robotics – Robotics

Robotics module for the Pybricks API.

class **DriveBase**(*left_motor*, *right_motor*, *wheel_diameter*, *axle_track*)

A robotic vehicle with two powered wheels and an optional support wheel or caster.

By specifying the dimensions of your robot, this class makes it easy to drive a given distance in millimeters or turn by a given number of degrees.

Positive distances and drive speeds mean driving **forward**. **Negative** means **backward**.

Positive angles and turn rates mean turning **right**. **Negative** means **left**. So when viewed from the top, positive means clockwise and negative means counterclockwise.

Parameters:
- **left_motor** (*Motor*) – The motor that drives the left wheel.
- **right_motor** (*Motor*) – The motor that drives the right wheel.
- **wheel_diameter** (dimension: mm) – Diameter of the wheels.
- **axle_track** (dimension: mm) – Distance between the points where both wheels touch the ground.

Driving for a given distance or by an angle

Use the following commands to drive a given distance, or turn by a given angle.

This is measured using the internal rotation sensors. Because wheels may slip while moving, the traveled distance and angle are only estimates.

straight(*distance*)

Drives straight for a given distance and then stops.

Parameters: **distance** (distance: mm) – Distance to travel.

turn(*angle*)

Turns in place by a given angle and then stops.

Parameters: **angle** (angle: deg) – Angle of the turn.

settings(*straight_speed*, *straight_acceleration*, *turn_rate*, *turn_acceleration*)

Configures the speed and acceleration used by `straight()` and `turn()`.

If you give no arguments, this returns the current values as a tuple.

You can only change the settings while the robot is stopped. This is either before you begin driving or after you call `stop()`.

Parameters:
- **straight_speed** (speed: mm/s) – Speed of the robot during `straight()`.
- **straight_acceleration** (linear acceleration: mm/s/s) – Acceleration and deceleration of the robot at the start and end of `straight()`.
- **turn_rate** (rotational speed: deg/s) – Turn rate of the robot during `turn()`.
- **turn_acceleration** (rotational acceleration: deg/s/s) – Angular acceleration and deceleration of the robot at the start and end of `turn()`.

Drive forever

Use `drive()` to begin driving at a desired speed and steering.

It keeps going until you use `stop()` or change course by using `drive()` again. For example, you can drive until a sensor is triggered and then stop or turn around.

drive(*drive_speed*, *turn_rate*)

Starts driving at the specified speed and turn rate. Both values are measured at the center point between the wheels of the robot.

Parameters:
- **drive_speed** (speed: mm/s) – Speed of the robot.
- **turn_rate** (rotational speed: deg/s) – Turn rate of the robot.

stop()

Stops the robot by letting the motors spin freely.

Measuring

distance()

Gets the estimated driven distance.

Returns: Driven distance since last reset.

Return type: distance: mm

angle()

Gets the estimated rotation angle of the drive base.

Returns: Accumulated angle since last reset.

Return type: angle: deg

`state`()

Gets the state of the robot.

This returns the current `distance()`, the drive speed, the `angle()`, and the turn rate.

Returns: Distance, drive speed, angle, turn rate

Return type: (distance: mm, speed: mm/s, angle: deg, rotational speed: deg/s)

`reset`()

Resets the estimated driven distance and angle to 0.

Measuring and validating the robot dimensions

As a first estimate, you can measure the `wheel_diameter` and the `axle_track` with a ruler. Because it is hard to see where the wheels effectively touch the ground, you can estimate the `axle_track` as the distance between the midpoint of the wheels.

In practice, most wheels compress slightly under the weight of your robot. To verify, make your robot drive 1000 mm using `my_robot.straight(1000)` and measure how far it really traveled. Compensate as follows:

- If your robot drives **not far enough**, **decrease** the `wheel_diameter` value slightly.
- If your robot drives **too far**, **increase** the `wheel_diameter` value slightly.

Motor shafts and axles bend slightly under the load of the robot, causing the ground contact point of the wheels to be closer to the midpoint of your robot. To verify, make your robot turn 360 degrees using `my_robot.turn(360)` and check that it is back in the same place:

- If your robot turns **not far enough**, **increase** the `axle_track` value slightly.
- If your robot turns **too far**, **decrease** the `axle_track` value slightly.

When making these adjustments, always adjust the `wheel_diameter` first, as done above. Be sure to test both turning and driving straight after you are done.

Using the DriveBase motors individually

Suppose you make a `DriveBase` object using two `Motor` objects called `left_motor` and `right_motor`. You **cannot** use these motors individually while the DriveBase is **active**.

The DriveBase is active if it is driving, but also when it is actively holding the wheels in place after a `straight()` or `turn()` command. To deactivate the `DriveBase`, call `stop()`.

Advanced Settings

The `settings()` method is used to adjust commonly used settings like the default speed and acceleration for straight maneuvers and turns. Use the following attributes to adjust more advanced control setttings.

You can only change the settings while the robot is stopped. This is either before you begin driving or after you call `stop()`.

> **`distance_control`**
>
> The traveled distance and drive speed are controlled by a PID controller. You can use this attribute to change its settings. See The Control Class for an overview of available methods.

> **`heading_control`**
>
> The robot turn angle and turn rate are controlled by a PID controller. You can use this attribute to change its settings. See The Control Class for an overview of available methods.