上海科技大学
ShanghaiTech University

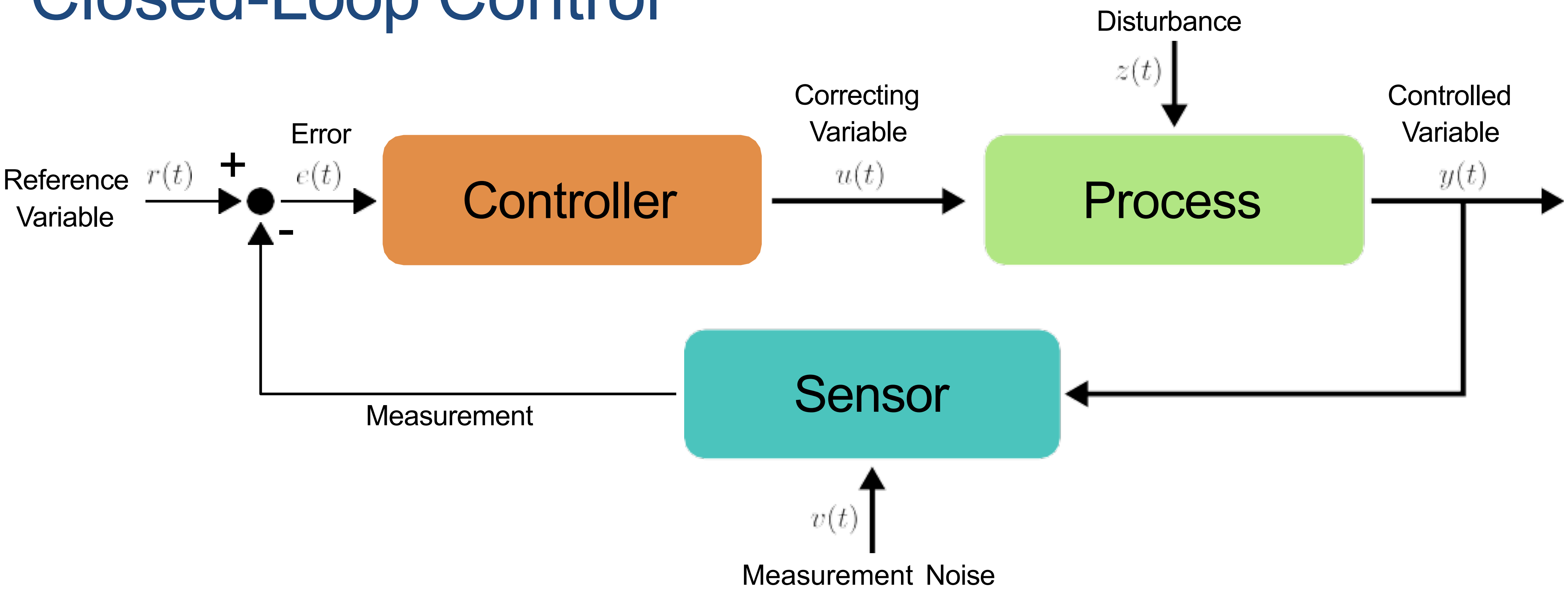
CS283: **Robotics Spring 2025: Mechatronics**

Sören Schwertfeger / **师泽仁**

ShanghaiTech University

CONTROL

Closed-Loop Control

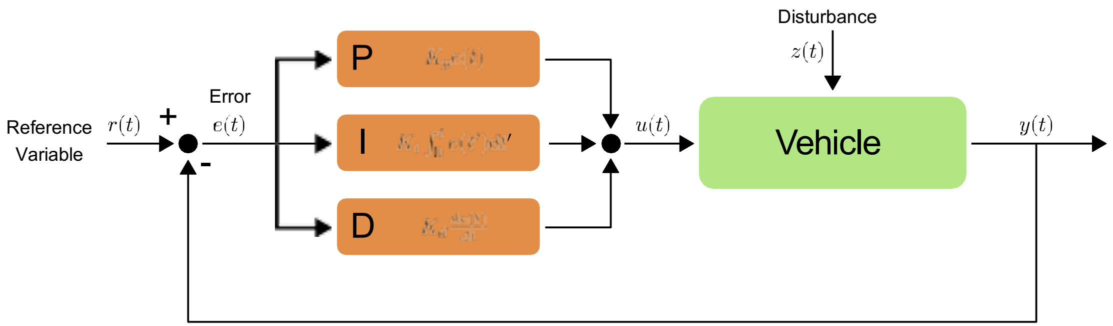


- Exploit feedback loop to minimize error between reference and measurement

Longitudinal Vehicle Control

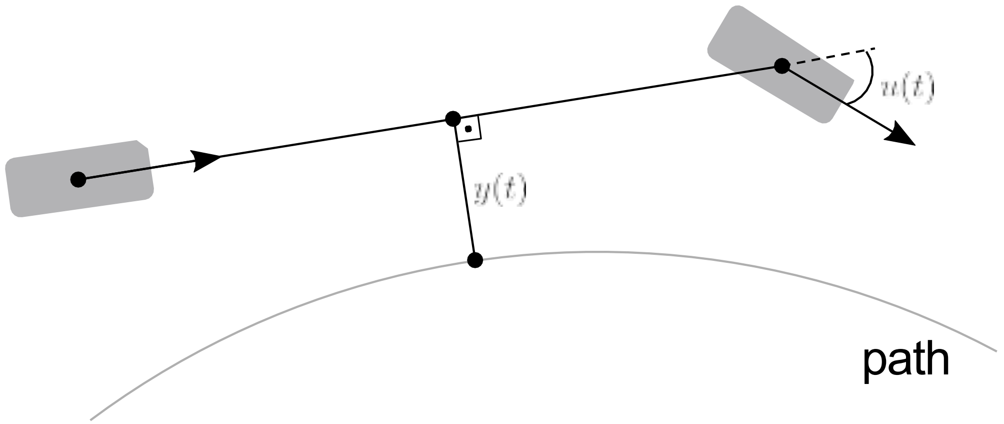
$$v(t) = v_{\text{max}} \left(1 - \exp\left(-\theta_1 \, d(t) - \theta_2\right)\right)$$

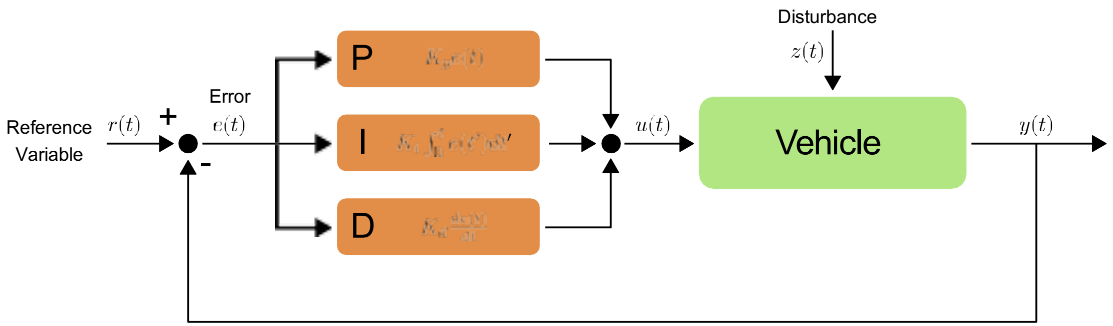
- $v(t)$: target velocity at time t
- $d(t)$: distance to preceding car



- Reference variable: $r(t) = v(t) =$ target velocity
- Correcting variable: $u(t) =$ gas/brake pedal
- Controlled variable: $y(t) =$ current velocity
- Error: $e(t) = v(t) - y(t)$

Lateral Vehicle Control





- Reference variable: $r(t) = 0 =$ no cross track error
- Correcting variable: $u(t) = \delta =$ steering angle
- Controlled variable: $y(t) =$ cross track error
- Error: $e(t) = -y(t) =$ cross track error

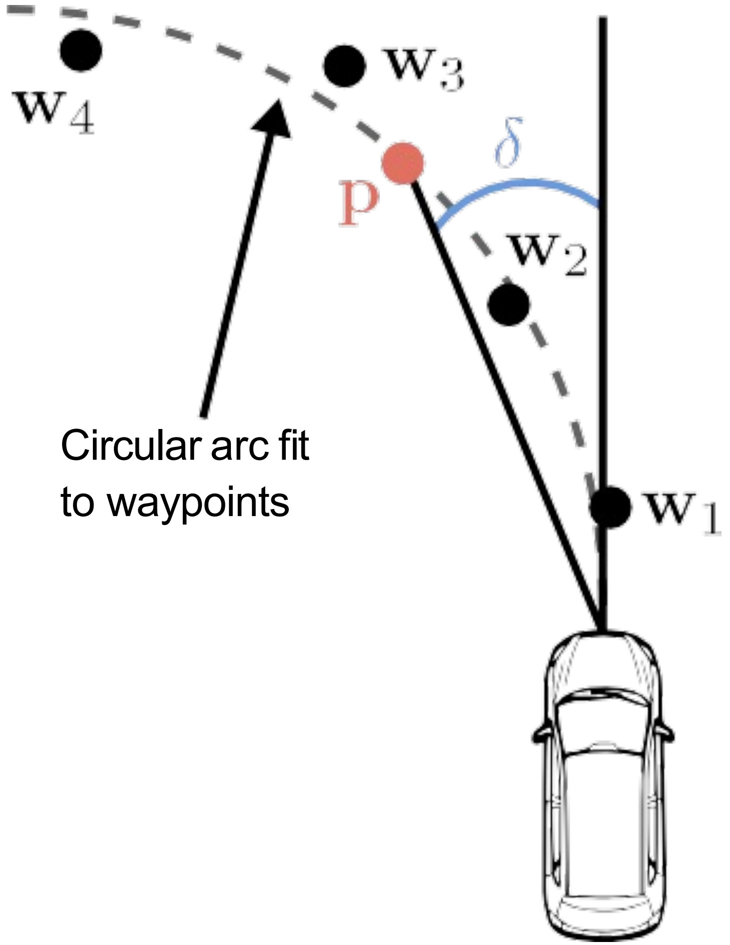
Waypoint-based Vehicle Control

- **Input:** Waypoints $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$
- **Velocity:** (Longitudinal PID control)

$$v = \frac{1}{K} \sum_{k=1}^{K} \frac{\left\| \mathbf{w}_k - \mathbf{w}_{k-1} \right\|_2}{\Delta t}$$

- **Steering angle:** (Lateral PID control)

$$\delta = \tan^{-1}\left(\frac{p_y}{p_x}\right)$$



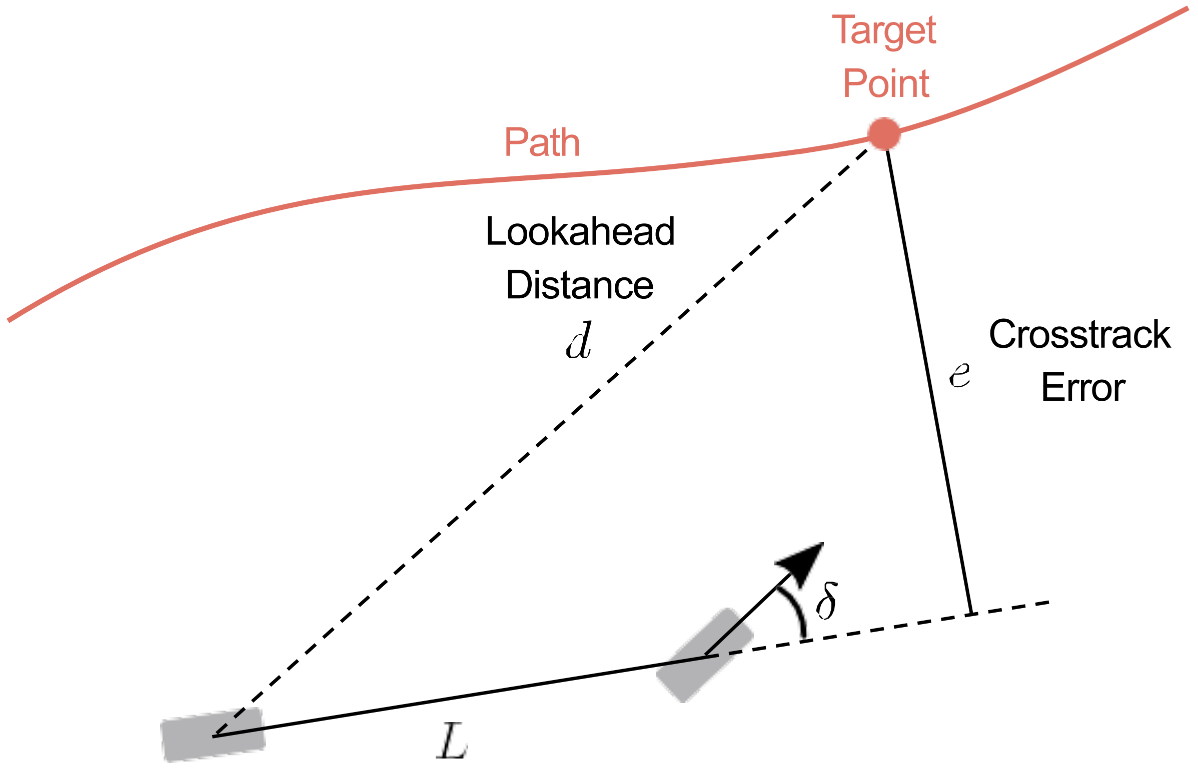
Chen, Zhou, Koltun and Krähenbühl: Learning by Cheating. CoRL, 2019.

GEOMETRIC CONTROL

Pure Pursuit Control

Goal:

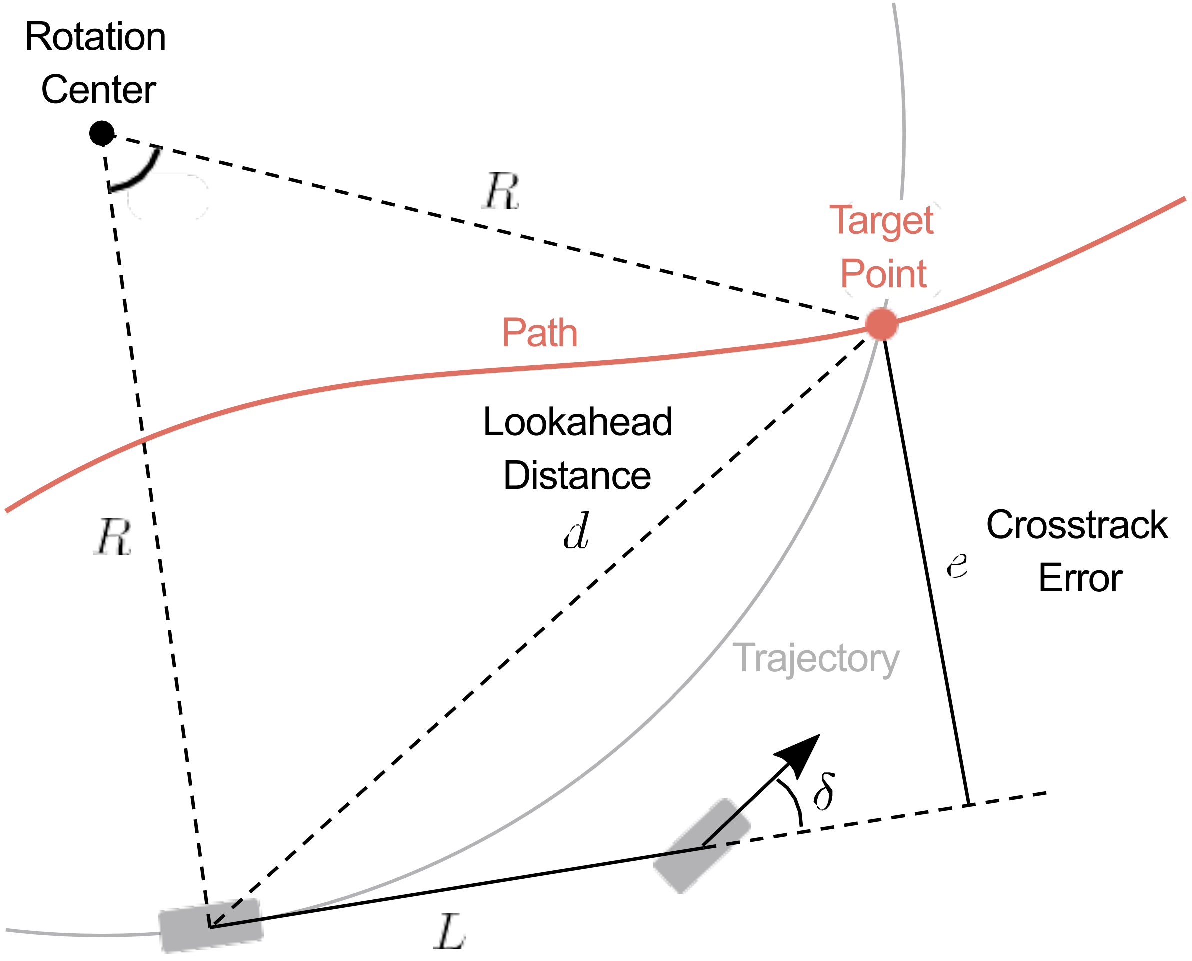
- Track **target point** at **lookahead distance** d to follow path
- Exploit geometric relationship between vehicle and path to follow



Pure Pursuit Control

Goal:

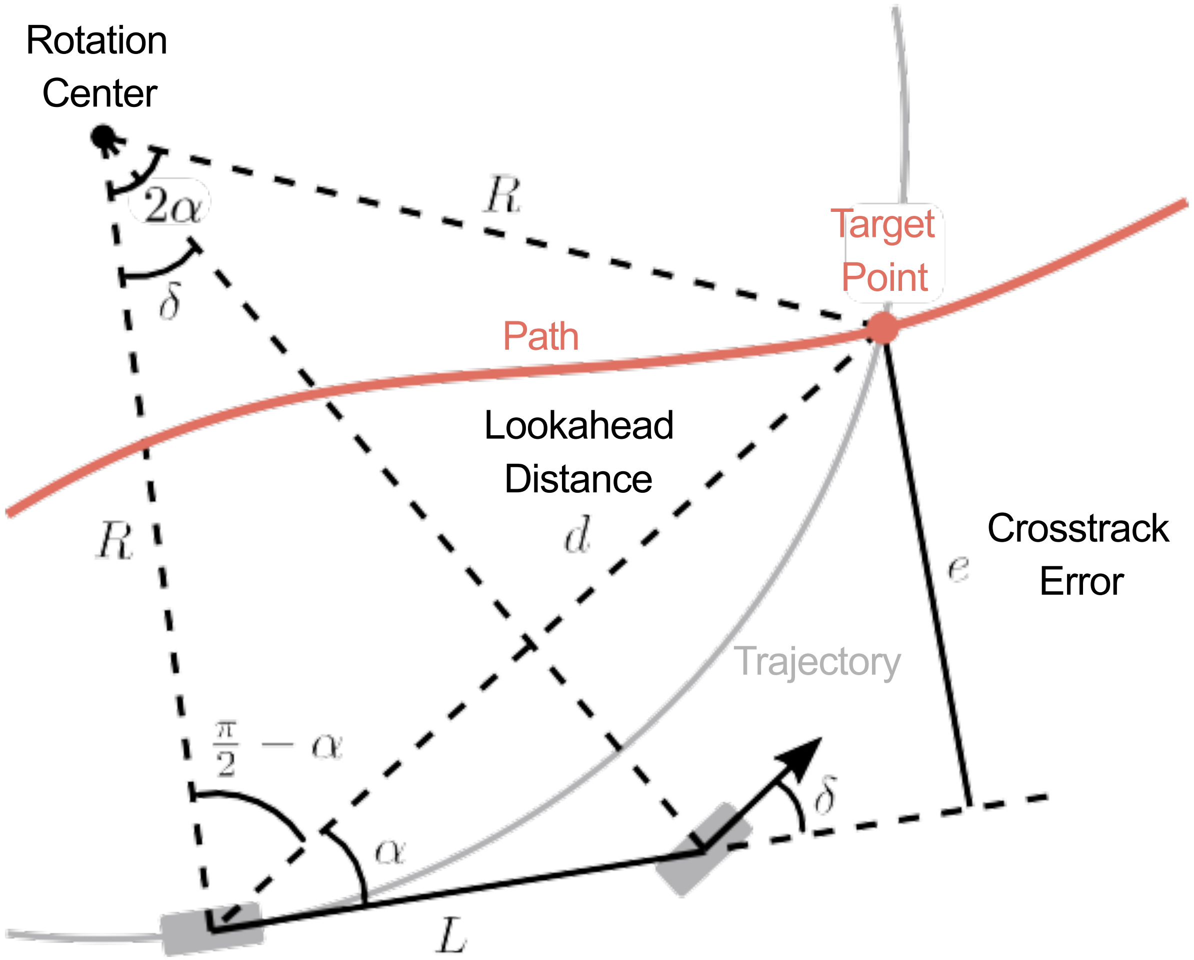
- Track **target point** at **lookahead distance** d to follow path
- Exploit geometric relationship between vehicle and path to follow
- Minimize **crosstrack error** e by following circular trajectory



Pure Pursuit Control

Goal:

- Track **target point** at **lookahead distance** d to follow path
- Exploit geometric relationship between vehicle and path to follow
- Minimize **crosstrack error** e by following circular trajectory
- Steering angle δ determined by angle α between vehicle heading direction and lookahead direction: $\delta(\alpha) = ?$



Pure Pursuit Control

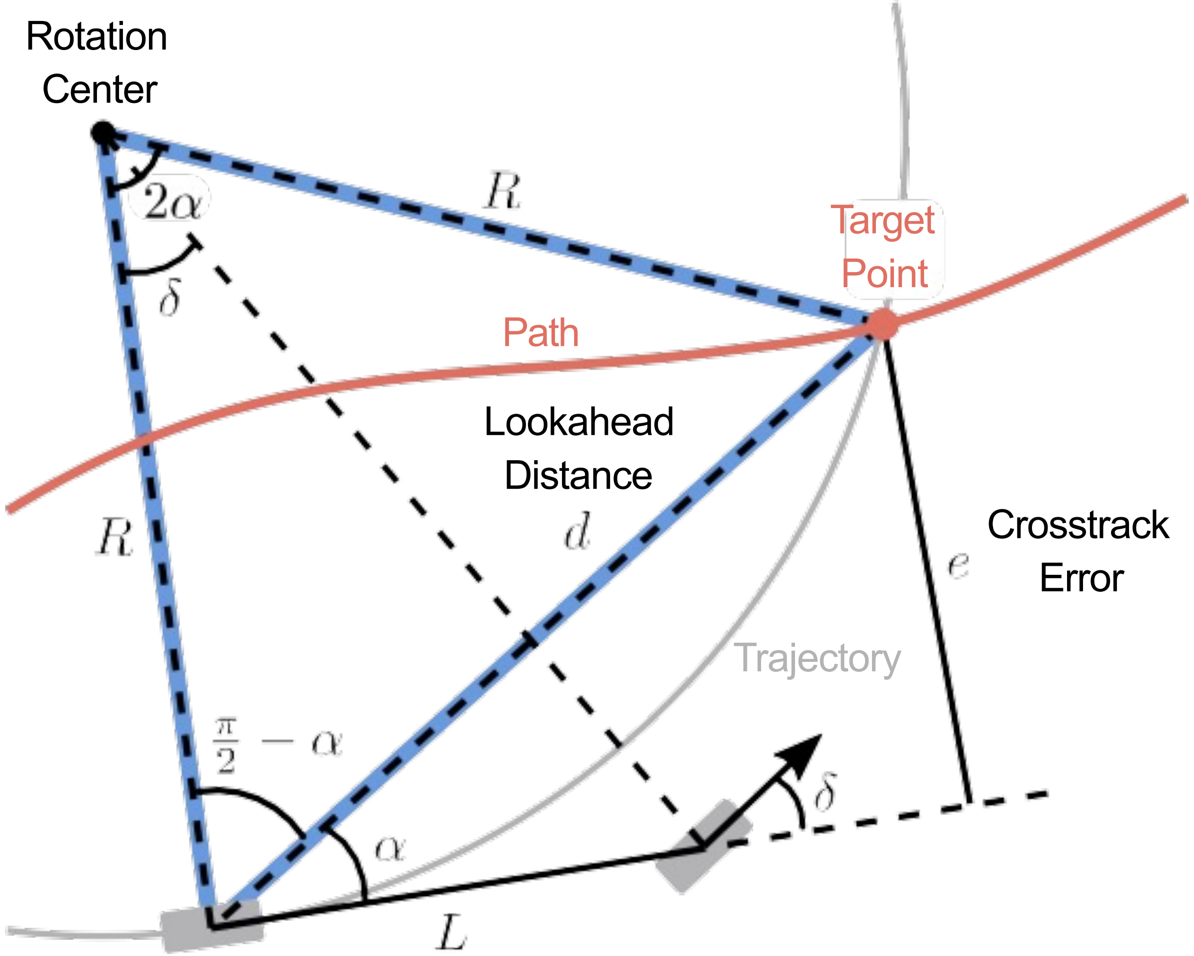
From the **law of sines:**

$$\frac{d}{\sin(2\alpha)} = \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)}$$

$$\frac{d}{2\sin\alpha\cos\alpha} = \frac{R}{\cos\alpha}$$

$$\kappa = \frac{1}{R} = \frac{2\sin\alpha}{d}$$

with κ the curvature of trajectory.



Pure Pursuit Control

The **steering angle** is calculated as:

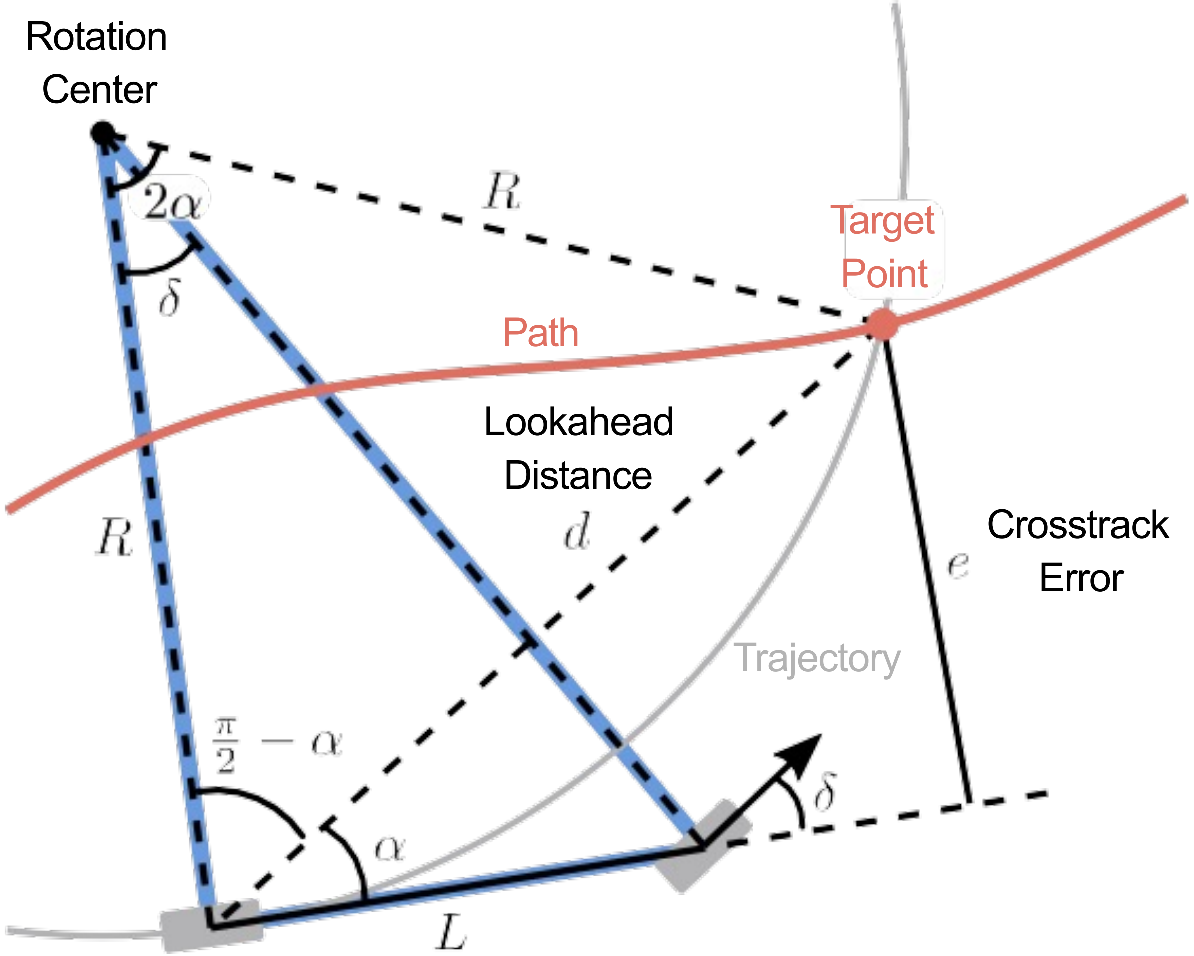
$$\tan(\delta) = \frac{L}{R} = \frac{2L\sin(\alpha)}{d}$$

$$\delta = \tan^{-1}\left(\frac{2L\sin(\alpha)}{d}\right)$$

$$\delta \approx \frac{2L\sin(\alpha)}{d}$$

- d is often based on vehicle speed v:

  $d = Kv$ with constant K



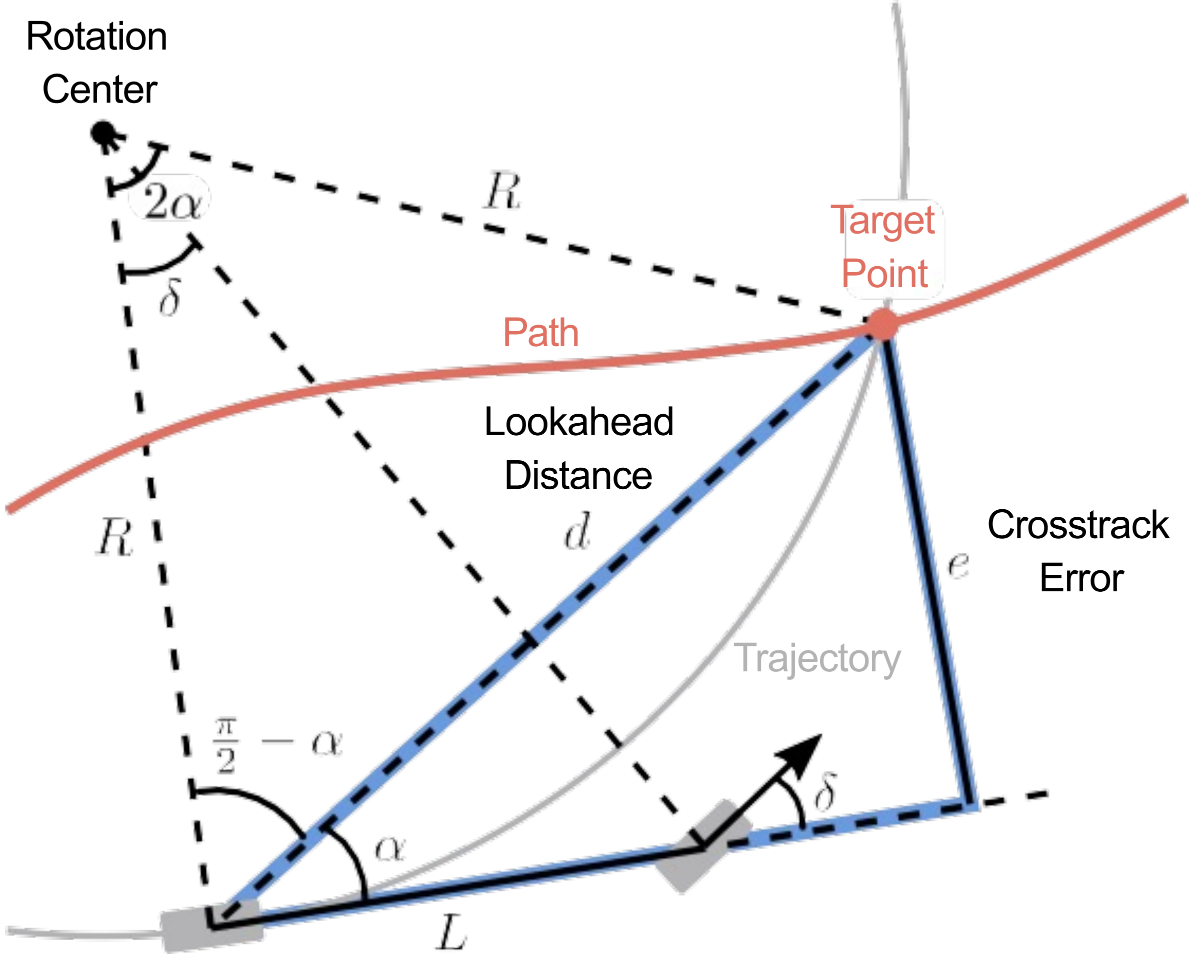
Pure Pursuit Control

In terms of **cross track error** we obtain:

$$\sin\alpha = \frac{e}{d}$$

$$\delta = \tan^{-1}\left(\frac{2L\sin(\alpha)}{d}\right)$$

$$\delta = \tan^{-1}\left(\frac{2Le}{d^2}\right) \approx \frac{2L}{d^2}e$$



- Pure pursuit acts as a **proportional** controller wrt. the crosstrack error
- d is often based on vehicle speed v:

$$d = Kv \quad \text{with constant } K$$

OPTIMAL CONTROL

Optimal Control

Recap: Dynamic Bicycle Model

$$\begin{bmatrix} \dot{v}_y \\ \dot{\psi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -\frac{c_r+c_f}{mv_x} & 0 & \frac{c_r l_r - c_f l_f}{mv_x} - v_x \\ 0 & 0 & 1 \\ \frac{l_r c_r - l_f c_f}{I_z v_x} & 0 & -\frac{l_f^2 c_f + l_r^2 c_r}{I_z v_x} \end{bmatrix} \underbrace{\begin{bmatrix} v_y \\ \psi \\ \omega \end{bmatrix}}_{\text{State } \mathbf{x}} + \begin{bmatrix} \frac{c_f}{m} \\ 0 \\ \frac{c_f}{I_z} l_f \end{bmatrix} \underbrace{\delta}_{\text{Input}}$$

With state $\mathbf{x}$ and front steering angle δ.

We can rewrite this equation as the following linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\delta \qquad \text{with} \qquad \mathbf{x} = (v_y, \psi, \omega)^T$$

Linear Quadratic Regulator (LQR): For the continuous-time linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\delta \qquad \text{with} \qquad \mathbf{x}(0) = \mathbf{x}_{\text{init}}$$

and quadratic cost functional defined as ($\mathbf{Q}$ is a diagonal weight matrix)

$$J = \frac{1}{2}\int_0^\infty \Delta\mathbf{x}^T(t)\mathbf{Q}\Delta\mathbf{x}(t) + q\delta(t)^2\,dt$$

with $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_{\text{target}}$. The feedback control $\delta(t)$ that minimizes J is given by

$$\delta(t) = -\mathbf{k}^T(t)\Delta\mathbf{x}(t)$$

with $\mathbf{k}(t) = \frac{1}{q}\mathbf{b}^T\mathbf{P}(t)$ and $\mathbf{P}(t)$ the solution to a Ricatti equation (no details here).

Model Predictive Control

Model Predictive Control

Generalizes LQR to:

- **Non-linear** cost function and dynamics (consider straight road leading into turn)
- **Flexible:** allows for receding window & incorporation of constraints
- **Expensive:** non-linear optimization required at every iteration (for global coordinates)

Formally:

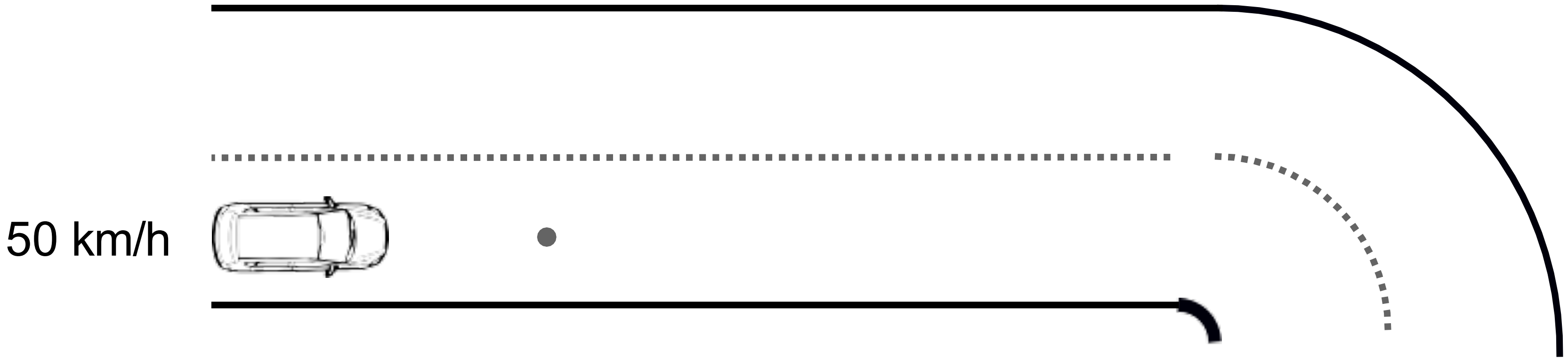
$$\underset{\delta_1,\ldots,\delta_T}{\operatorname{argmin}} \quad \sum_{t=1}^{T} C_t(\mathbf{x}_t, \delta_t) \qquad \text{(sum of costs)}$$

$$\begin{aligned} s.t. \quad & \mathbf{x}_1 = \mathbf{x}_{\text{init}} && \text{(initialization)} \\ & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \delta_t) && \text{(dynamics model)} \\ & \underline{\delta} \leq \delta_t \leq \overline{\delta} && \text{(constraints)} \end{aligned}$$

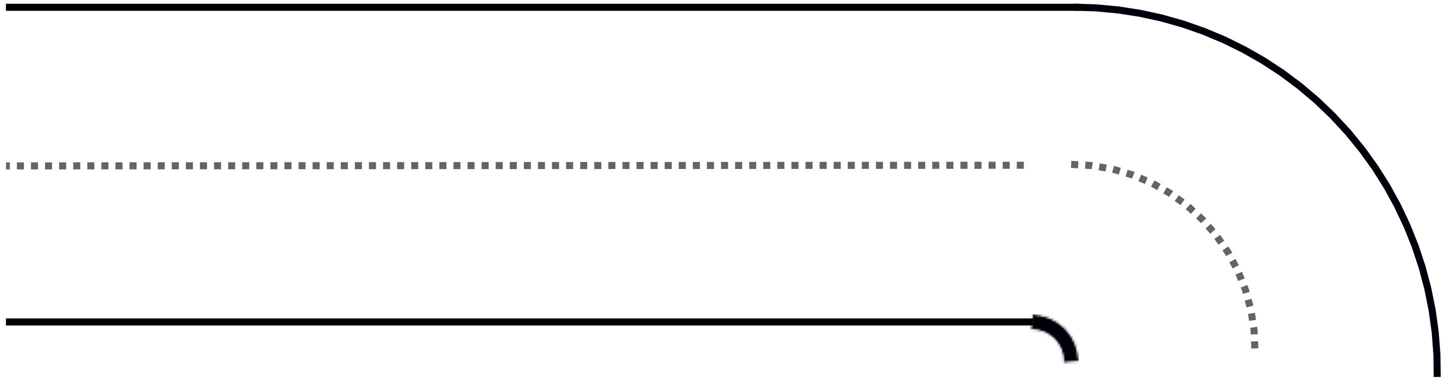
- Unroll dynamic model T times ⇒ apply non-linear optimization to find $\delta_1, \ldots \delta_T$

Model Predictive Control

PID Control / Path Tracking

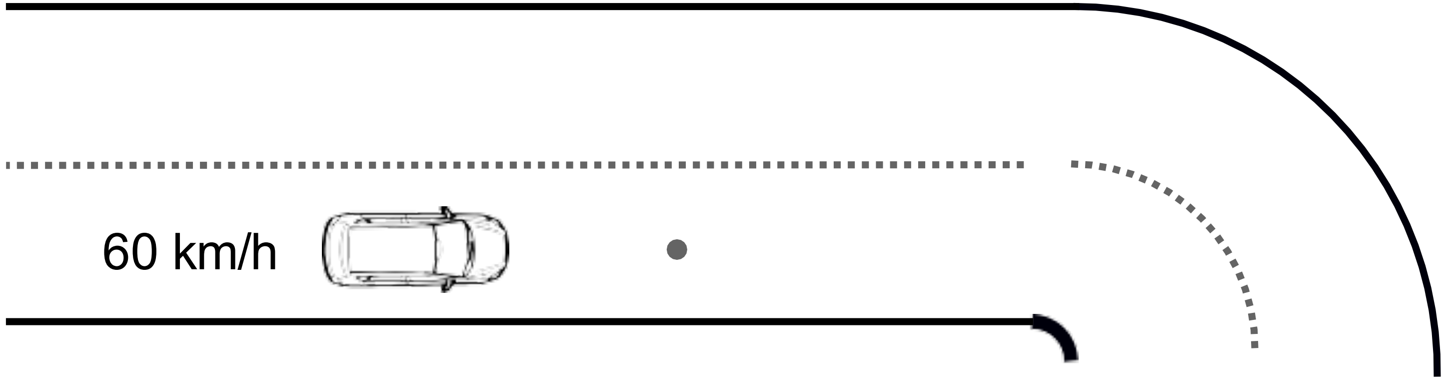


Model Predictive Control

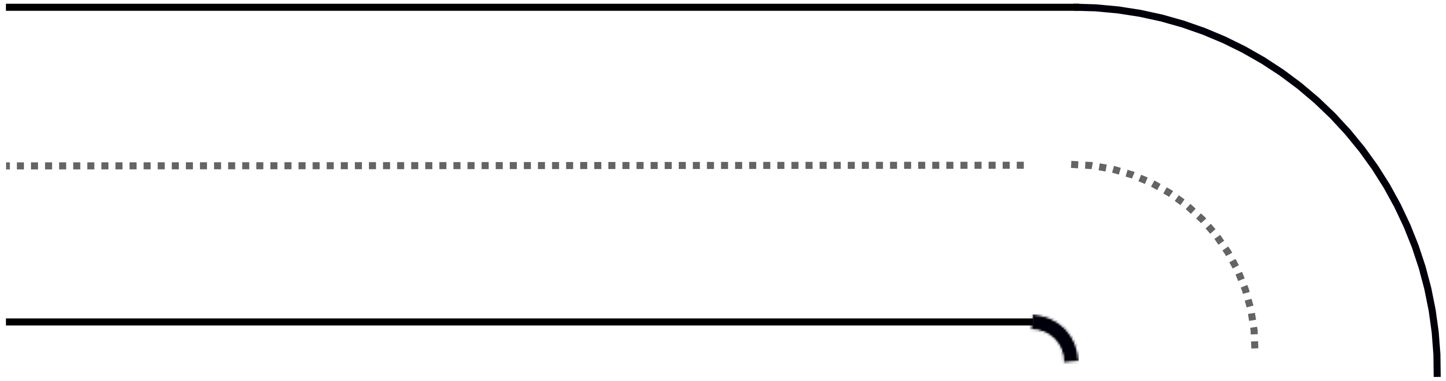


Model Predictive Control

PID Control / Path Tracking

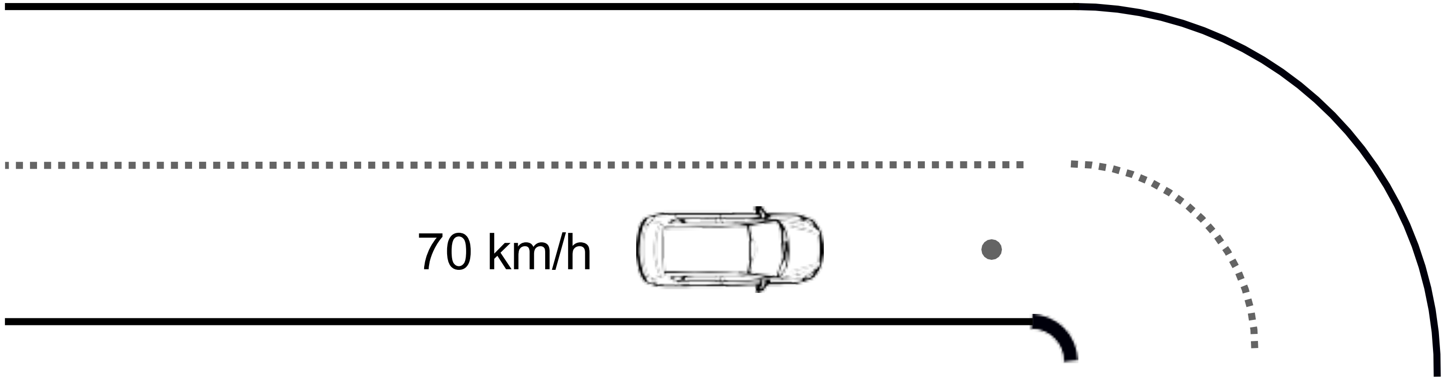


Model Predictive Control

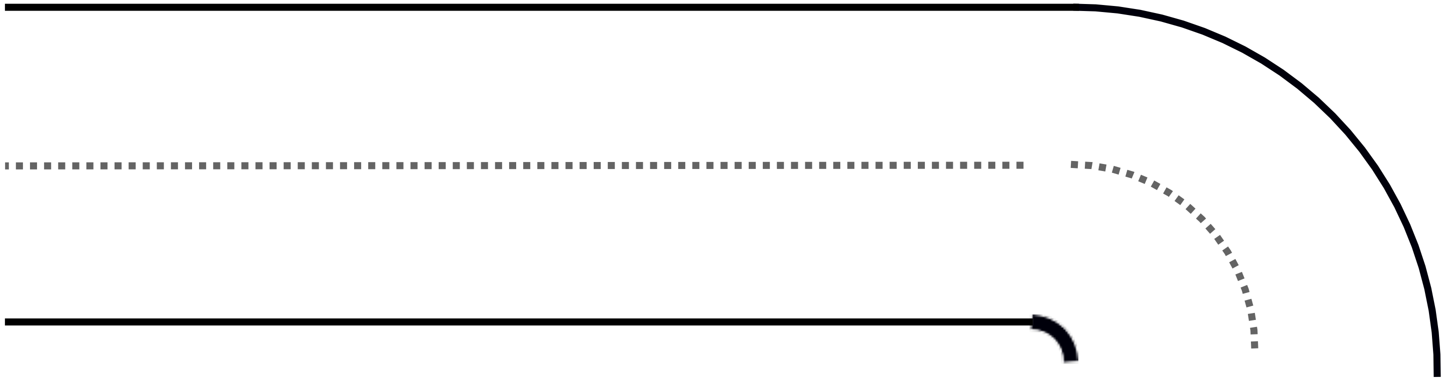


Model Predictive Control

PID Control / Path Tracking

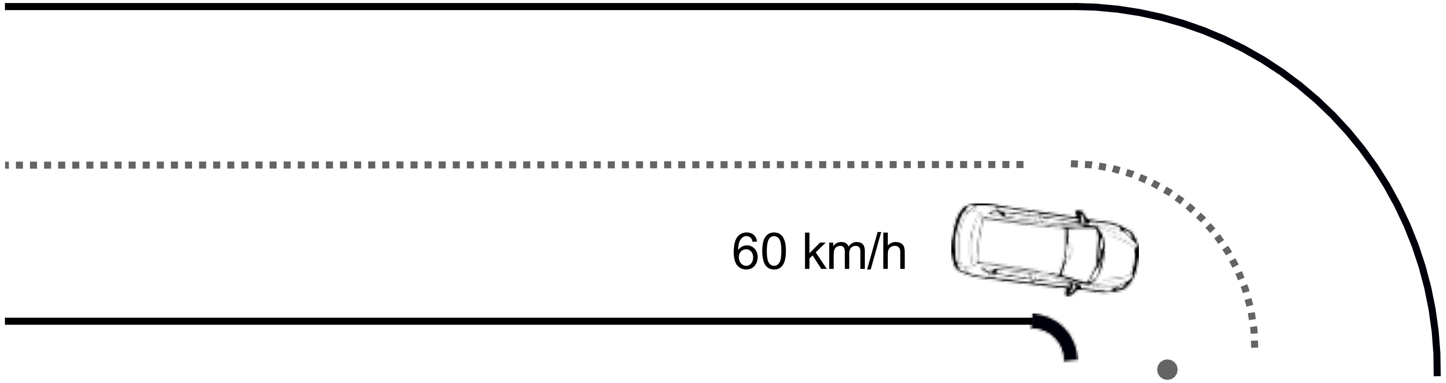


Model Predictive Control

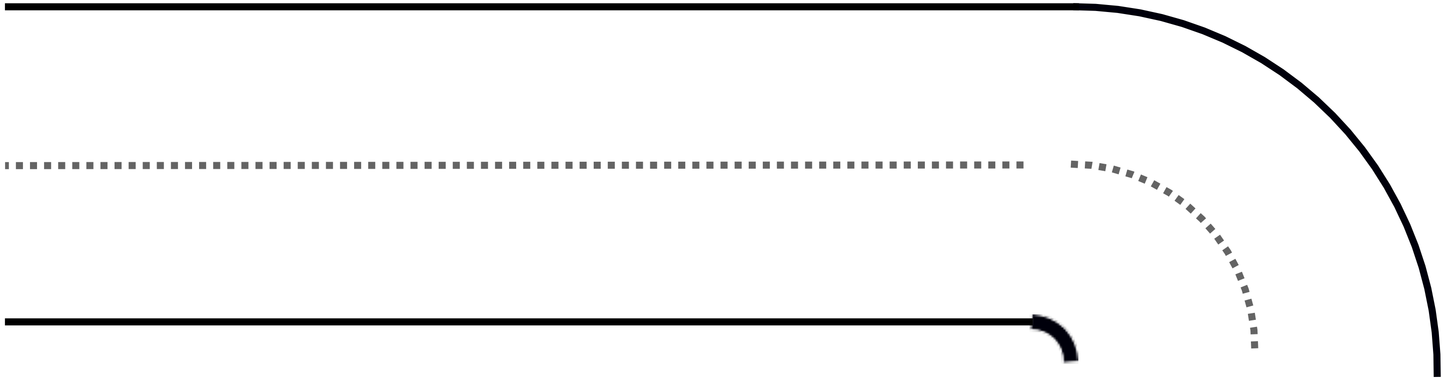


Model Predictive Control

PID Control / Path Tracking

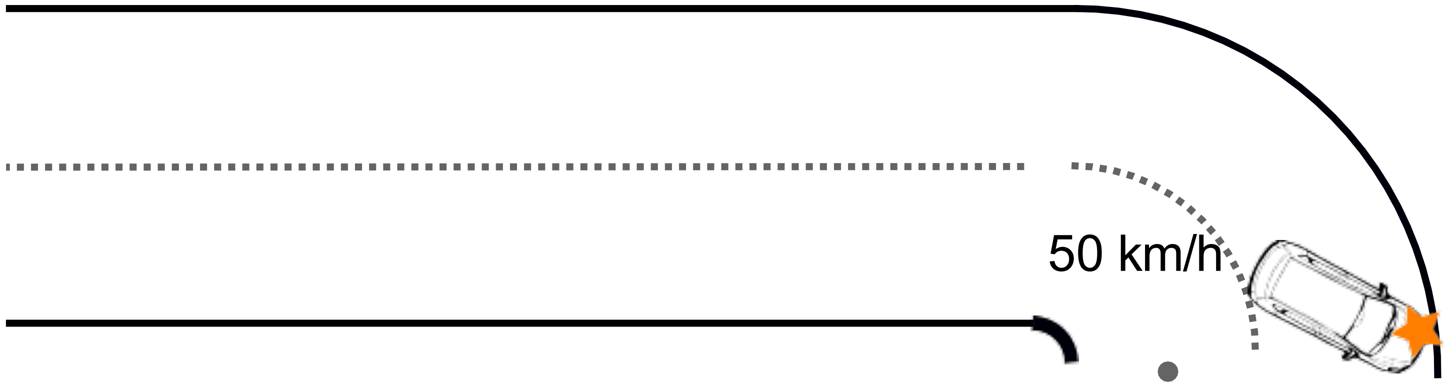


Model Predictive Control

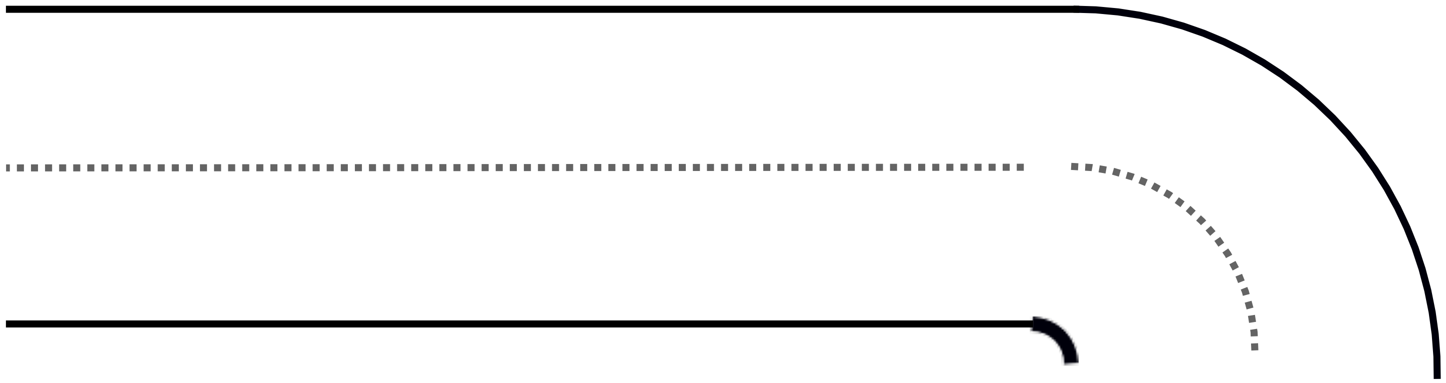


Model Predictive Control

PID Control / Path Tracking

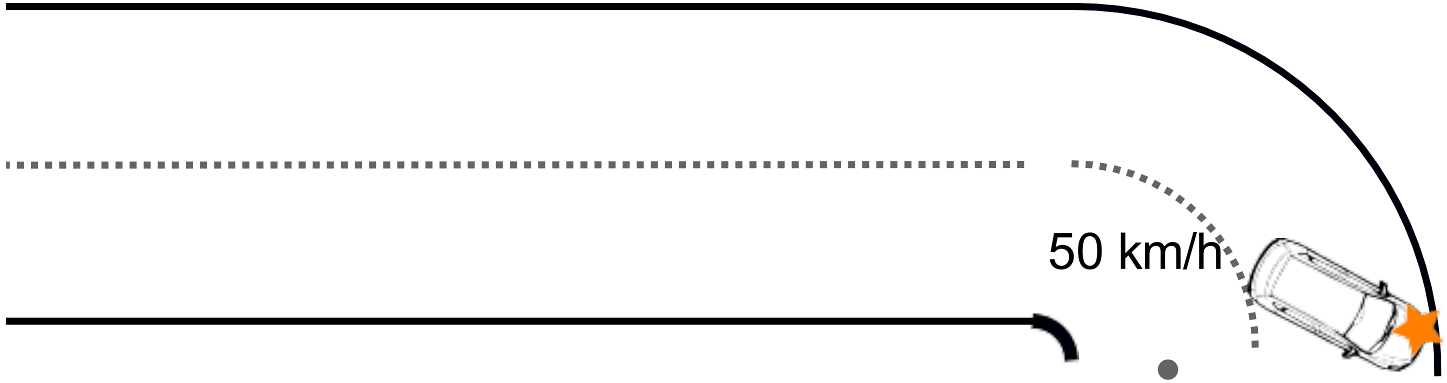


Model Predictive Control

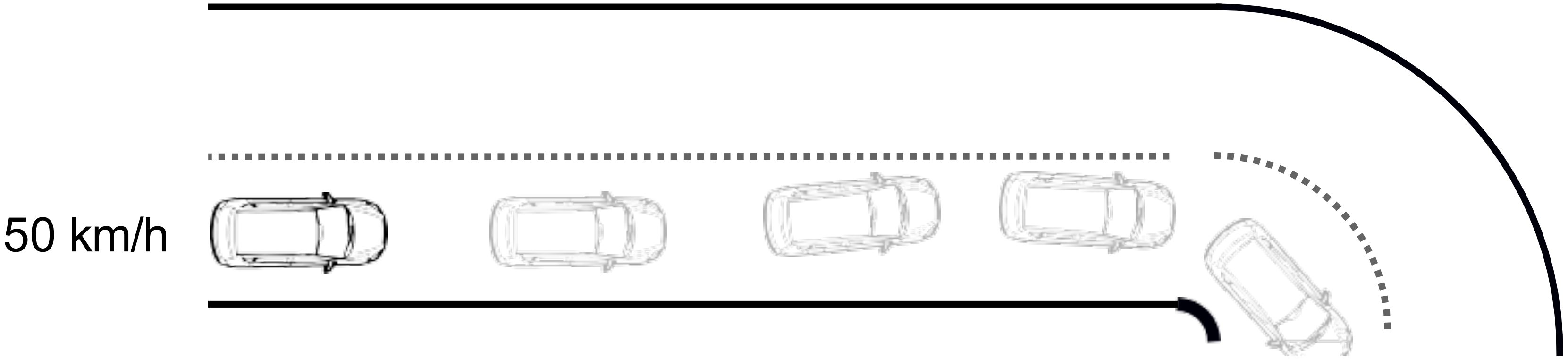


Model Predictive Control

PID Control / Path Tracking

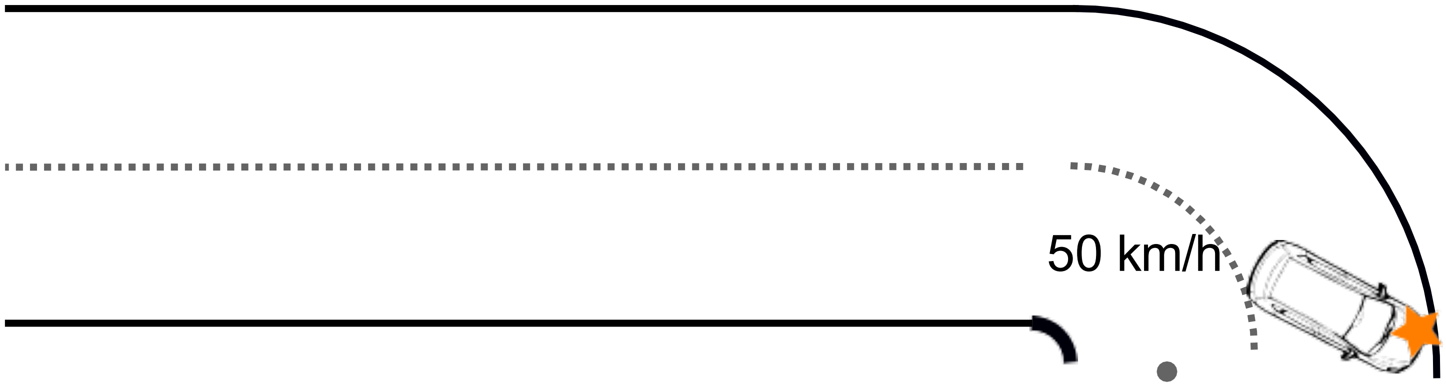


Model Predictive Control

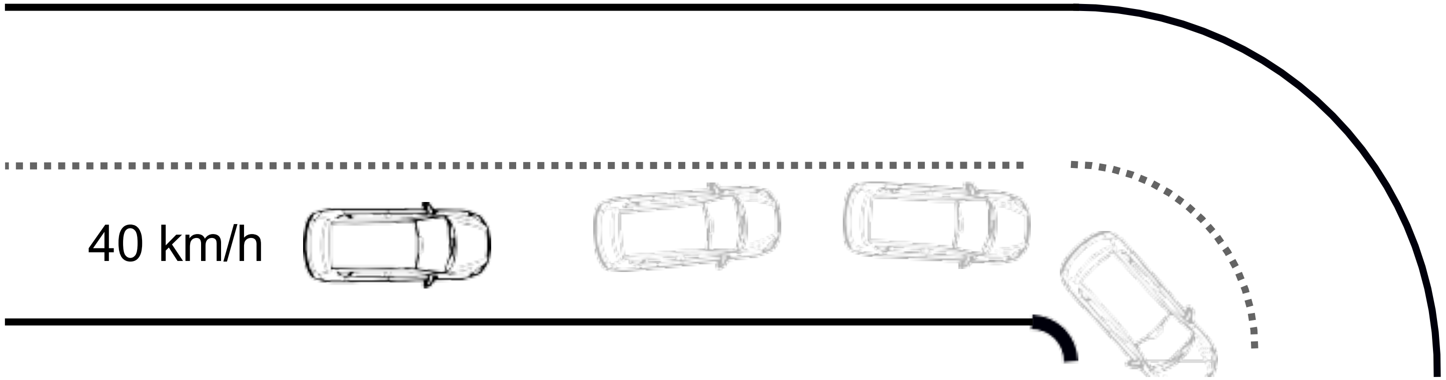


Model Predictive Control

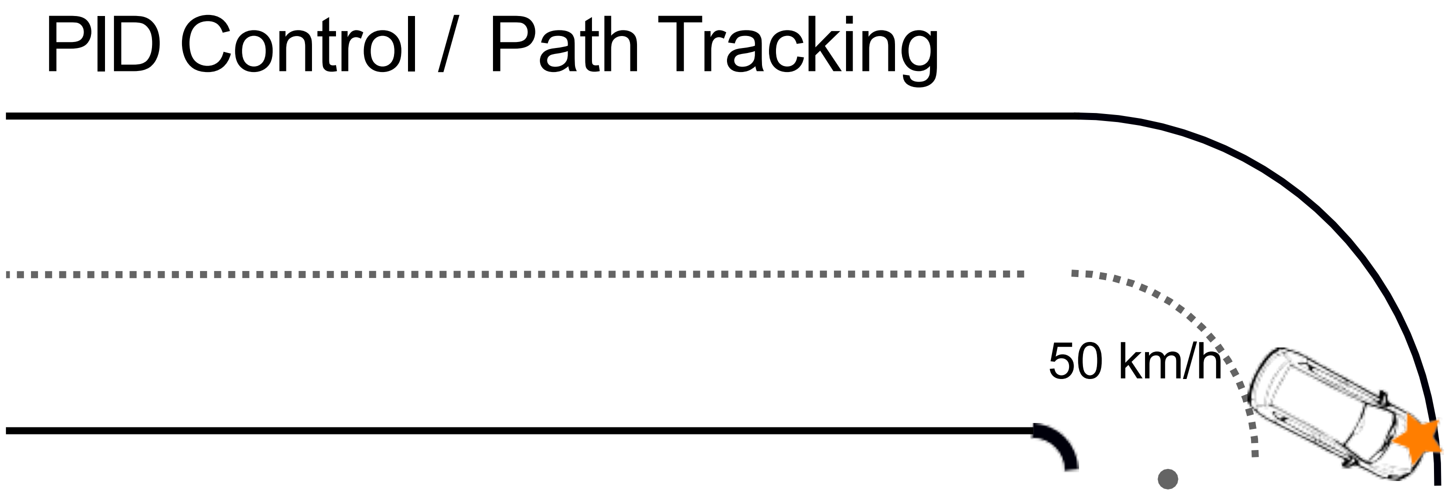
PID Control / Path Tracking

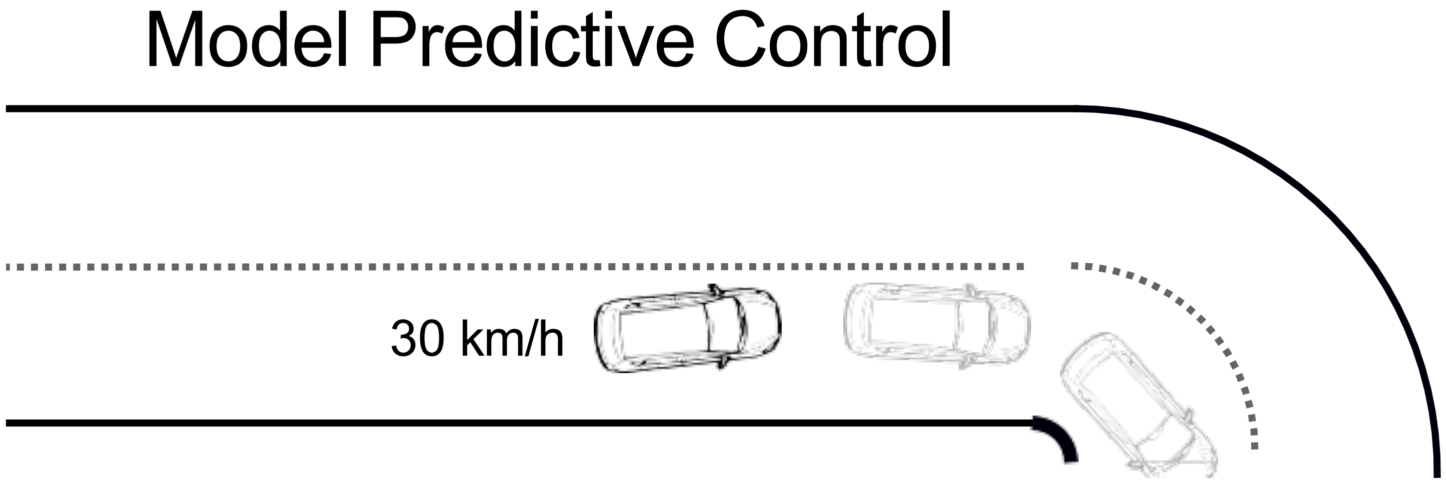


Model Predictive Control



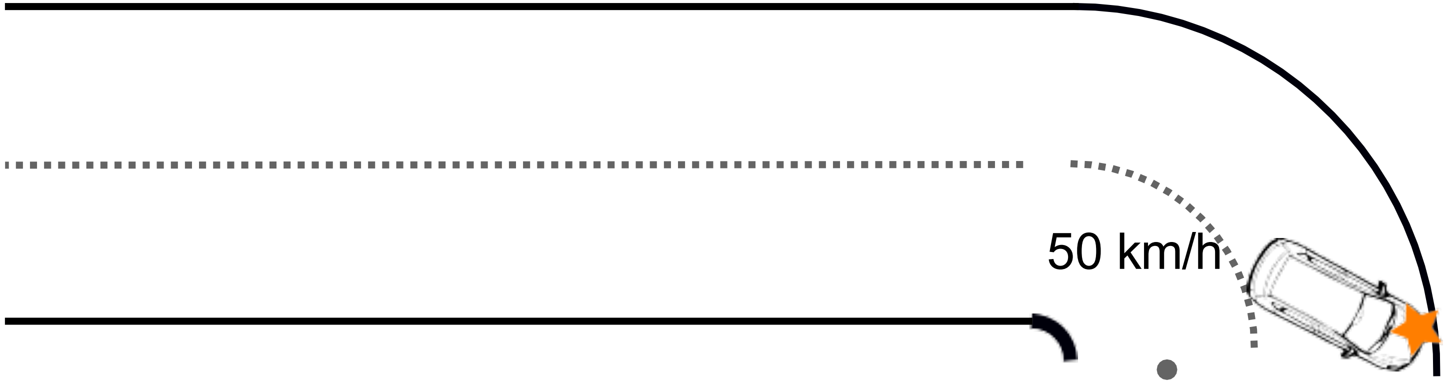
Model Predictive Control



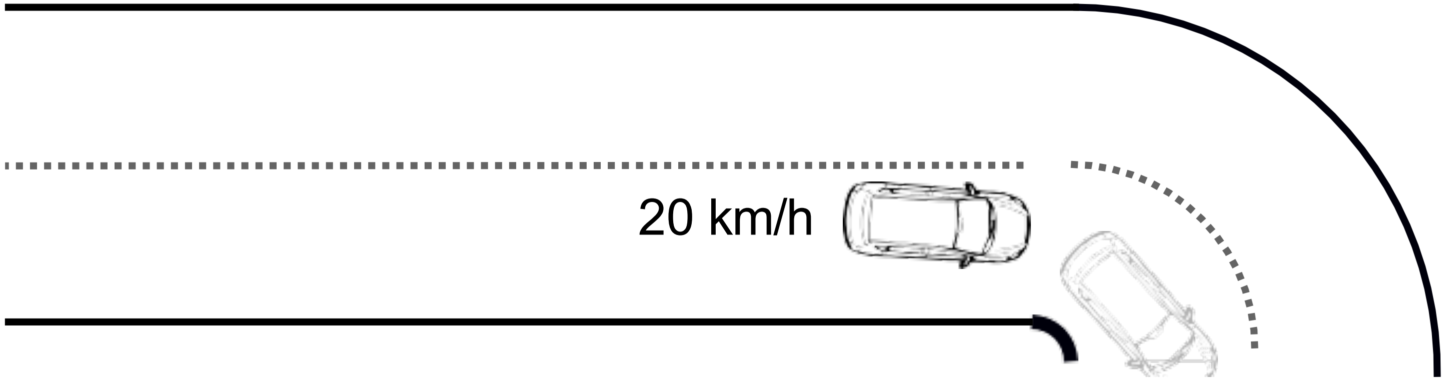


Model Predictive Control

PID Control / Path Tracking

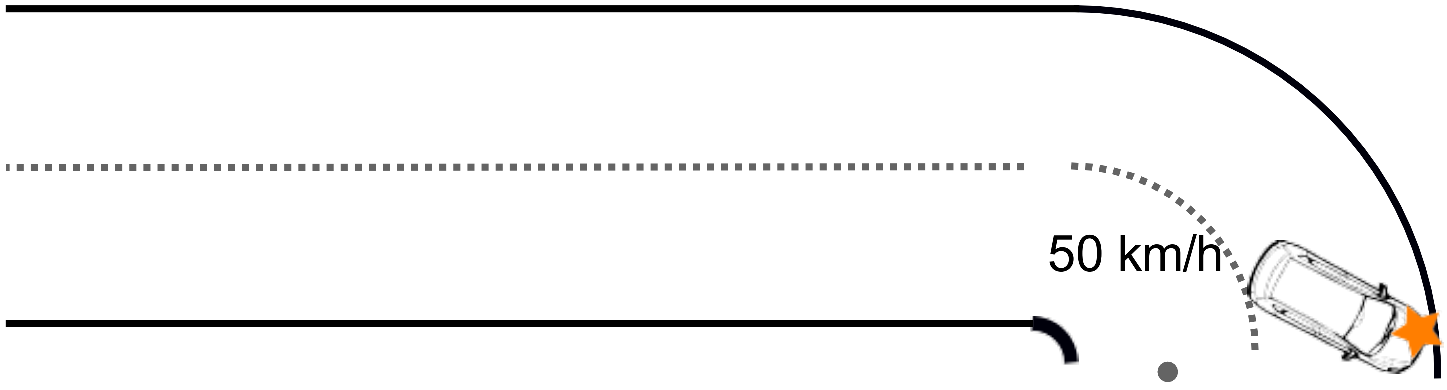


Model Predictive Control

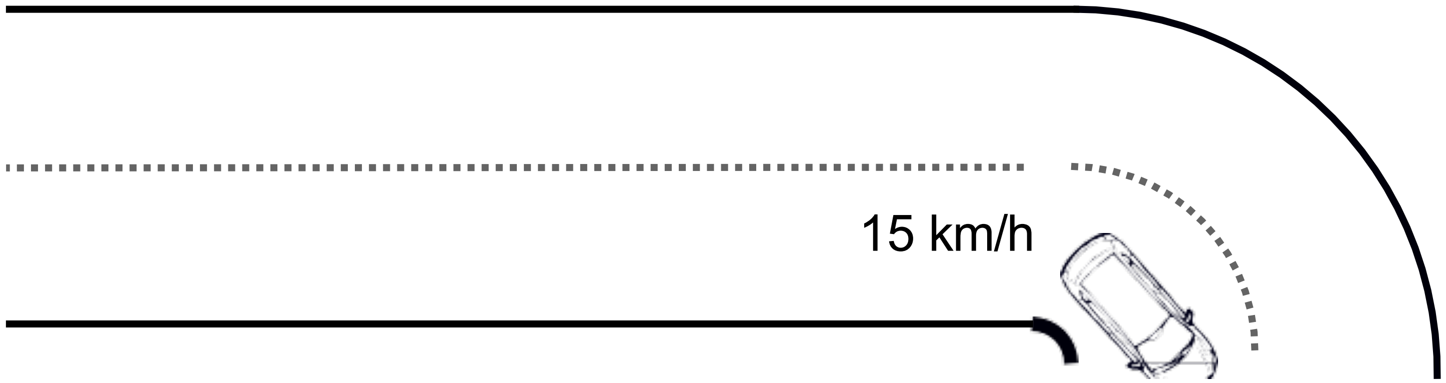


Model Predictive Control

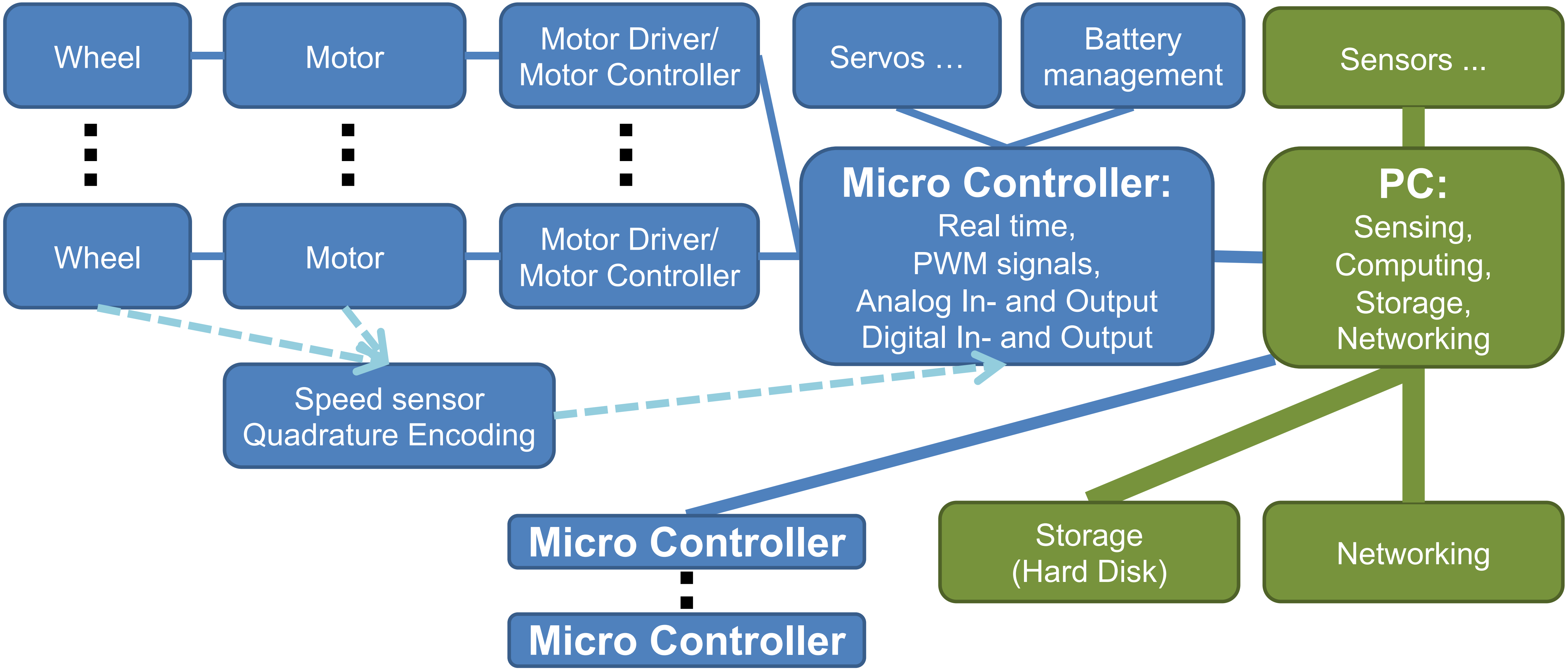
PID Control / Path Tracking



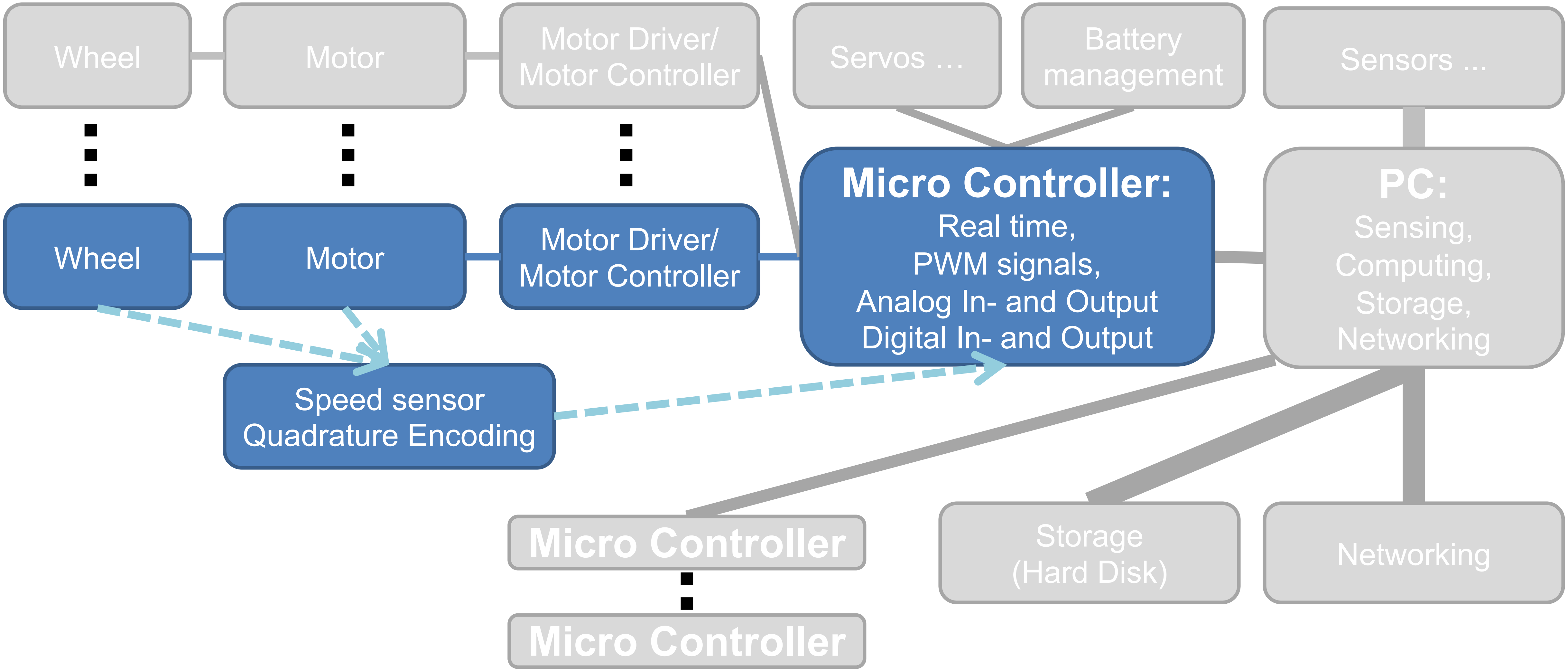
Model Predictive Control



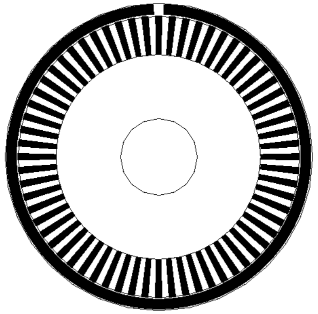
Overview Hardware

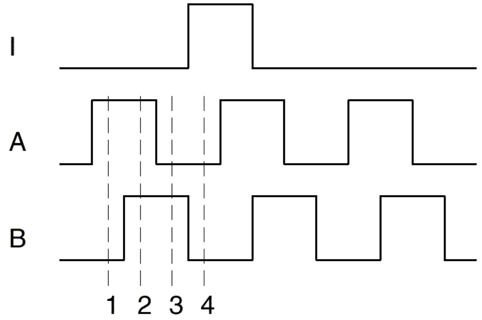


The Mechatronics of Wheeled Locomotion



DC Motor with Gearbox and Quadrature Encoder

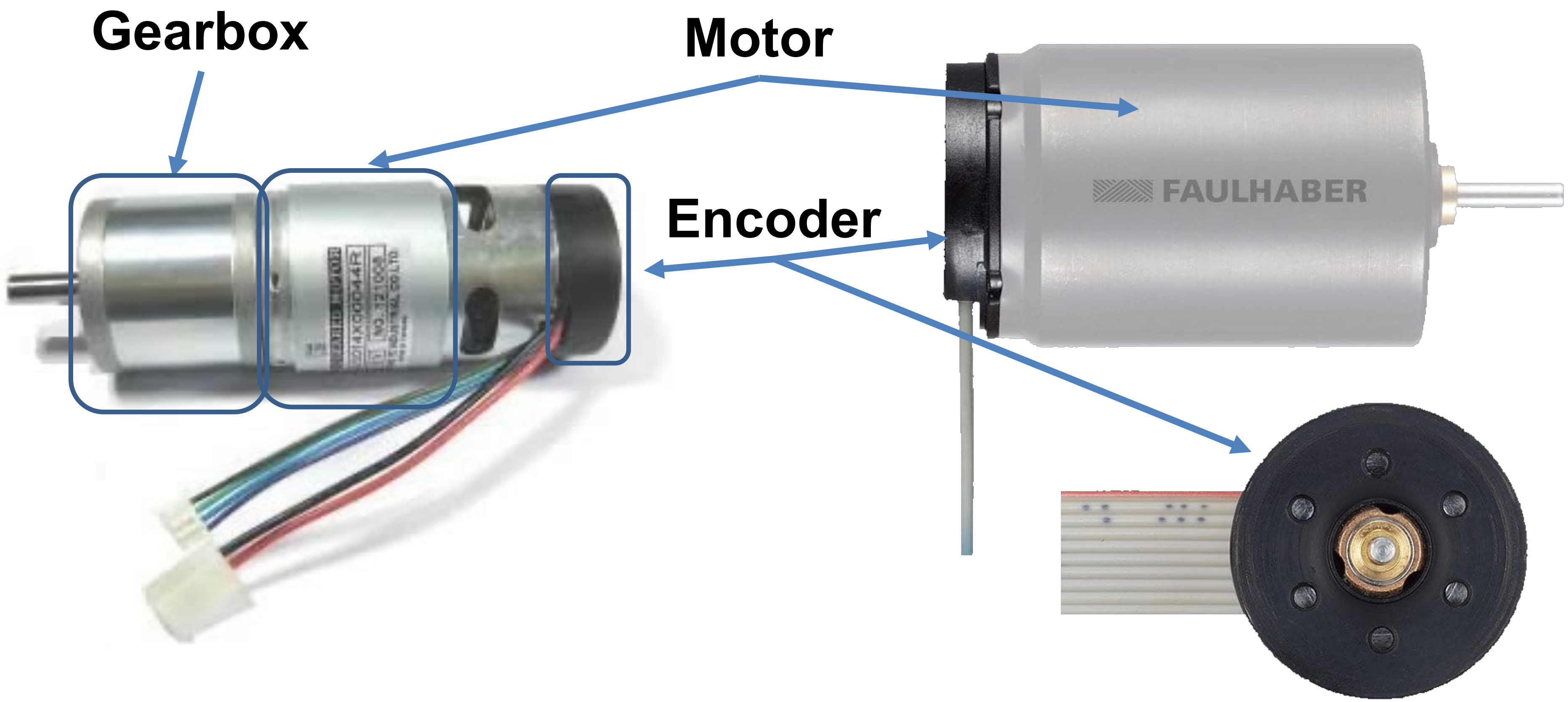




State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

Encoder Wires
Enc GND
Enc Vcc (5V)
Ch A
Ch B
Index (Optional)
Optional: Negative Signals for saver transmission

DC Motor Wires
Motor A
Motor B



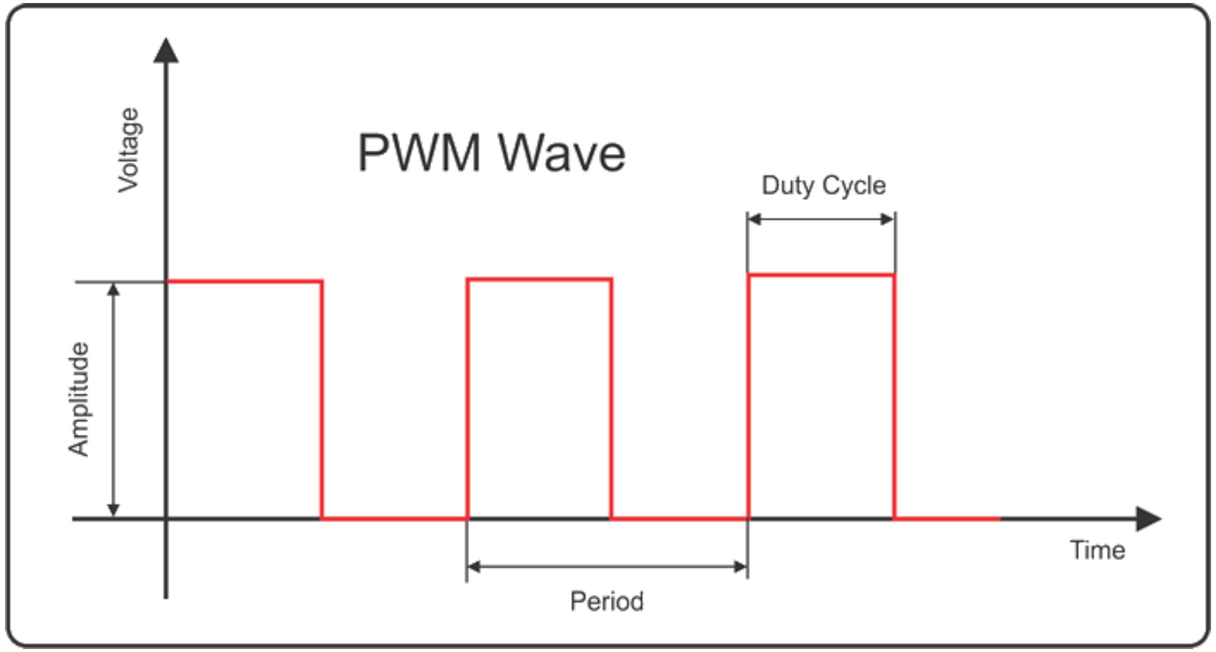
Provide Brief Overview of the following topics:

- Motor Driver
- DC brushed and brushless Motors & Servos
- Gears

MOTOR DRIVER

Image: zembedded.com

Recap: PWM

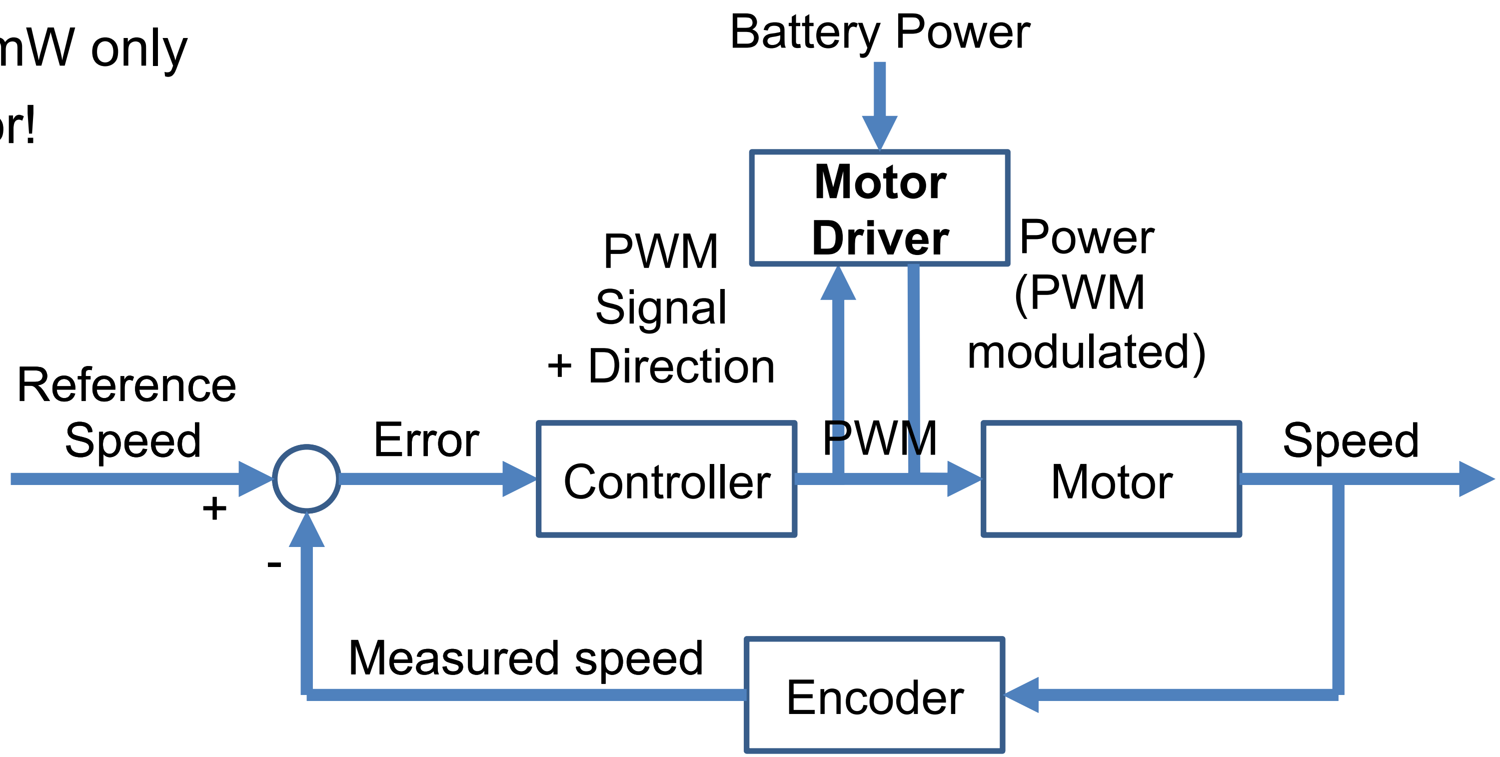


- How can Controller control power?
  - Cannot just tell the motor “use more power”
  - Output of (PID) controller is a signal
  - Typical: Analogue signal
- Pulse Width Modulation (PWM)
  - Signal is either ON or OFF
  - Ratio of time ON vs. time OFF in a given interval: amount of power
  - Frequency in kHz (= period less than 1ms)
  - Very low power loss
- Signal (typica 5V or 3.3V) to Motor Driver
- Used in all kinds of applications:
  - electric stove; audio amplifiers, computer power supply (hundreds of kHz!)

Power to the Motor

- Direct Current Motor (DC Motor):
  - Two wires for power input
  - Directly connect DC motor to PWM signal?
  - Limited current!
  - E.g.: Arduino: max 30mA => 150mW only
  - Clearpath Jackal: 250W per motor!

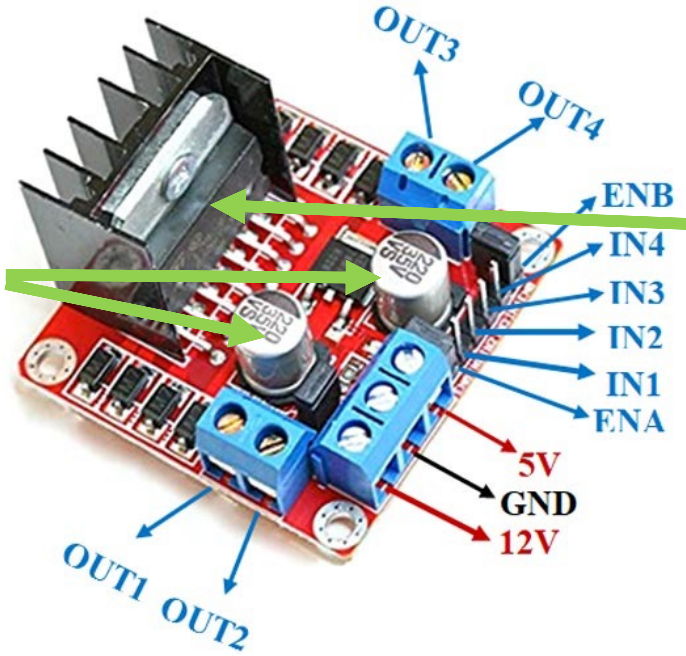
- Need a device to power the motor
- Mobile robots: battery power!

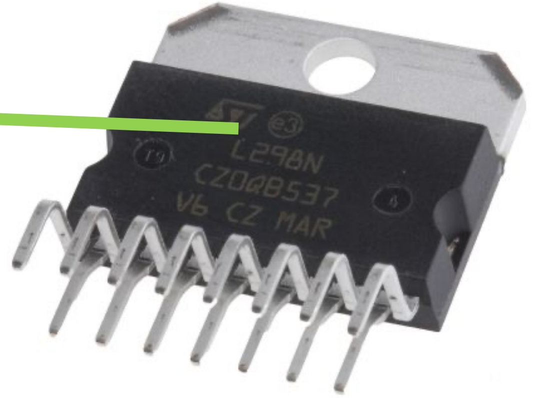


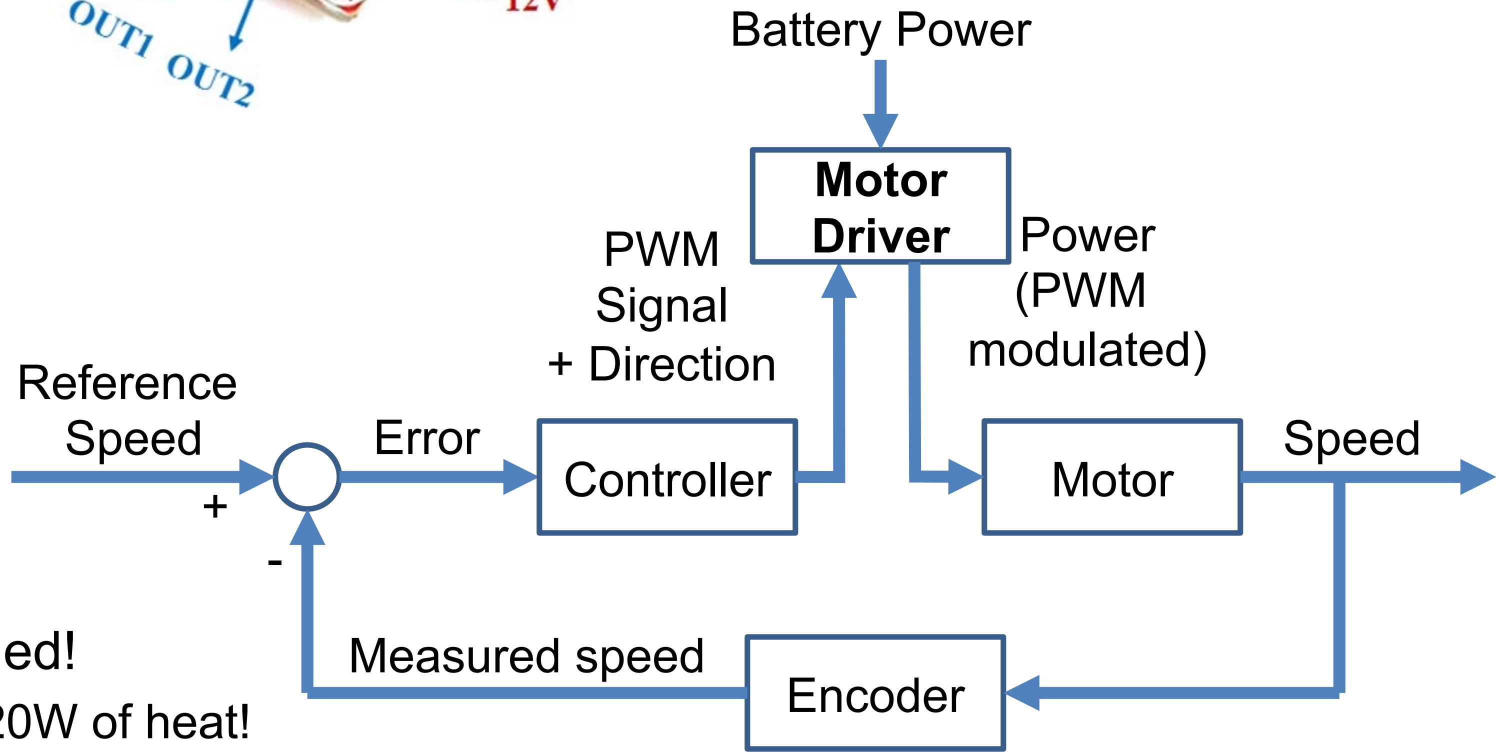
Motor Driver

- Motor Driver
  - Input:
    - PWM signal
    - Direction of rotation
    - Battery + & -
    - Optional: Enable =>
      - Emergency Stop
  - Output:
    - Two lines to the DC motor
  - Popular: L298N dual motor driver
    - Up to 48V & 4A
  - High Efficiency (maybe 95%) – but still get's hot – cooling needed!
    - E.g.: 2x200W motor still produce 20W of heat!

Capacitors to smooth output power

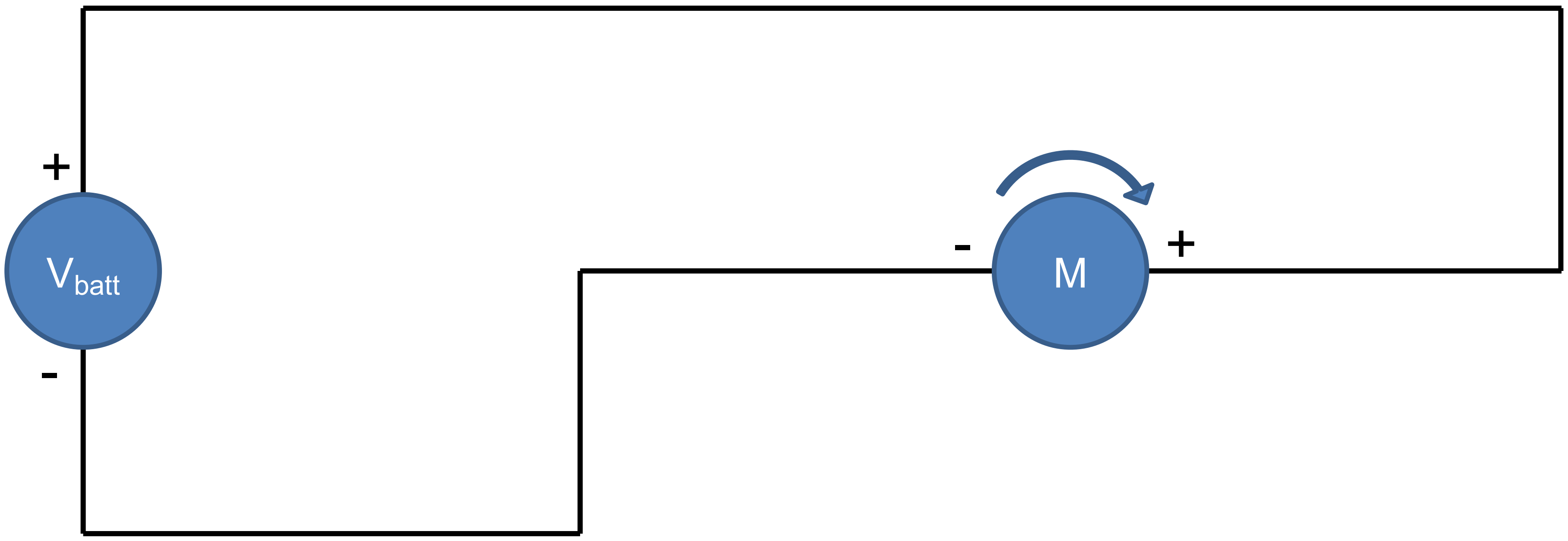






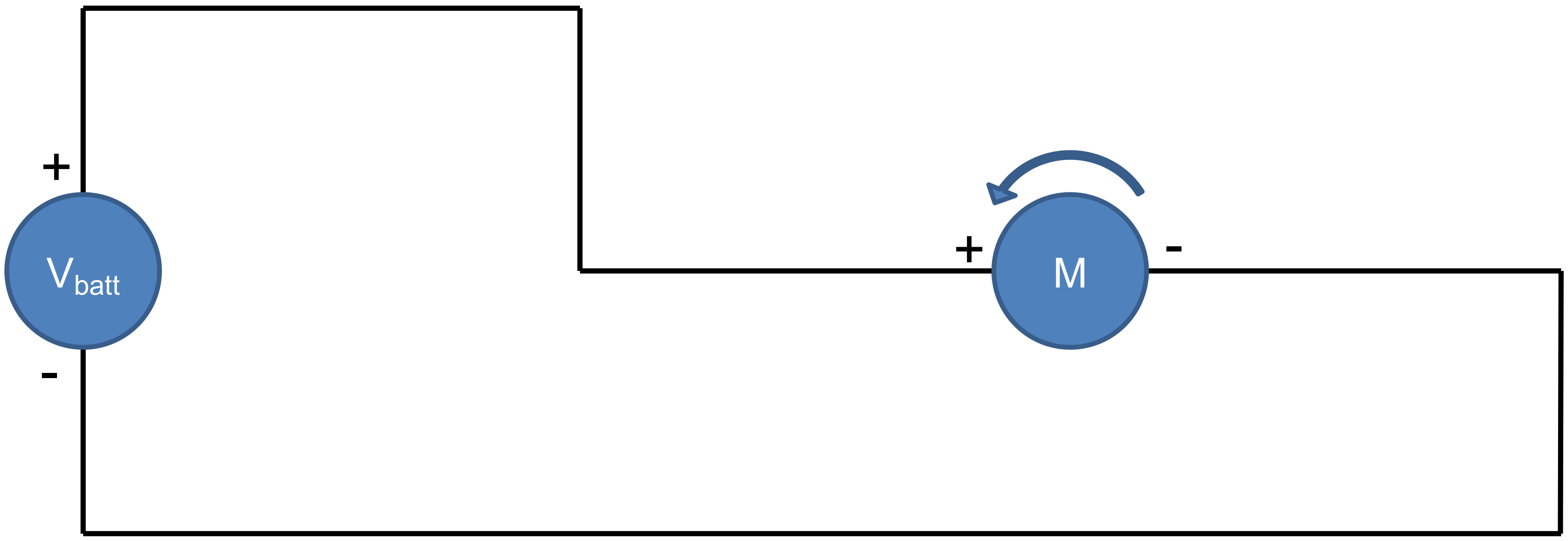
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



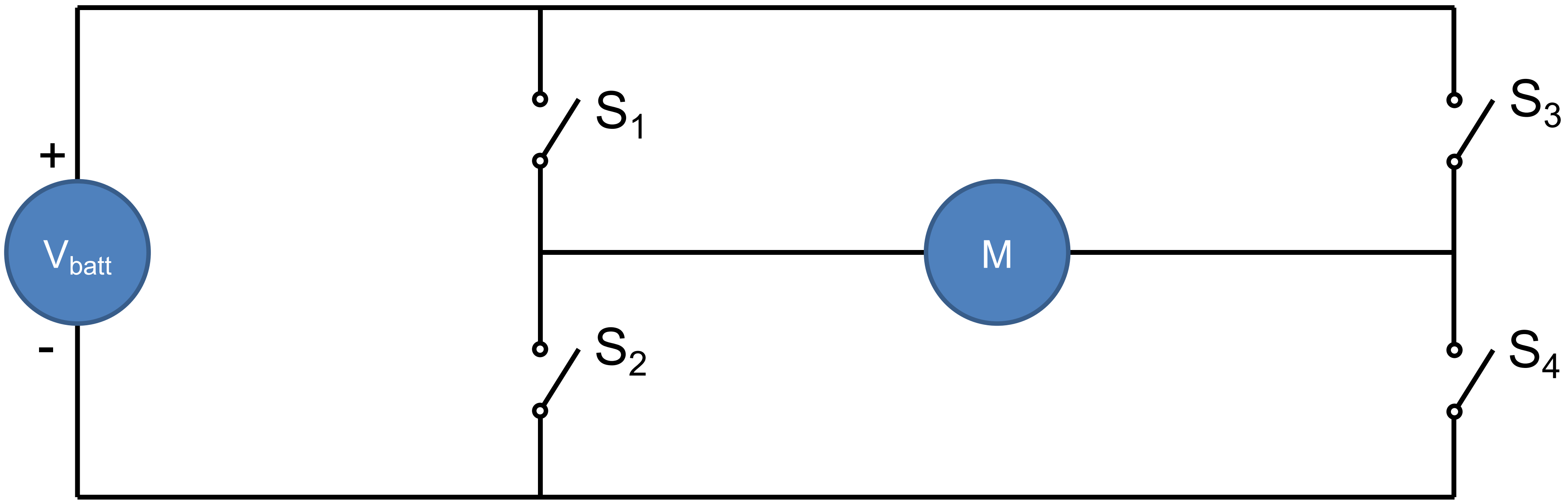
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



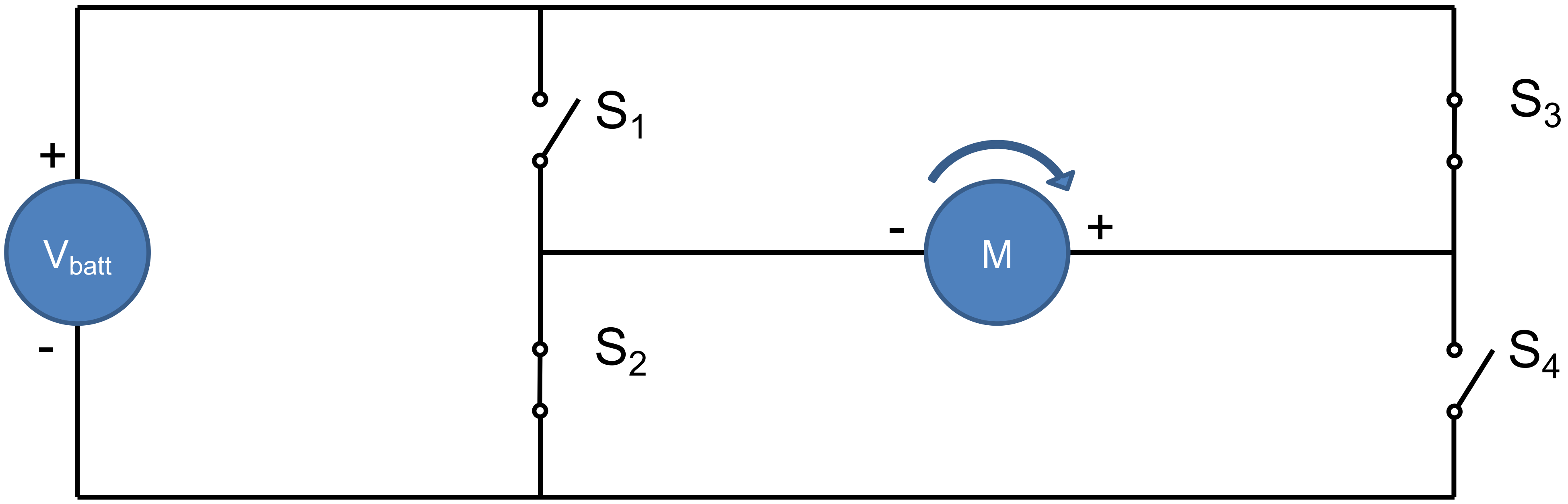
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



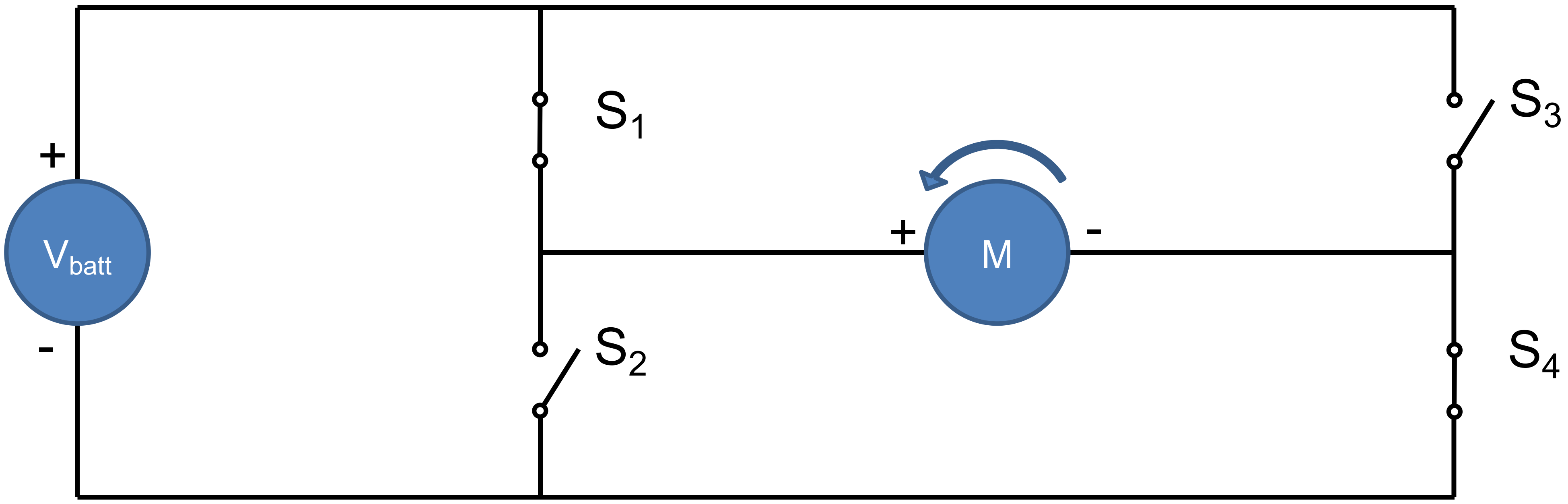
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



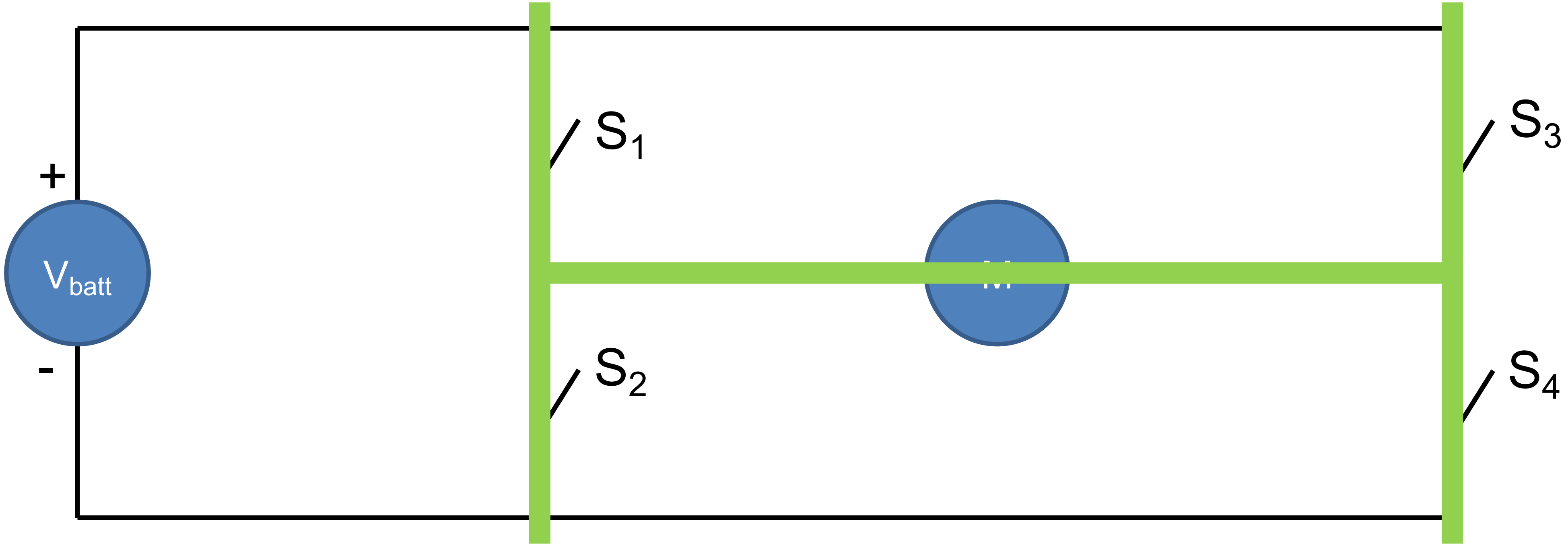
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



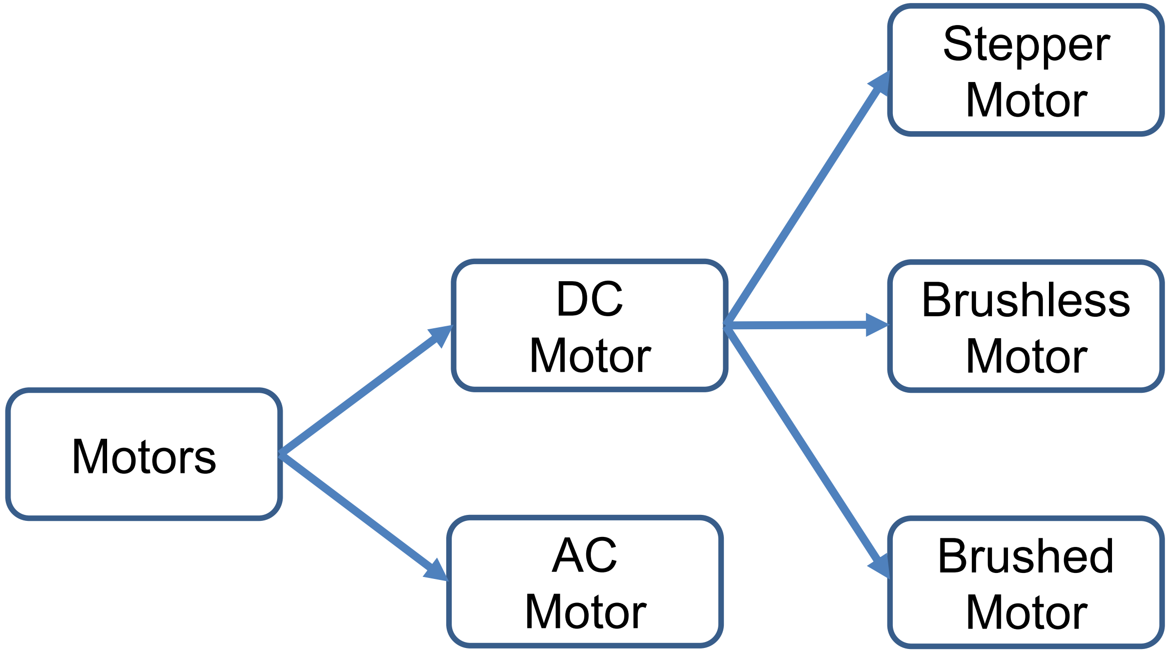
How does a Motor Driver work? H-Bridge

- H-Bridge:
  - Change direction of energy flow -> change direction of motor
  - Also: switch motor on and off



MOTORS

Electrical Motor Types



- DC Motor: Direct Current Motor
- AC Motor: Alternating Current Motor
- Stepper motor:
  - Switching power steps one tooth/ coils forward
  - Open loop control: no encoder needed
  - Low resolution; open loop; torque must be well known
- Brushed motor:
  - Use brushes to power rotating coils => low efficiency and high wear
- Brushless (BL) motor:
  - Electronically control which coil to power => high efficiency low wear
  - Need dedicated controller

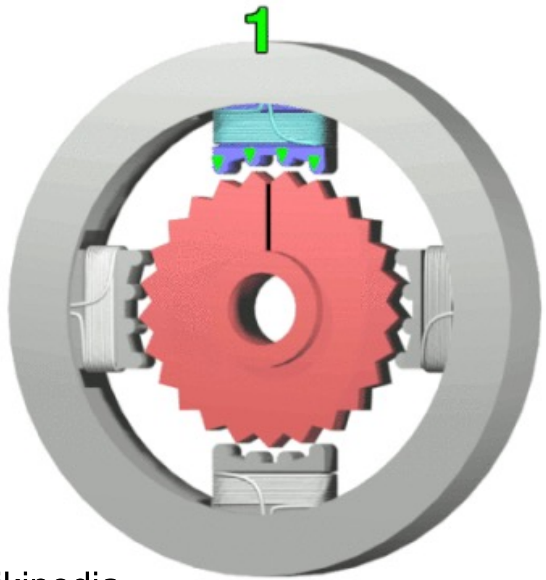
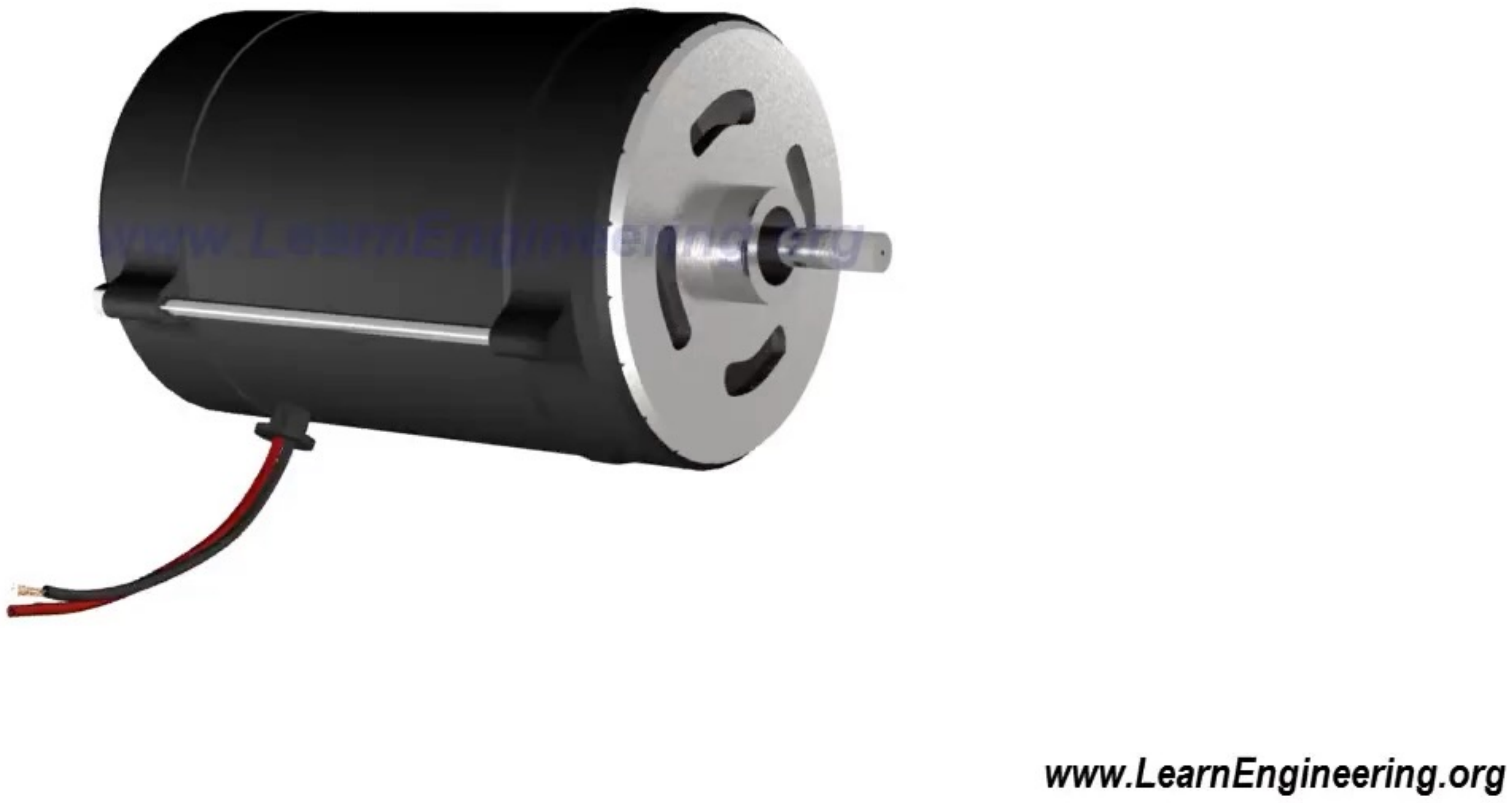


Image: Wikipedia



https://www.youtube.com/watch?v=CWulQ1ZSE3c

https://www.youtube.com/watch?v=bCEiOnuODac

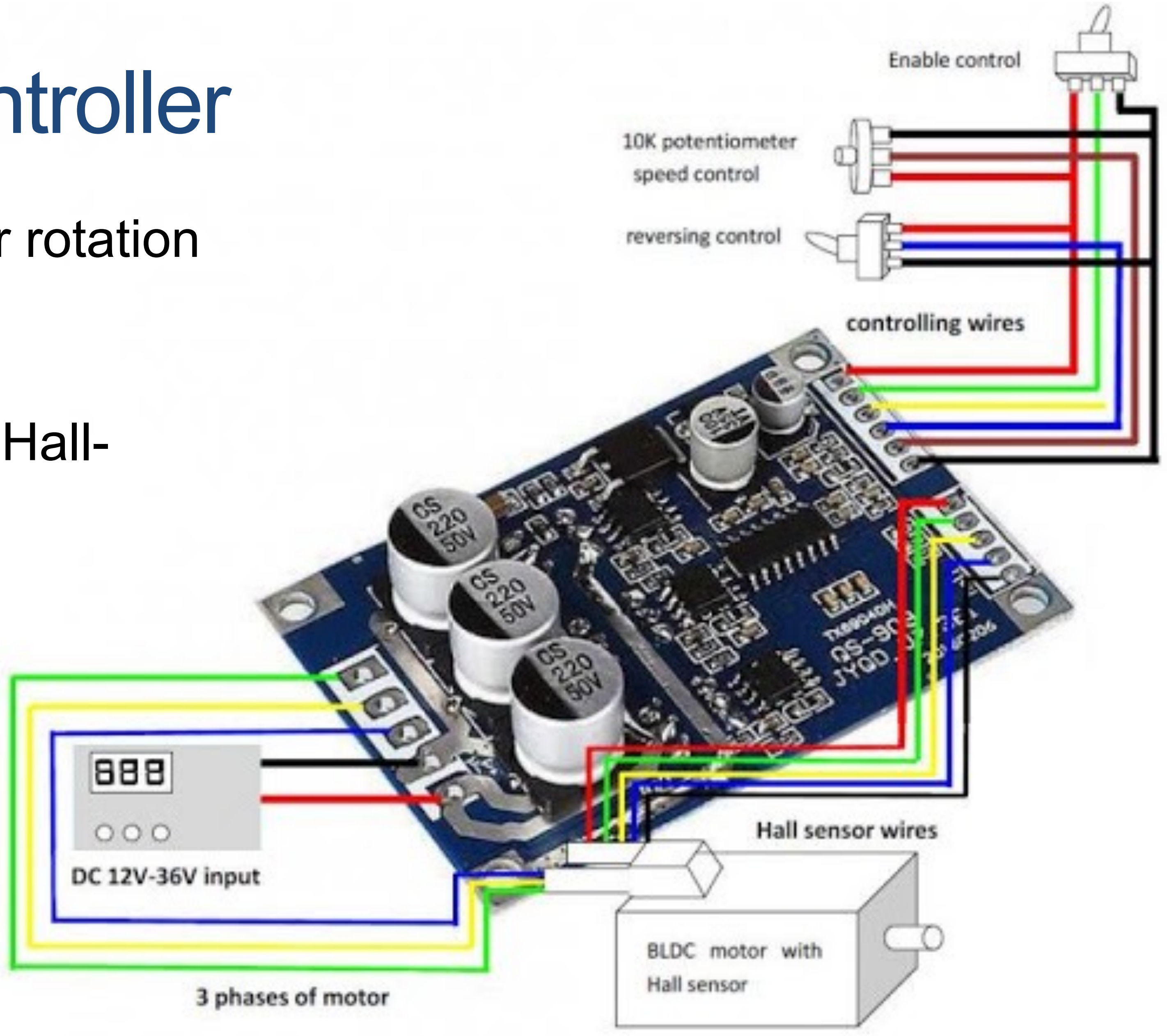
Brushless Motor Controller

- Needs BLDC Controller
  - Does also the job of Motor Driver

- Sensorless BLDC motor:
  - Just apply power to coils in correct order
  - Motor might briefly turn backwards in the beginning
  - Works well for fast spinning motors (e.g. quadcopter)
  - May use the back-EMF (electromotive force) to estimate position

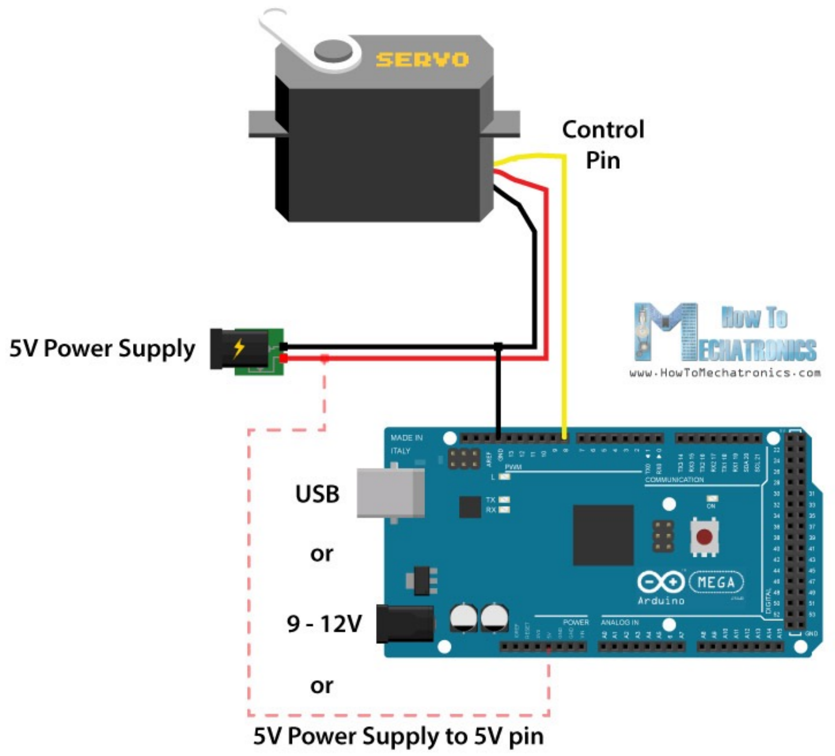
Brushless Motor Controller

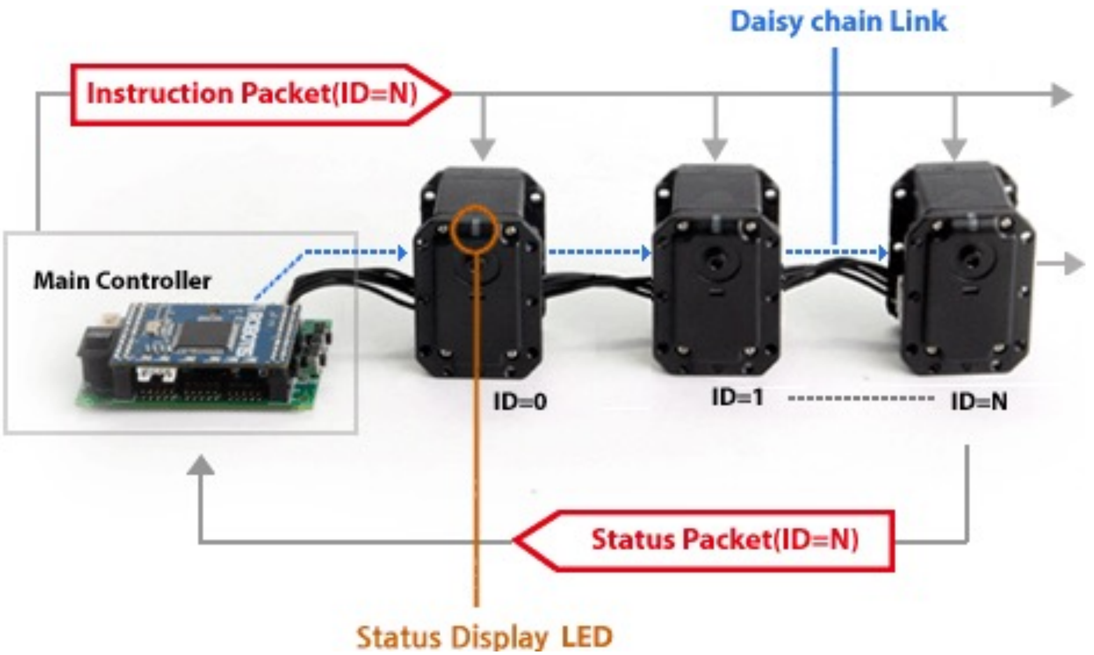
- Hall sensor only 3 positions per rotation
  - Quadrature encoder: up to 4096
- For high torque; low speeds: 3 Hall-effect sensors needed!
- External PID speed control may still be needed!
- Brushless: 20%-30% better efficiency



Servo Motor

- Combines Controller & Motor Driver in the motor
- Input may be analogue (e.g. PWM signal) or digital (e.g. Dynamixel)
- Input specifies a certain (angular) pose for the servo!
  - Servo moves and stays there.
- Continuous Rotation Servos: open loop, speed controlled motors

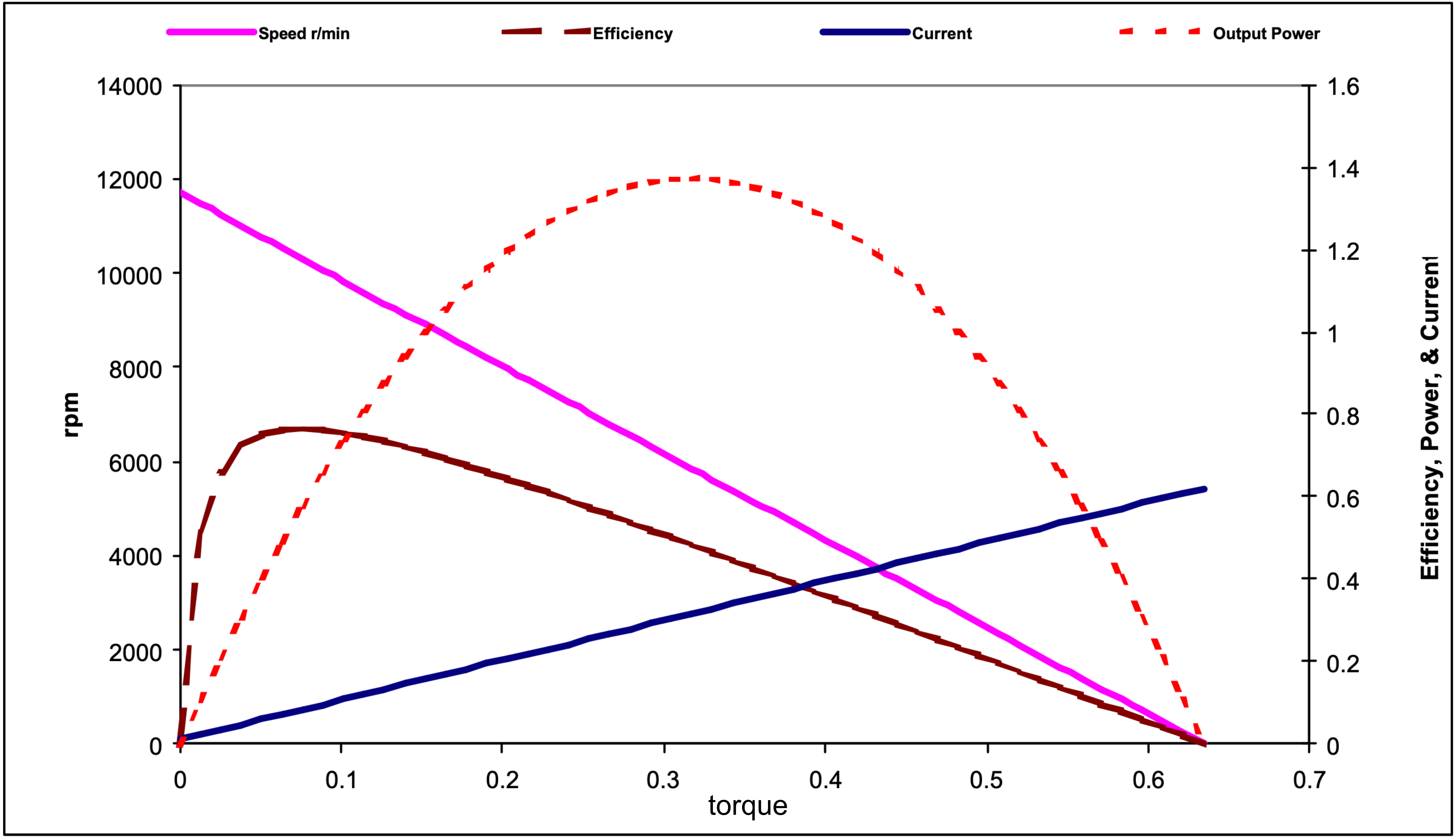




DC Motor Characteristics

- Torque: rotational equivalent to force (aka moment)
  - Measured in Nm (Newton meter)
  - Torque determines the rate of change of angular momentum
- Stall torque:
  - Maximum torque in a DC motor => maximum current => may melt coils
- Maximum energy efficiency:
  - At certain speed/ certain torque
- No-load-speed:
  - Maximum speed; little power consumption

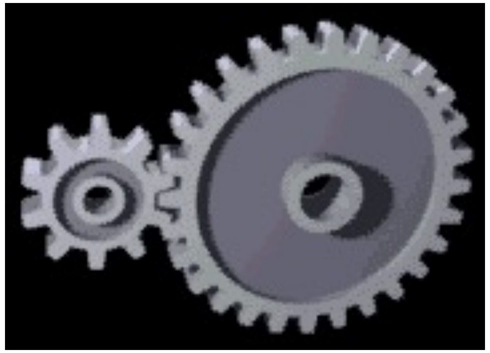
- High-power motors (e.g. humanoid robots) get very hot/ need cooling!


Speed r/min
Efficiency
Current
Output Power
rpm
torque
Efficiency, Power, & Current
14000
12000
10000
8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1.6
1.4
1.2
1
0.8

GEARS

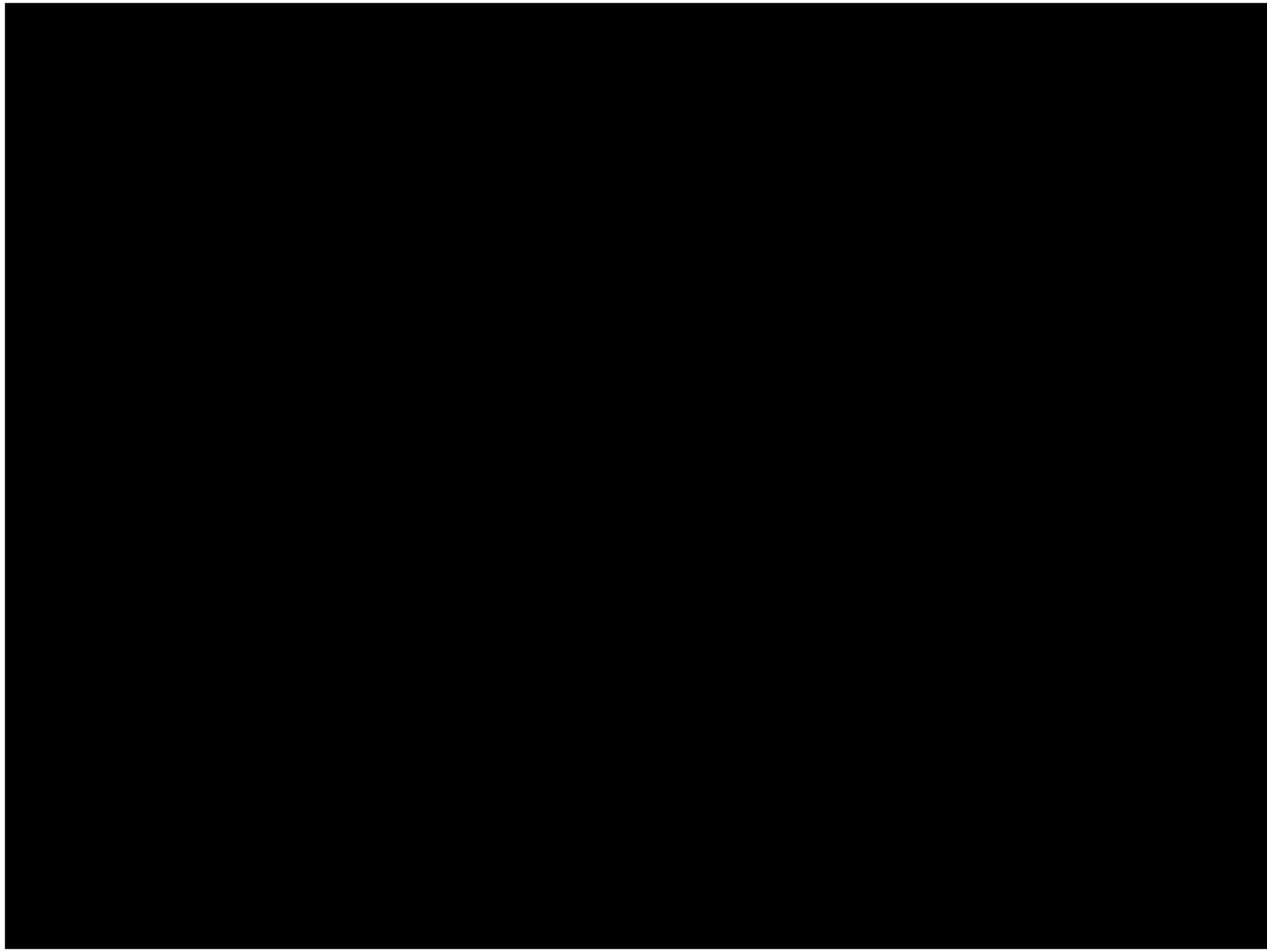
Gears

- Trade speed for torque
- See previous characteristic of DC motor: efficiency highest at high speeds
- Robotics: needs HIGH torque:
  - Inertia of mobile robot (high mass!)
  - Driving uphill
  - Robot arm: lift mass (object and robot arm) at long distances (lever!) – gravity!
- Most important property: Number of teeth => Gear Ratio $= \frac{DrivenGearTeeth}{DriveGearTeeth}$
- Torque = Motor Torque * Gear Ratio
- Speed = Motor Speed / Gear Ratio
- Teeth have same size =>
  gear diameter proportional to Number of teeth…



Gears

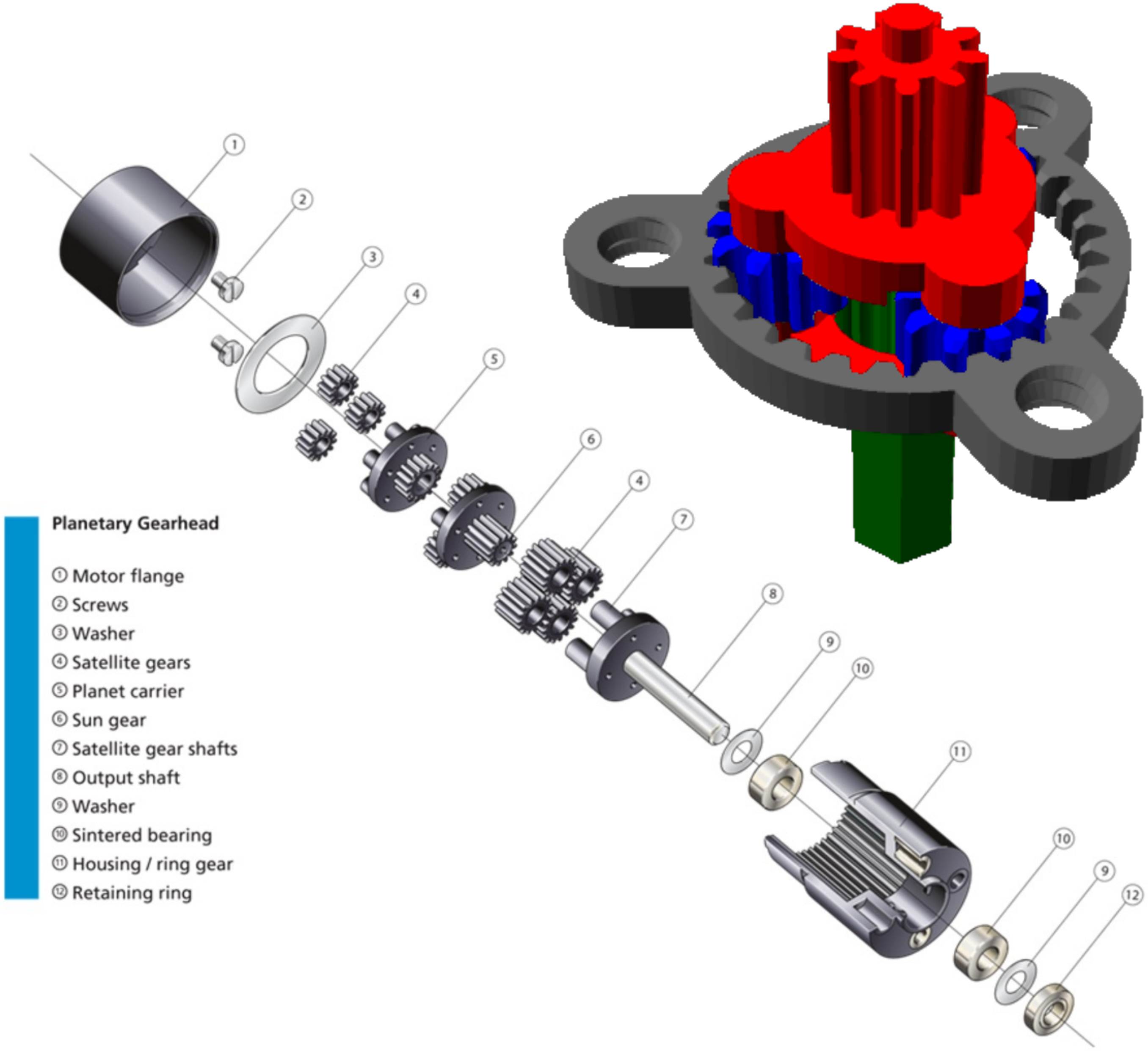
- Must be well designed to provide constant force transmission
  - Low wear/ low noise
- Back drivable: Can the wheel move the motor?
- Spur Gear reverses rotation direction!
- Backlash: when reversing direction: short moment of no force transmission => error in position estimate of wheel!



https://www.youtube.com/watch?v=8s4zm_ajxAA

Planetary Gear

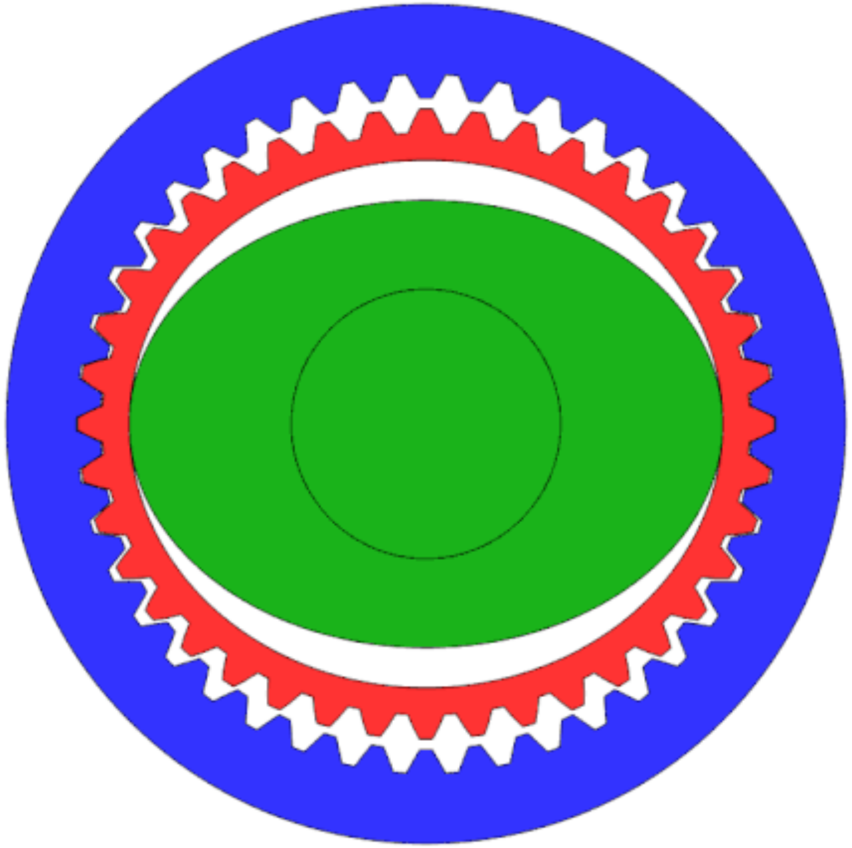
- Aka epicyclic gear train
- Quite common!
- Ratios: 3:1 … 1526:1
- Typical setup:
  - Sun (green) to motor
  - Carrier (red) output
  - Planets (blue): support
  - Ring (black): constraints the planets
  - => Ratio = 1:(1 + N_{Ring}/N_{Sun})



Harmonic Drive

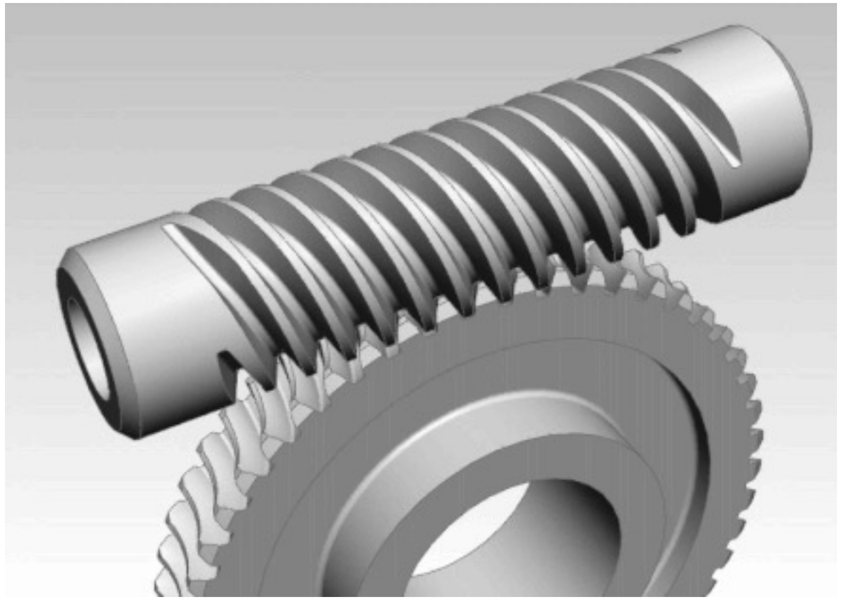
- High reduction in small volume (30:1 to 320:1)
- No backlash
- Light weight

- Used in robotics,
  e.g. robotic arms
  (e.g. our Schunk arm!)



$$\text{reduction ratio} = \frac{\text{flex spline teeth} - \text{circular spline teeth}}{\text{flex spline teeth}}$$

More Gears



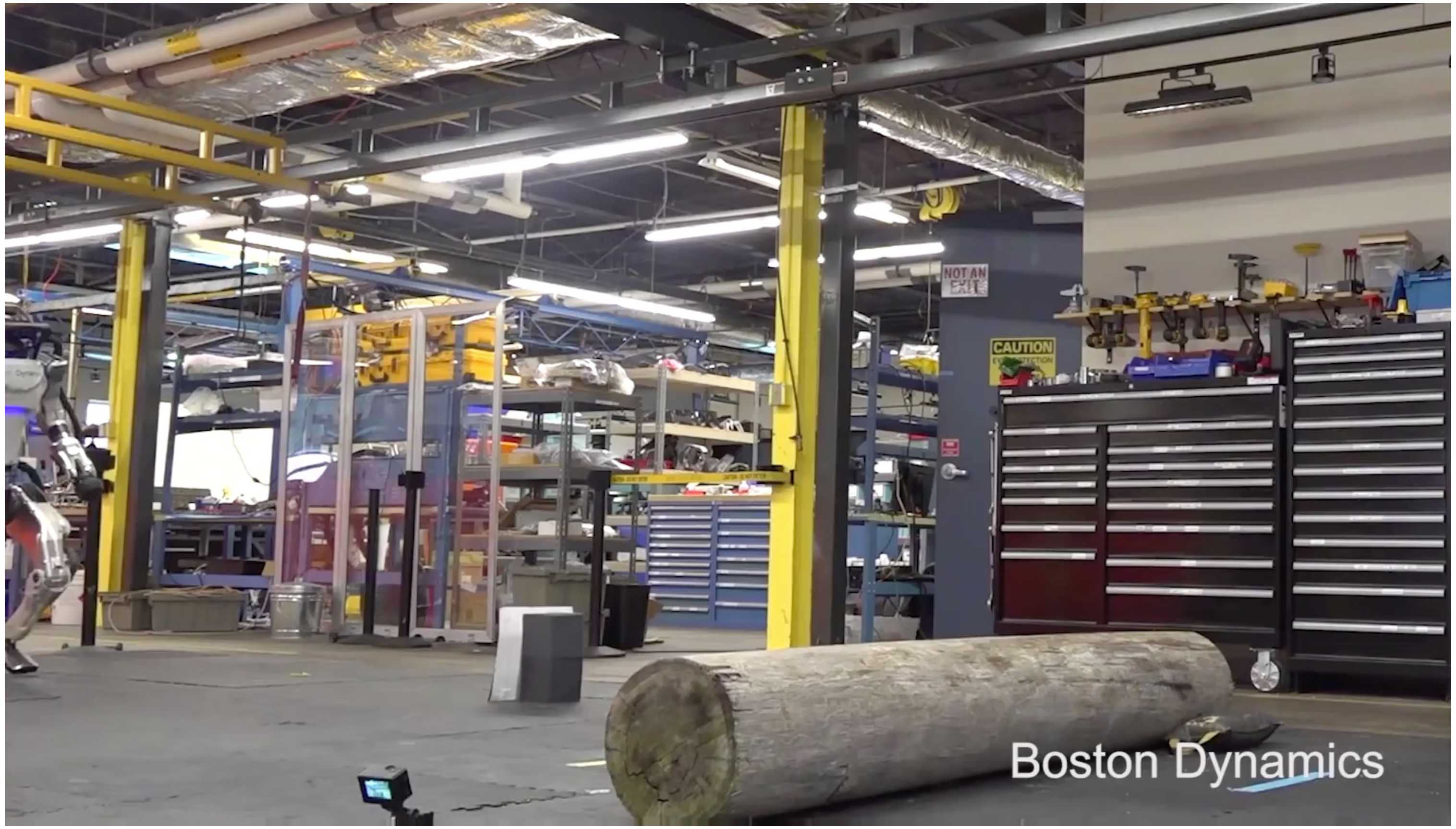
- Rack and pinion
  - linear drive
- Worm drive
  - Very high torque
  - Ratio: N_{Wheel} : 1
  - Locking (not back-drivable) gear)
- Bevel gear
  - Mainly to change direction

Summary: Mechatronics of Controlling a Wheel

1. Use Encoder and PID to control the speed of the motor
2. Send PWM signals to Motor Driver using dedicated circuits in CPU
3. Use Motor Driver to send high voltage & high current to motor
4. Use DC Brushed or Brushless motor to drive gear
5. Use Gear to get the required torque/ speed
6. Connect wheel to gear

ALTERNATIVES

Hydraulics



- 28 Hydraulic actuated joints
- Why?
  - Compact actuators with high torque – do not get hot!
  - Low mass
  - One central, highly efficient motor to pressurize the hydraulic fluid
- Actuation controlled via controlling valves

Synthetic Muscles

- Electroactive polymer: Apply voltage => change shape by 30% OR: …

Others

- Piezoelectric actuation
  - Small motions only
  - Very fast and precise
- Pneumatic actuator
  - Uses compressible gas
- Thermal-driven actuation