9765346581869997
error: 0.023465341813000307
done
```

After testing, remember the value of `odom_linear_scale_correction` and modify the `linear_scale_x` parameter in `yahboomcar_bringup_M1_launch.py`.

Path:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_bringup/launch/yahboomcar_bringup_M1_launch.py
```

```

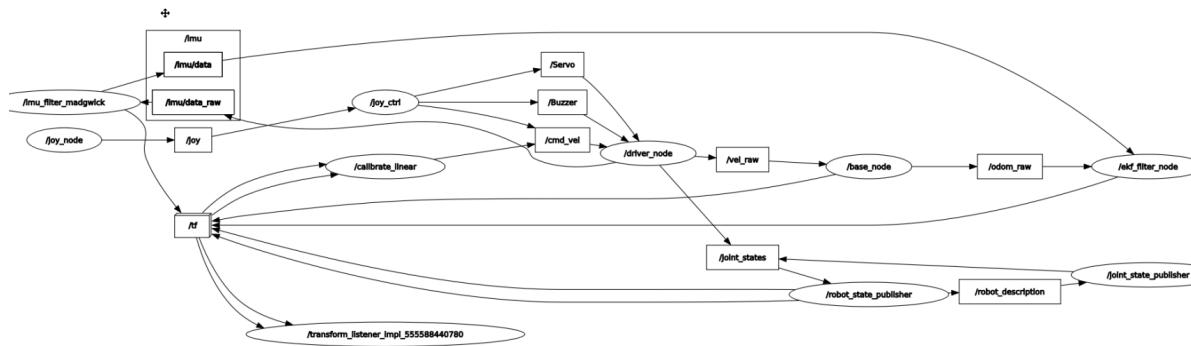
def generate_launch_description():
    rviz_node = Node(
        arguments=['-d', LaunchConfiguration('rvizconfig')],
    )
    imu_filter_config = os.path.join(
        get_package_share_directory('yahboomcar_bringup'),
        'param',
        'imu_filter_param.yaml'
    )
    driver_node = Node(
        package='yahboomcar_bringup',
        executable='Mcnamu_driver_M1',
    )
    base_node = Node(
        package='yahboomcar_base_node',
        executable='base_node_M1',
        # 当使用ekf融合时，该tf有ekf发布
        parameters=[{
            'pub_odom_tf': LaunchConfiguration('pub_odom_tf'),
            'linear_scale_x': 1.0,
            'linear_scale_y': 1.0,
        }]
    )
    imu_filter_node = Node(
        package='imu_filter_madgwick',
        executable='imu_filter_madgwick_node',
        parameters=[imu_filter_config]
    )
    ekf_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(os.path.join(
            get_package_share_directory('robot_localization'), 'launch'),
            '/ekf_M1.launch.py'))
    )
    yahboom_joy_node = Node(
        package='yahboomcar_ctrl',
        executable='yahboom_joy_M1',
    )
    joy_node = Node(
        package='joy',
        executable='joy_node',
    )

```

3. View the Node Relationship Graph

Open a terminal and enter the command:

```
ros2 run rqt_graph rqt_graph
```



In the node relationship graph above:

- The **imu_filter** node is responsible for filtering the raw IMU data from the chassis **/imu/data** and publishing the filtered data **/imu/data**.
- The **/ekf_filter_node** node subscribes to the raw IMU data from the chassis **/odom_raw** and the filtered IMU data **/imu/data**, performs data fusion, and publishes the **/odom** topic.
- The **calibrate_linear** node listens for the TF transformation of odom->base_footprint and publishes the **/cmd_vel** topic to control the movement of the robot chassis.

4. Core Source Code Analysis

This program primarily utilizes TensorFlow to monitor coordinate transformations. By listening to the coordinate changes between `base_footprint` and `odom`, the robot learns "how far I've traveled/how many degrees I've turned."

Code path,

```
~/yahboomcar_ross2_ws/yahboomcar_ross/src/yahboomcar_bringup/yahboomcar_bringup/calibrate_angular_M1.py
```

The implementation for monitoring TensorFlow coordinate transformations is the `get_position` method in the `CalibrateLinear` class:

```
def get_position(self):
    try:
        now = rclpy.time.Time()
        transform = self.tf_buffer.lookup_transform(
            self.base_frame,
            self.odom_frame,
            now,
            timeout=rclpy.duration.Duration(seconds=1.0))
    return transform

    except (LookupException, ConnectivityException, ExtrapolationException):
        self.get_logger().info('transform not ready')
        raise
```

The implementation of determining the robot chassis displacement and controlling its movement is the `on_timer` method (timer callback function) in the `CalibrateLinear` class:

```
def on_timer(self):
    move_cmd = Twist()
    #self.get_param()
    self.start_test =
    self.get_parameter('start_test').get_parameter_value().bool_value
    self.odom_linear_scale_correction =
    self.get_parameter('odom_linear_scale_correction').get_parameter_value().double_
    value
    self.rate = self.get_parameter('rate').get_parameter_value().double_value
    self.test_distance =
    self.get_parameter('test_distance').get_parameter_value().double_value
    self.direction =
    self.get_parameter('direction').get_parameter_value().double_value
    self.tolerance =
    self.get_parameter('tolerance').get_parameter_value().double_value
    self.speed = self.get_parameter('speed').get_parameter_value().double_value
    if self.start_test:
        self.position.x = self.get_position().transform.translation.x
        self.position.y = self.get_position().transform.translation.y
        print("self.position.x: ",self.position.x)
        print("self.position.y: ",self.position.y)
        distance = sqrt(pow((self.position.x - self.x_start), 2) +
                        pow((self.position.y - self.y_start), 2))
        distance *= self.odom_linear_scale_correction
        print("distance: ",distance)
        error = distance - self.test_distance
        print("error: ",error)
```

```
#start = time()
if not self.start_test or abs(error) < self.tolerance:
    self.start_test =
rcpy.parameter.Parameter('start_test', rcpy.Parameter.Type.BOOL, False)
    all_new_parameters = [self.start_test]
    self.set_parameters(all_new_parameters)

    print("done")
else:
    move_cmd.linear.x = copysign(self.speed, -1 * error)
    '''if self.direction:
        print("x")
        move_cmd.linear.x = copysign(self.speed, -1 * error)
    else:
        move_cmd.linear.y = copysign(self.speed, -1 * error)
        print("y")'''
    self.cmd_vel.publish(move_cmd)

else:
    self.x_start = self.get_position().transform.translation.x
    self.y_start = self.get_position().transform.translation.y

    self.cmd_vel.publish(Twist())
```