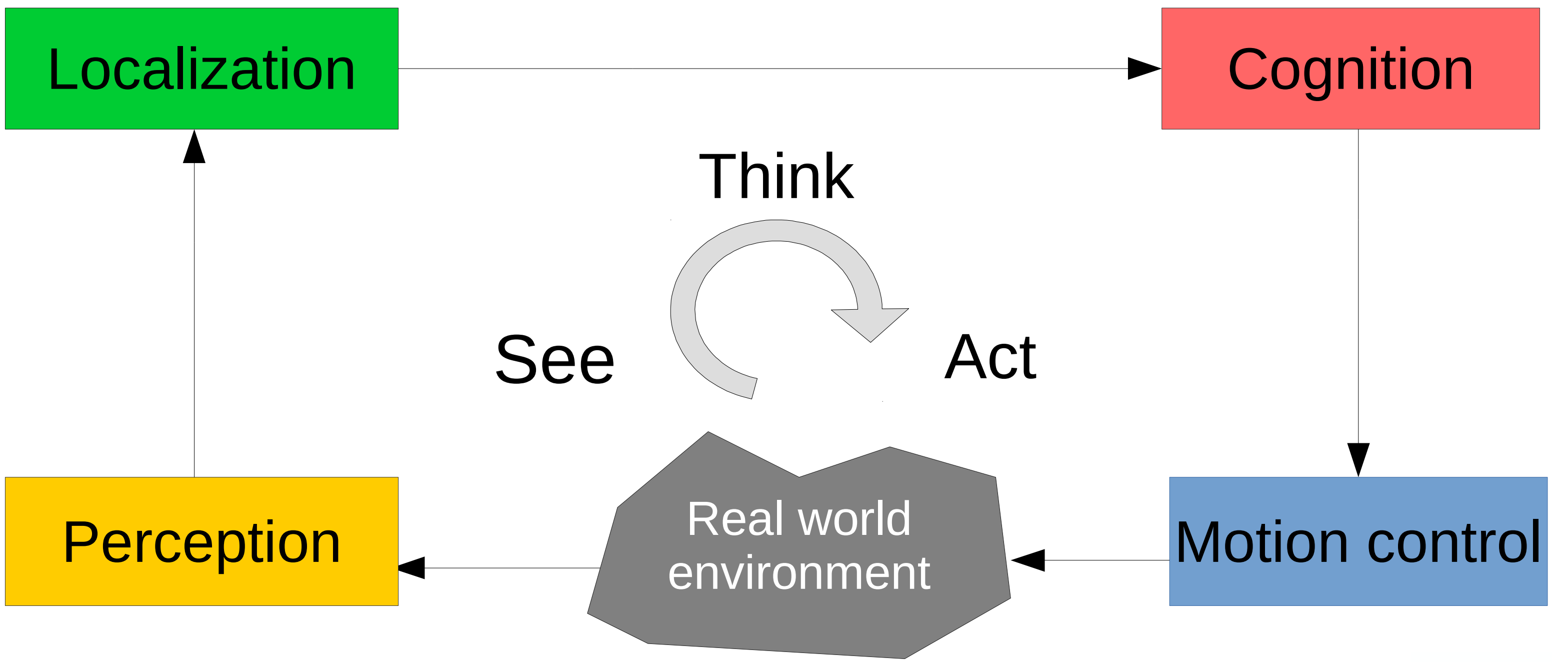
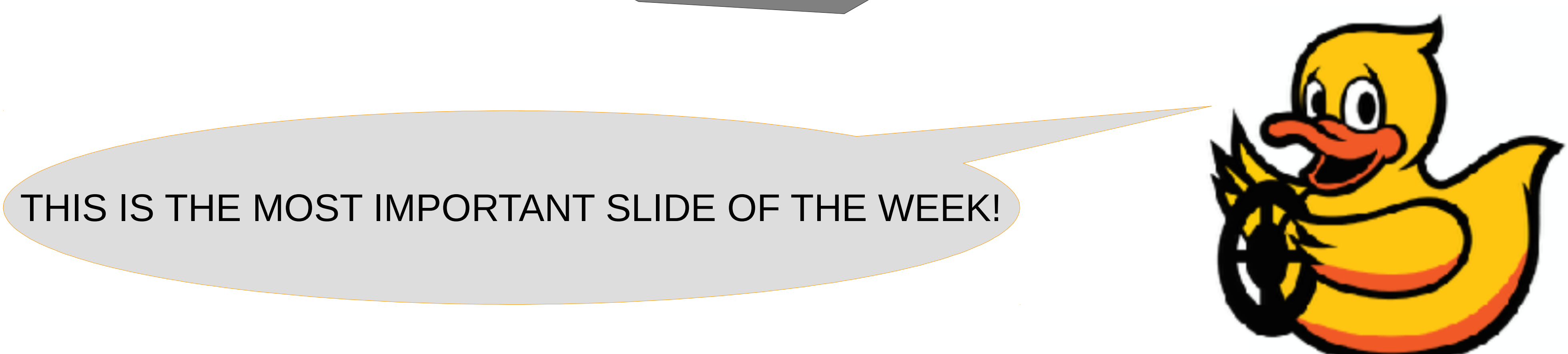
Autonomous mobile robot

See – think – act



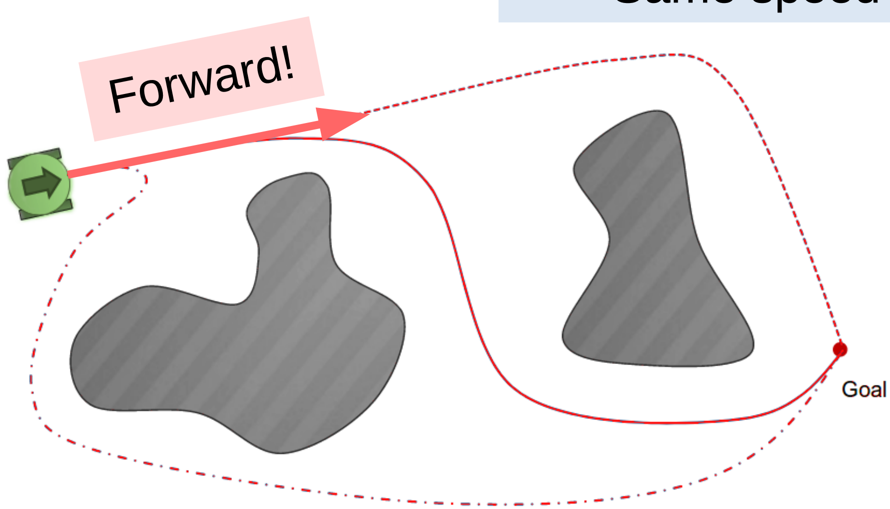


Autonomous Mobile Robots – R. Siegwart, I. Nourbakhsh, D. Scaramuzza

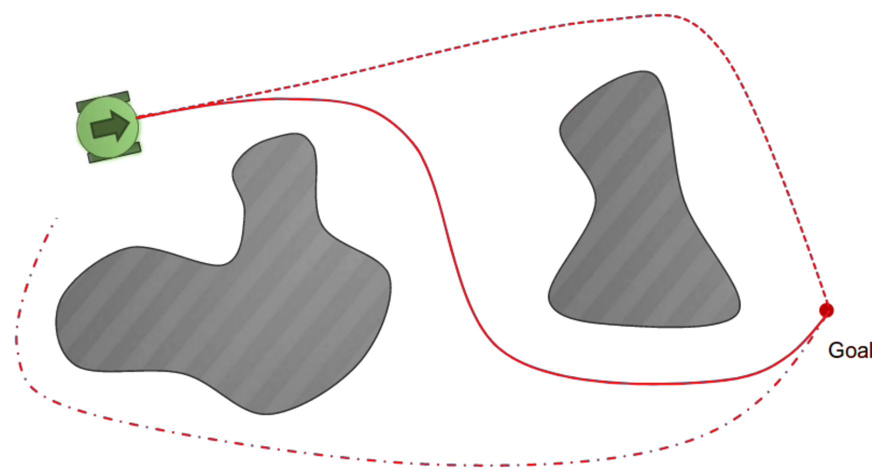
Motion control

Movement command → Movement execution → Actuator commands → Acting

Wheels have to turn:
- Forward
- Same speed for both



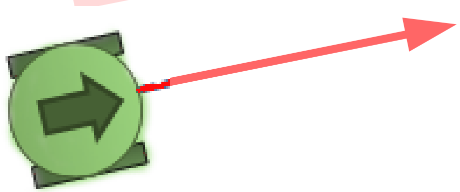
Wheels, turn!



Motion control - joystick

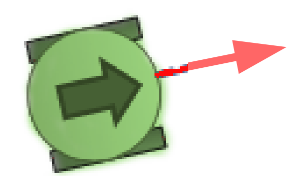
Joystick → Movement execution → Actuator commands → Acting

Forward!



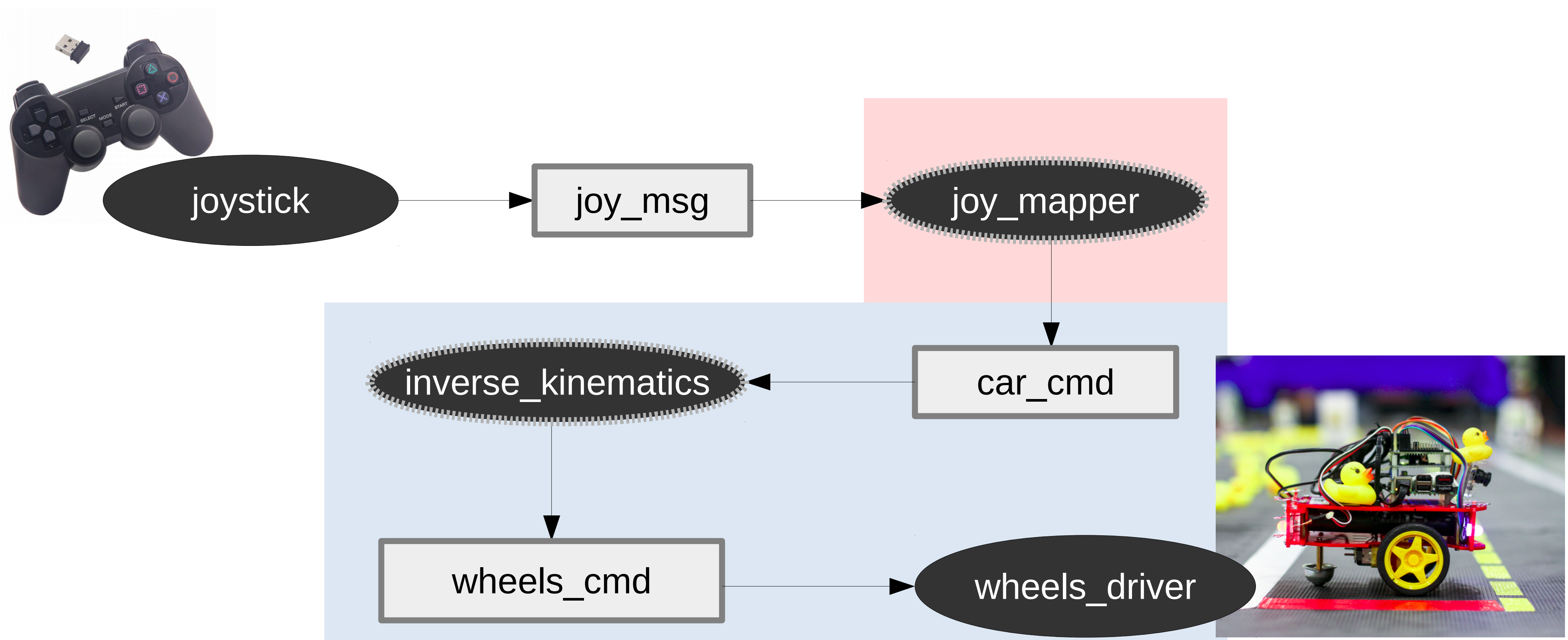
Wheels have to turn:
- Forward
- Same speed for both

Wheels, turn!



Drive with a joystick

Behind the scene:



Step 1: Joy_mapper

What buttons are being pressed?

At which linear and angular velocities should the car go?

Buttons and axis

Structure of a joy_msg

2 tables

Buttons: joy_msg.buttons

Index	Button name on the actual controller
0	A
1	B
2	X
3	Y
4	LB
5	RB
6	back
7	start
8	power
9	Button stick left
10	Button stick right

Axis: joy_msg.axis

Index	Axis name on the actual controller
0	Left/Right Axis stick left
1	Up/Down Axis stick left
2	Left/Right Axis stick right
3	Up/Down Axis stick right
4	RT
5	LT
6	cross key left/right
7	cross key up/down

For buttons:
- 0 if not pressed
- 1 if pressed

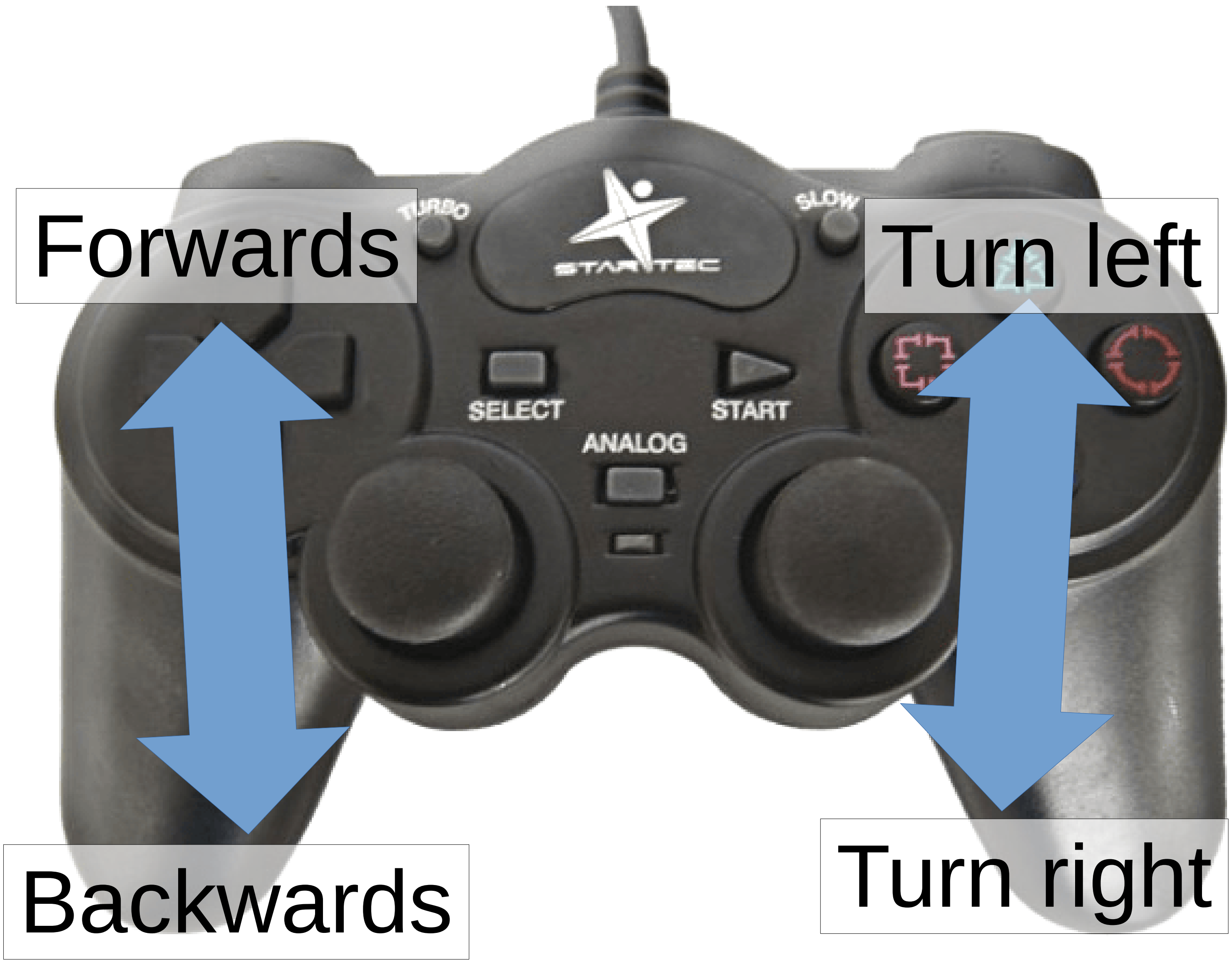
For axis:
- 0 if not touched
- 1 if up
- -1 if down

Structure of a car_cmd

Linear velocity: car_cmd.v

Angular velocity: car_cmd.omega

Driving the car



Function to complete:

publishControl(self)

1) Change the 0's on lines 74 and 75

Linear velocity v :

0	if left joystick is not touched
v_gain	if left joystick is up
$- v\_gain$	if left joystick is down

Angular velocity omega :

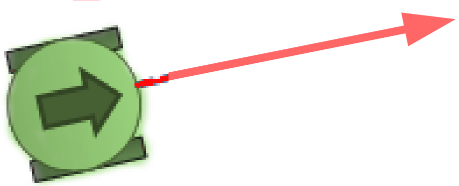
0	if right joystick is not touched
$omega\_gain$	if right joystick is up
$- omega\_gain$	if right joystick is down

Let’s try something else!

Motion control - joystick

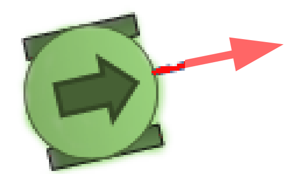
Joystick → Movement execution → Actuator commands → Acting

Forward!



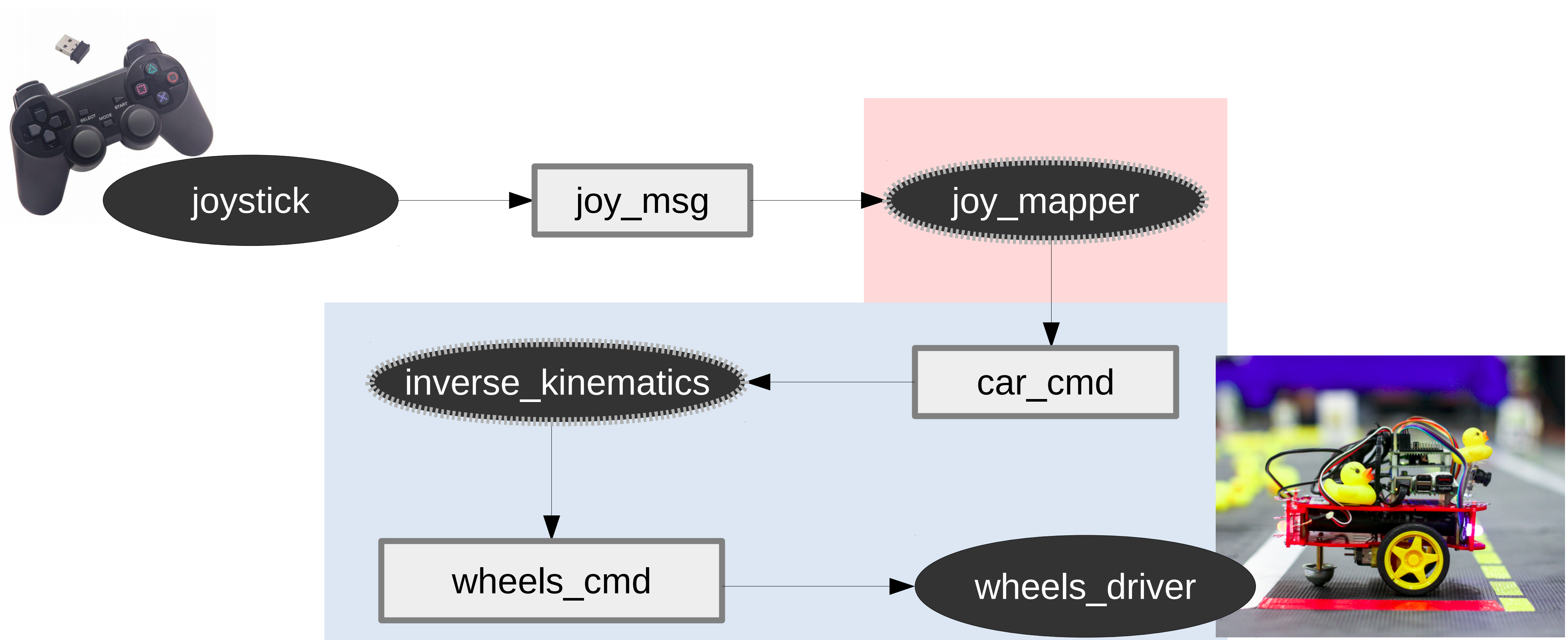
Wheels have to turn:
- Forward
- Same speed for both

Wheels, turn!



Drive with a joystick

Behind the scene:



Step 2: Inverse_kinematics

At which linear and angular velocities should the car go?

At which velocity should each wheel motor turn?

Structure of a car_cmd

Linear velocity: car_cmd.v

Angular velocity: car_cmd.omega

Structure of a wheel_cmd

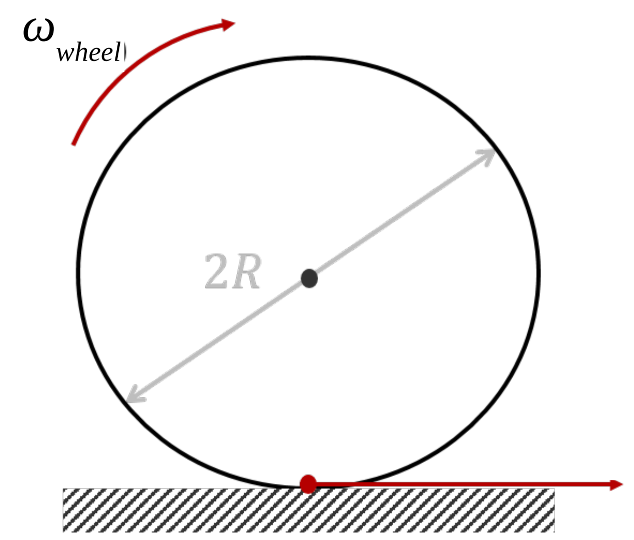
duty cycle right wheel: wheels_cmd.vel_right

duty cycle left wheel: wheels_cmd.vel_left

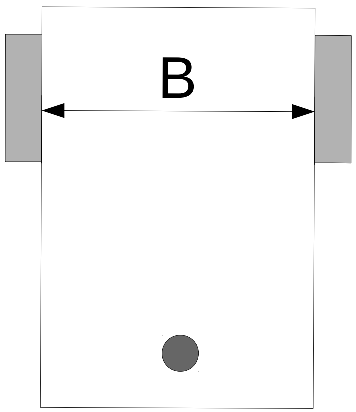
Wheel angular velocity

Car linear velocity : v_{car}
Car angular velocity : ω_{car}
Wheel radius : R
Baseline : B

Wheel angular velocity : ω_{wheel}



$$\omega_{wheel} = \frac{v_{car} \pm (\omega_{car} \cdot B/2)}{R}$$



\+ if right wheel
\- if left wheel

Your turn!

car_cmd_callback(self, msg_car_cmd)

<u>Code the wheel velocities</u> lines 171 and 172

Reminder:

$$\omega_{wheel} = \frac{v_{car} \pm \left(\omega_{car} \cdot B/2\right)}{R}$$

+ if right wheel
- if left wheel

Structure of a car_cmd

Linear velocity: car_cmd.v

Angular velocity: car_cmd.omega

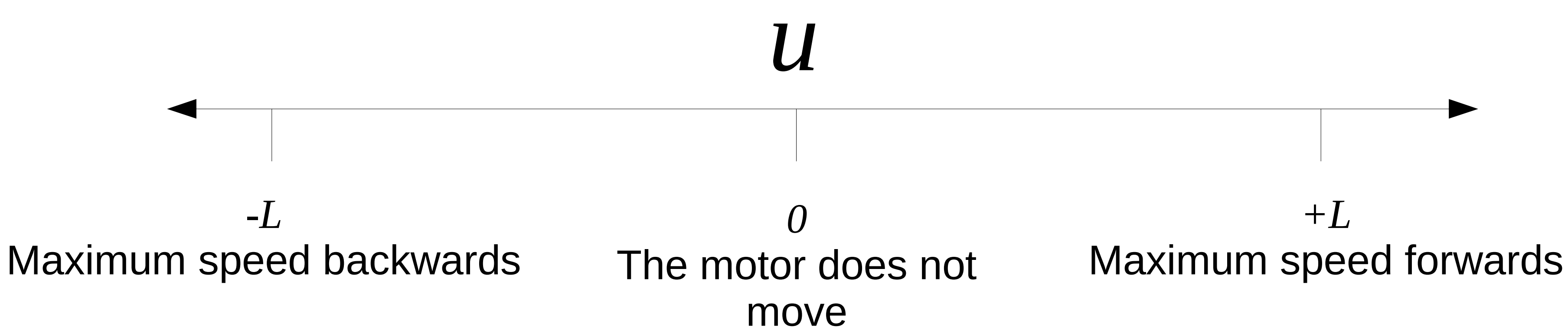
Structure of a wheel_cmd

duty cycle right wheel: wheels_cmd.vel_right

duty cycle left wheel: wheels_cmd.vel_left

What is duty cycle u?

Command to the motor:



Gain and trim

Notice what is happening next in the code:

```
# assuming same motor constants k for both motors
    k_r = self.k
    k_l = self.k

# adjusting k by gain and tri
    k_r_inv = (self.gai
    k_l_inv = (self.gain

# conversion from motor rotation rate to dut ycle
    u_r = omega_r * k_r_inv
    u_l = omega_l * k_l_inv
```

But, what are *g*, *t* and *k*?

$$u_r = \omega_r * (g + t) / k$$

$$u_l = \omega_l * (g - t) / k$$

Gain and trim

$$u_r = \omega_r * (g + t) / k$$
$$u_l = \omega_l * (g - t) / k$$

K : motor constant. Translates *ω* to *u* with *u* = *ω/k*

g : gain. Allows to control the general speed.

t : trim. Allows to calibrate the two motors.

Calibrate the motors?
We will have to do that!

But first, your turn!

car_cmd_callback(self, msg_car_cmd)

Constraint $u\_r\_limited$ and $u\_l\_limited$ lines 179 and 180

$$
\begin{cases}
-L & \text{if } u < -L \\
u & \text{if } -L < u < L \\
L & \text{if } u > L
\end{cases}
$$

Challenge: do it without “if”!
Hint: use min and max functions.

Motor calibration

The motors are not exactly the same!

We have to calibrate them to make sure they turn at the same speed.

$$u_r = \omega_r * (g + t) / k$$

$$u_l = \omega_l * (g - t) / k$$

We have to choose *t* so that Moose goes straight forward when told to do so!

If we increase *t*, in which direction will the bot turn?

Let’s try?