Natural, Mathematical & Physical Sciences

31/1/2022

Dr Francesco Ciriello

Department of Engineering

4CCE1MCP: Design, Making a Connection

Week 24

Introduction to Control Design

Learning Outcomes

- Explain basic principles of controls and explain in words a non-expert would understand what is meant by targets, controller, actuation, plant, measurements
- Implement open and closed loop controllers to reach a setpoint target
- Implement logic-driven controllers using state machines
- Implement a controller to complete your individual coursework project

Agenda

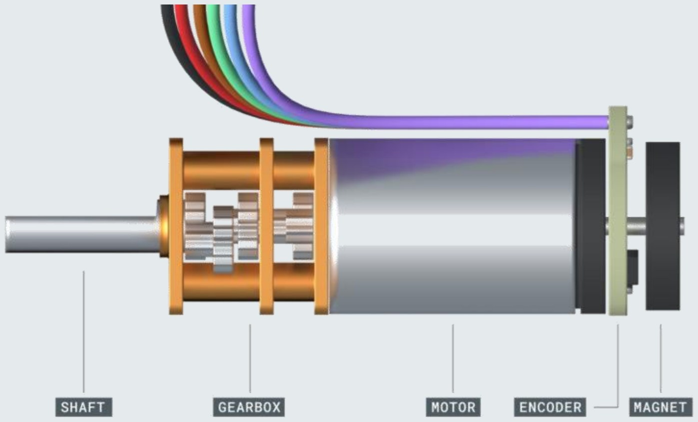
Components of a Control System

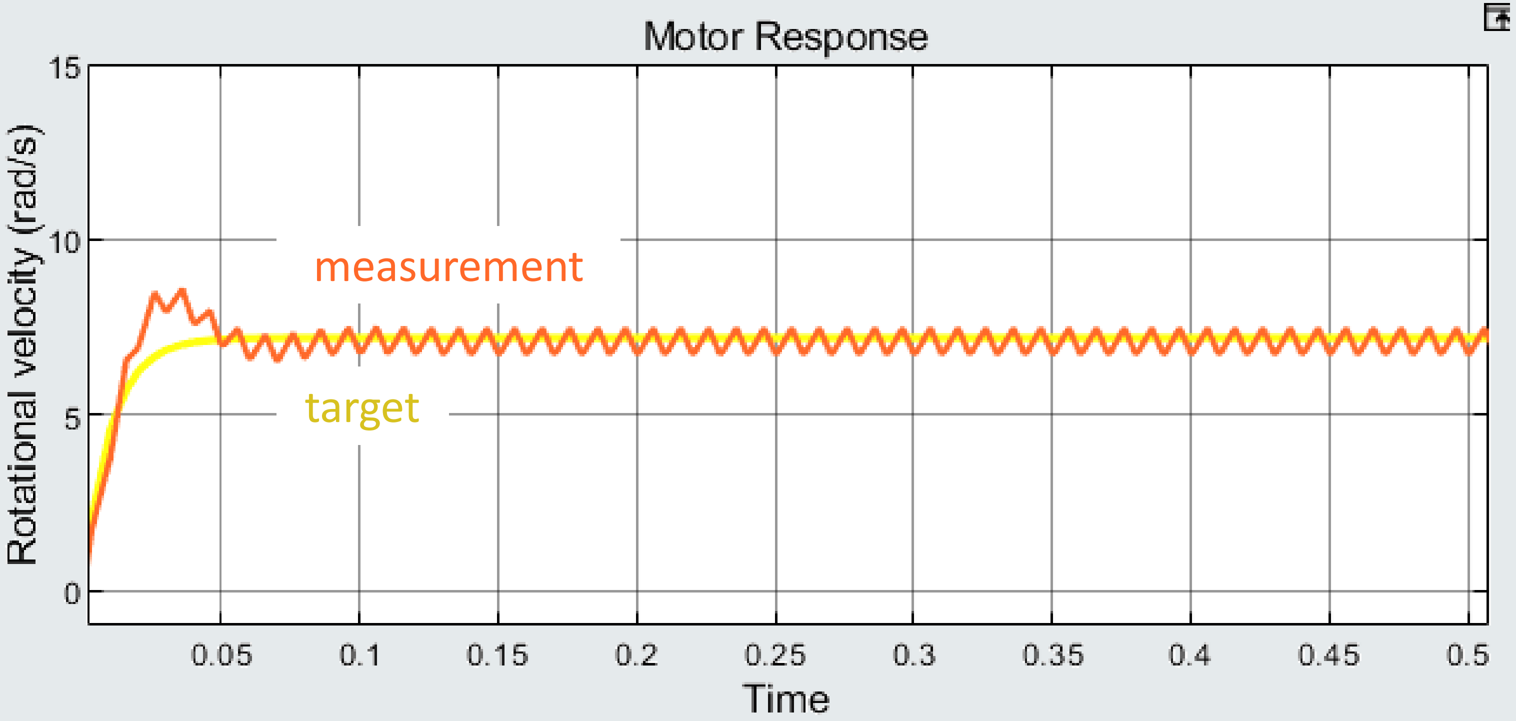
Plant modelling

Control
- Open loop control
- Feedback control
- Logic-driven control
- Kinematic vs Dynamic control

Control System Design

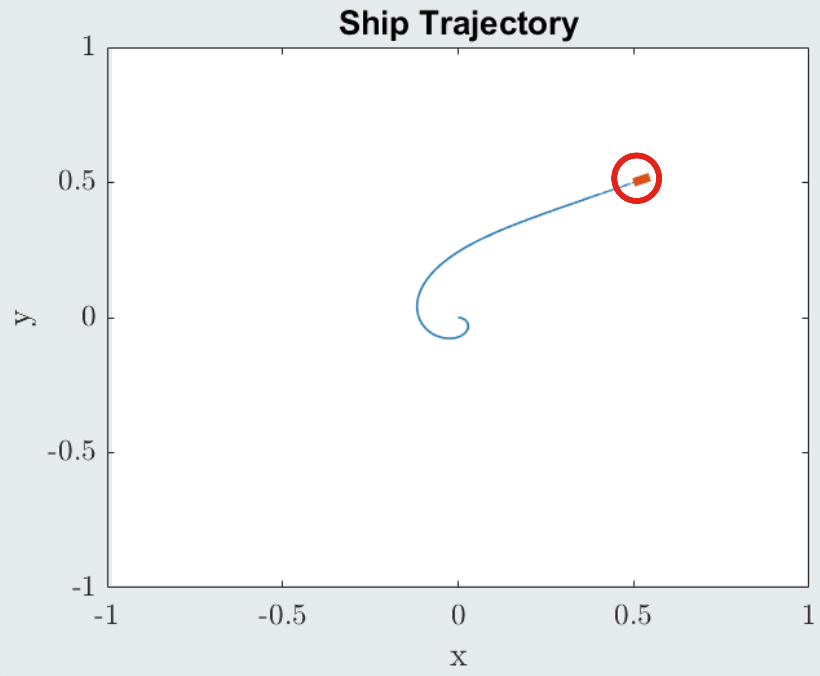
How do we design **components** to behave as intended?

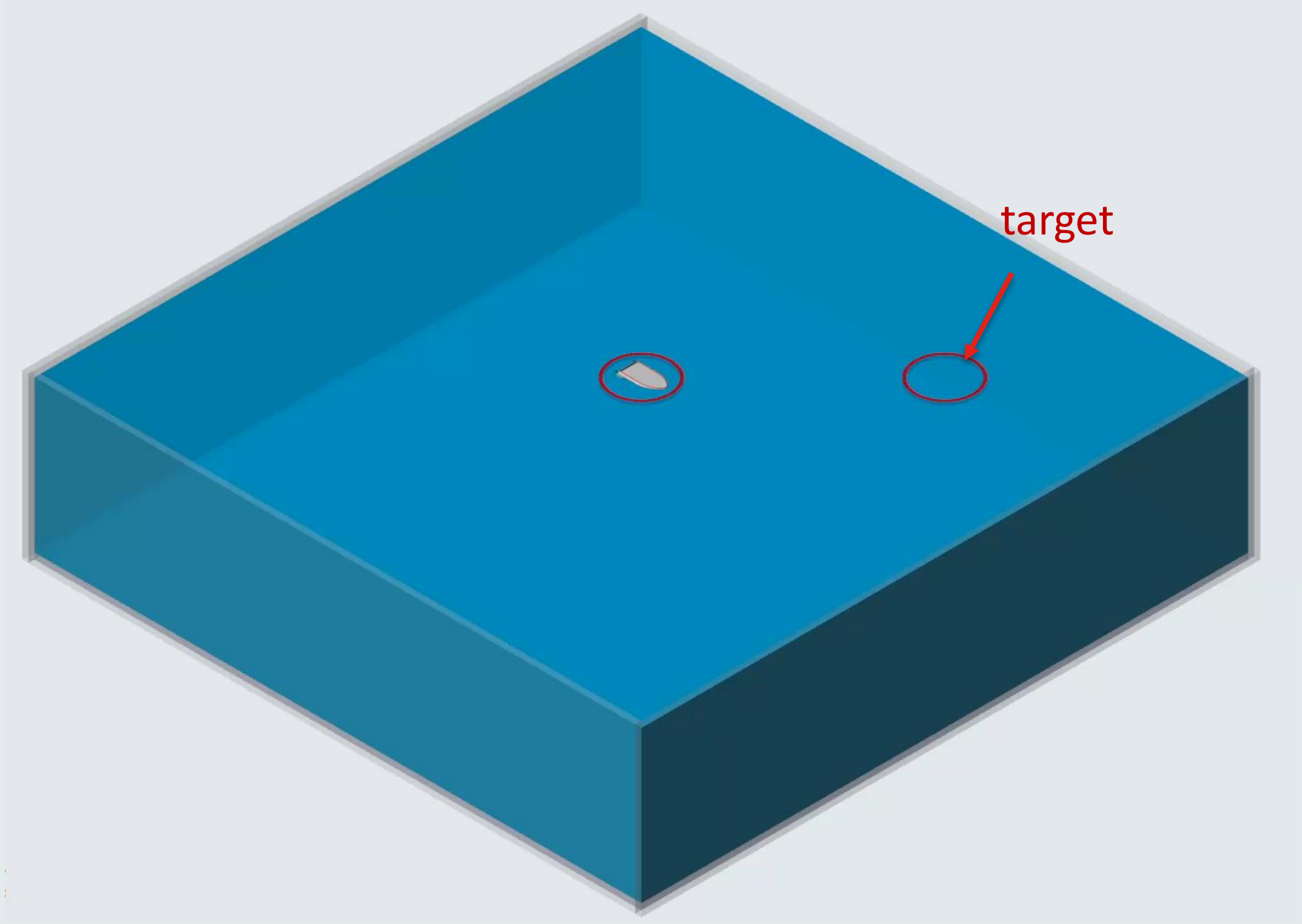




Control System Design

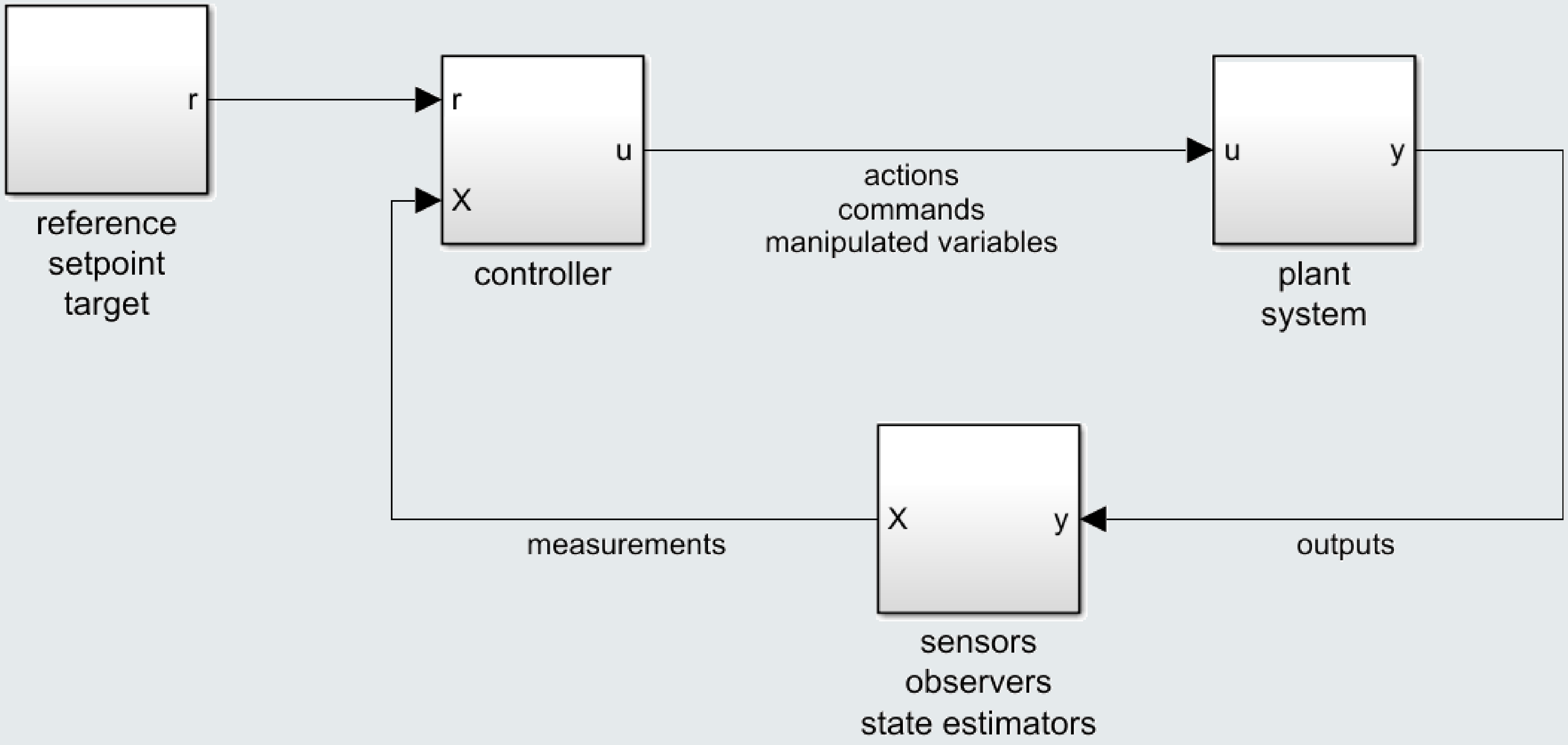
How do we design **systems** to behave as intended?



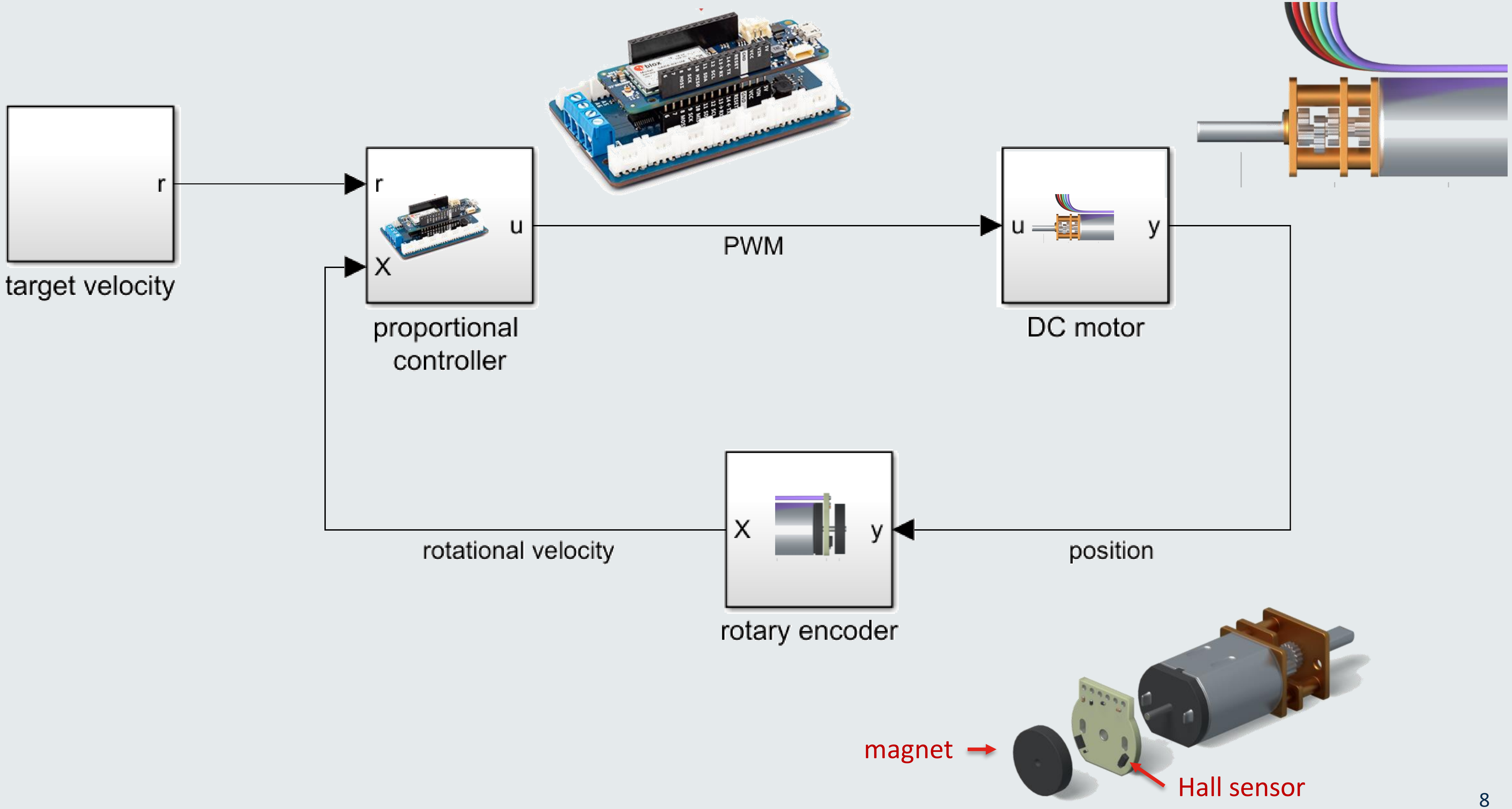


Control Systems

Control diagrams are used to design control systems

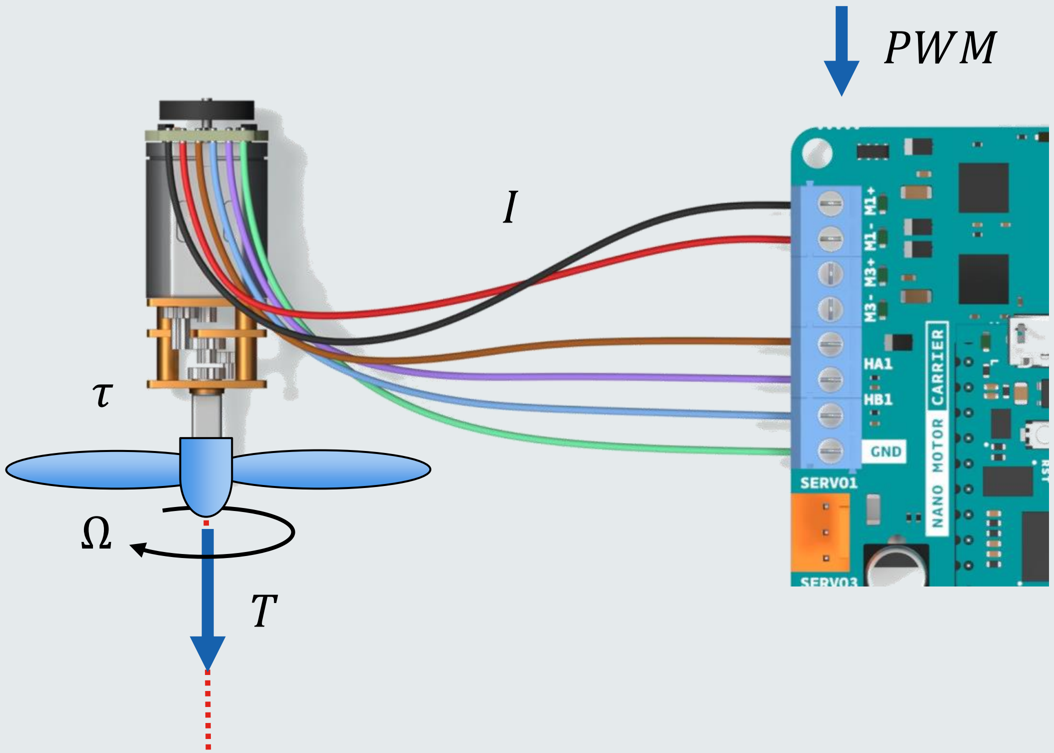


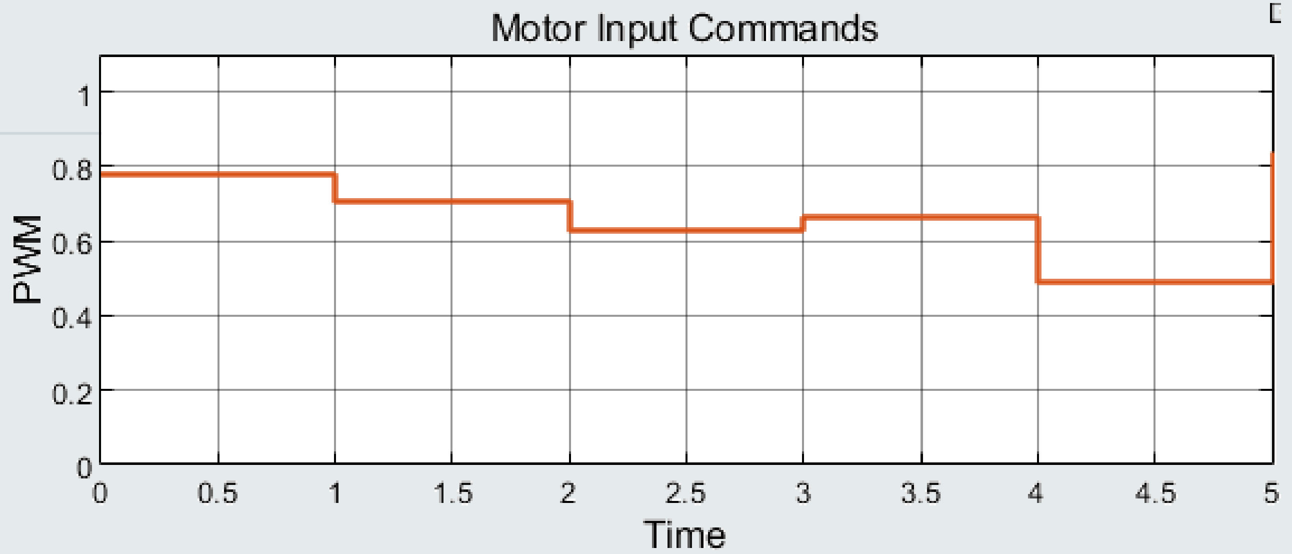
Control Systems

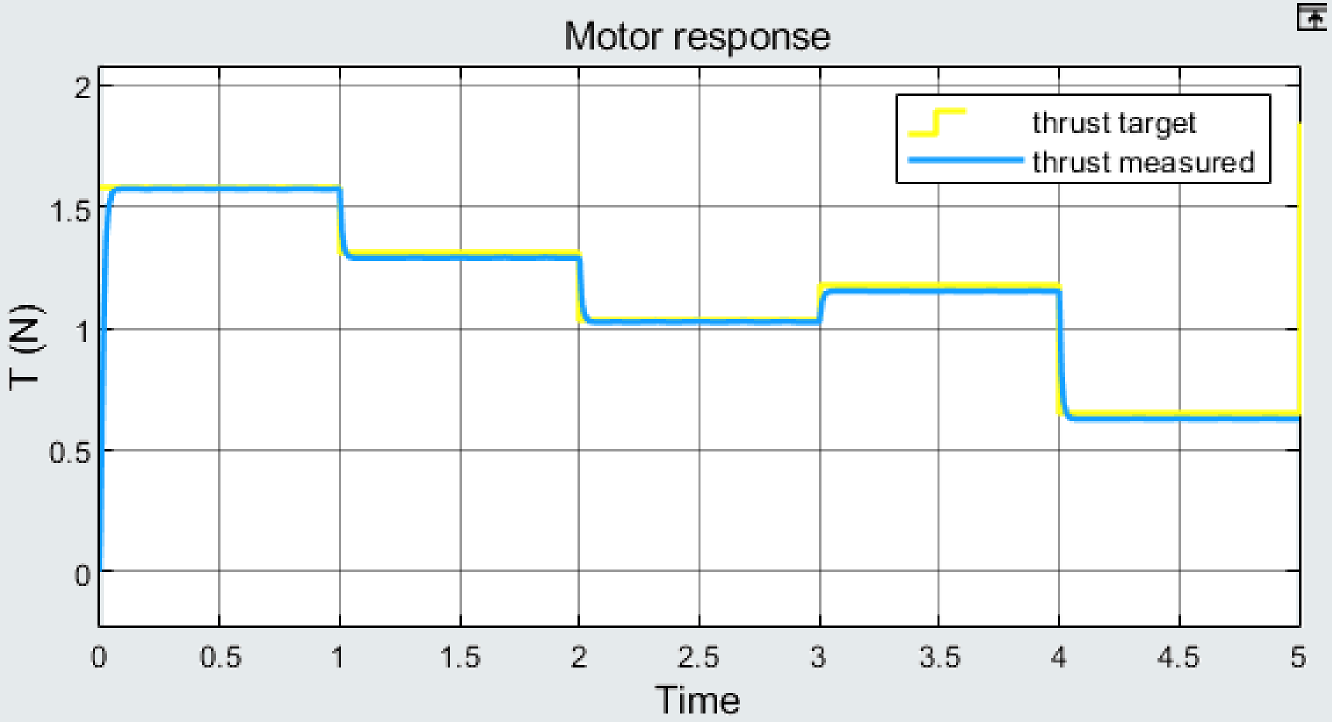


Motor Control

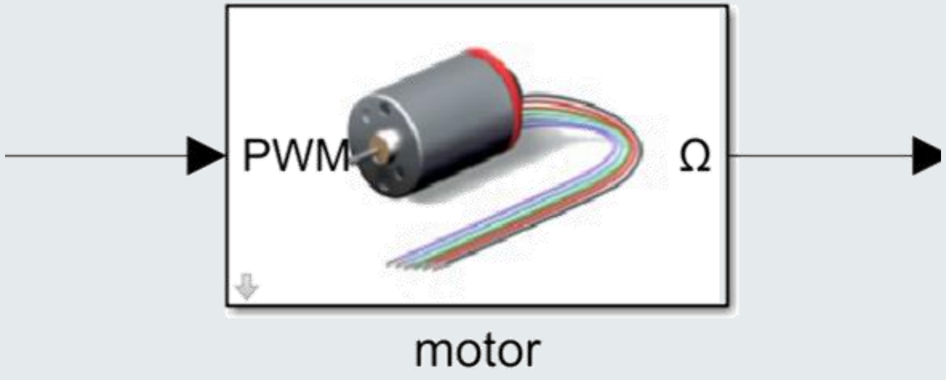
Control PWM to motor to generate a target thrust

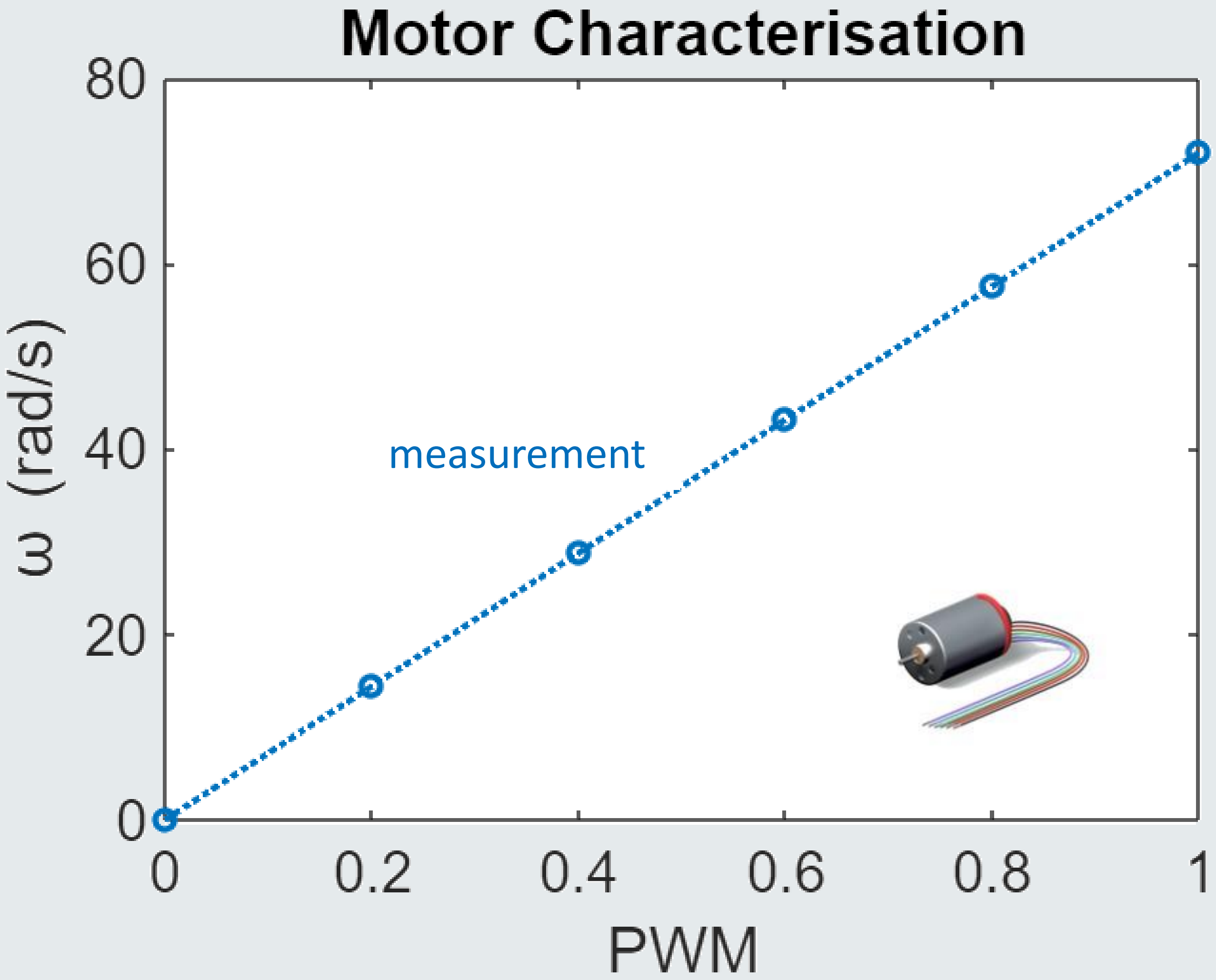




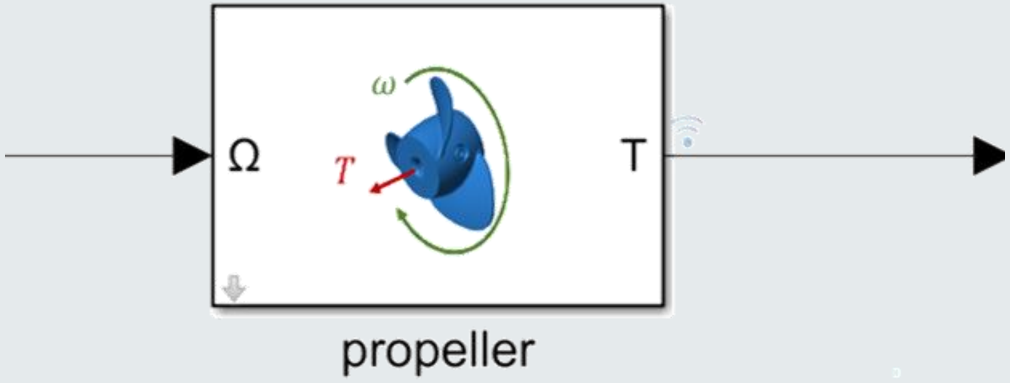


Components

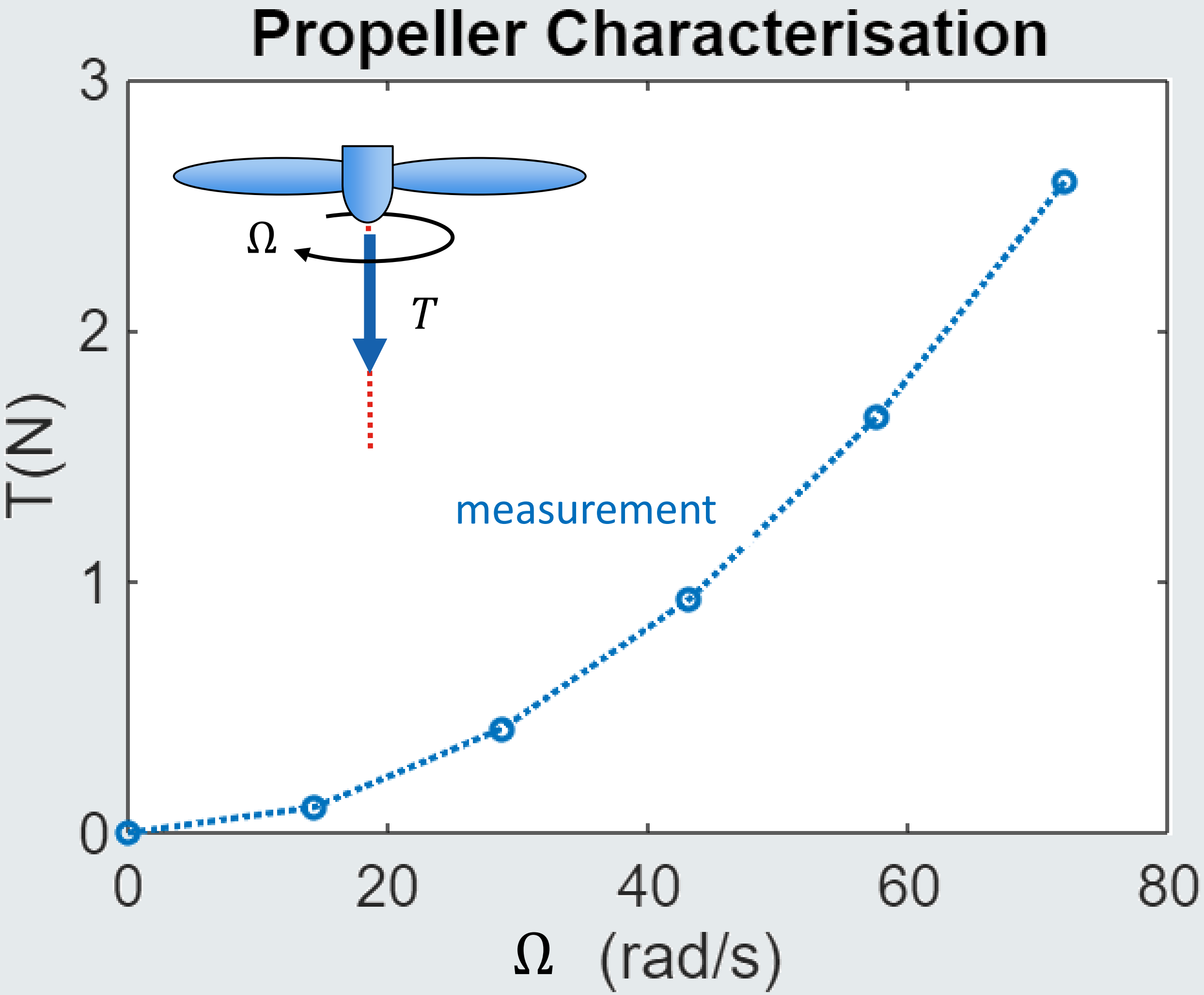

PWM
Ω
motor


Motor Characterisation
80
60
40
20
0
ω (rad/s)
0
0.2
0.4
0.6
0.8
1
PWM
measurement

Components

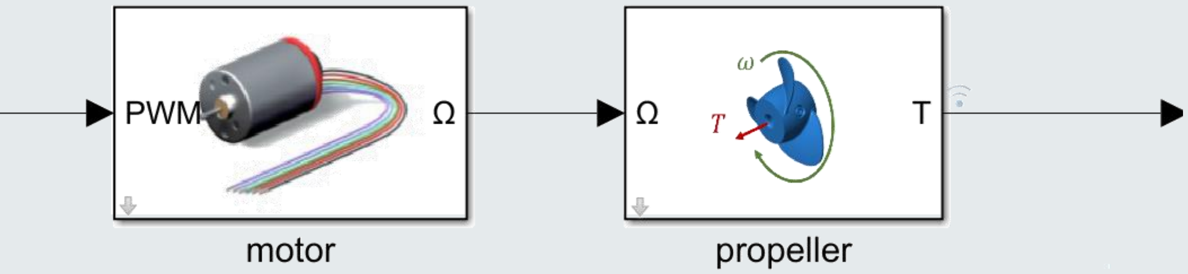


Ω
ω
T
T
propeller

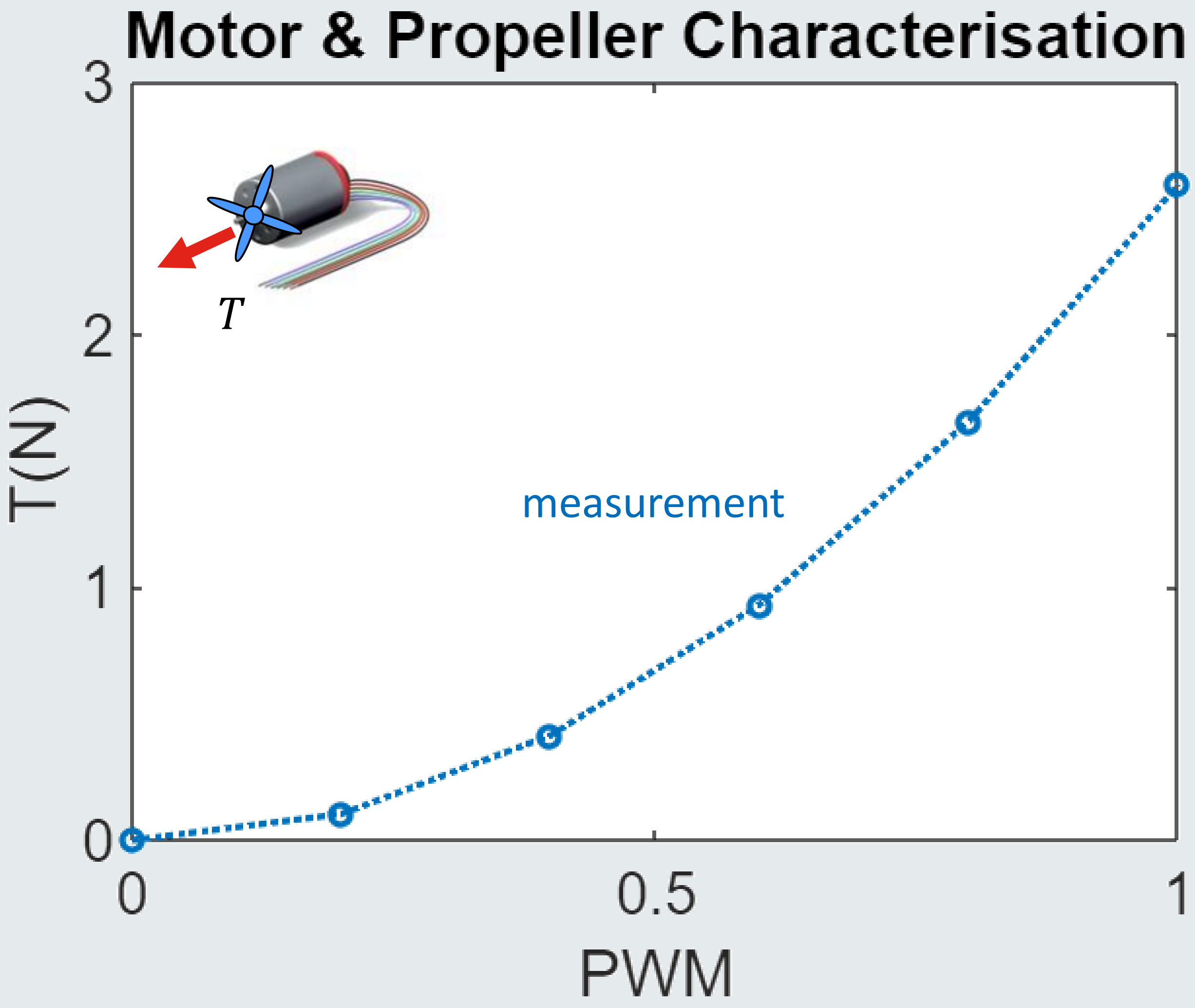


Propeller Characterisation
T(N)
3
2
1
0
0
20
40
60
80
Ω (rad/s)
Ω
T
measurement

Systems



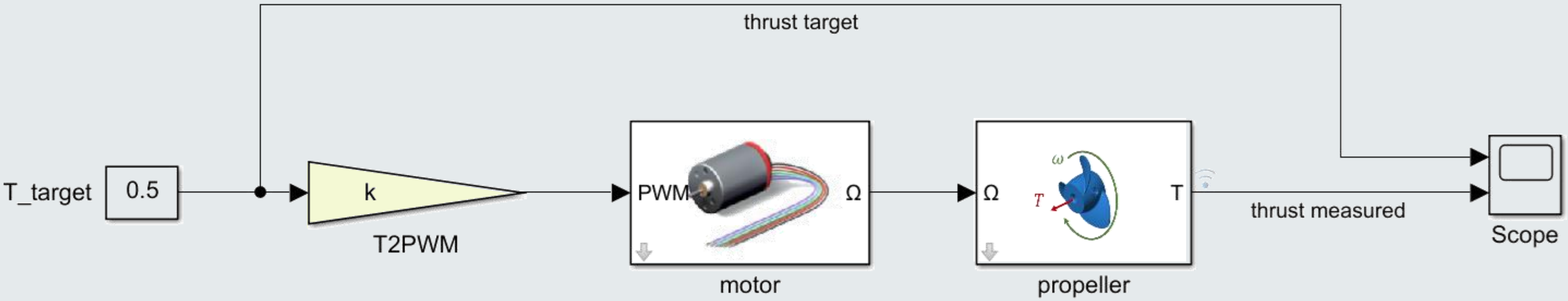
PWM
Ω
motor
Ω
T
propeller



Motor & Propeller Characterisation
T(N)
PWM
measurement
T
0
1
2
3
0
0.5
1

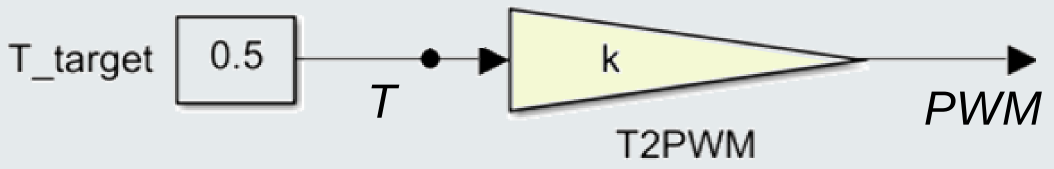
Open-loop (Feedforward) Control

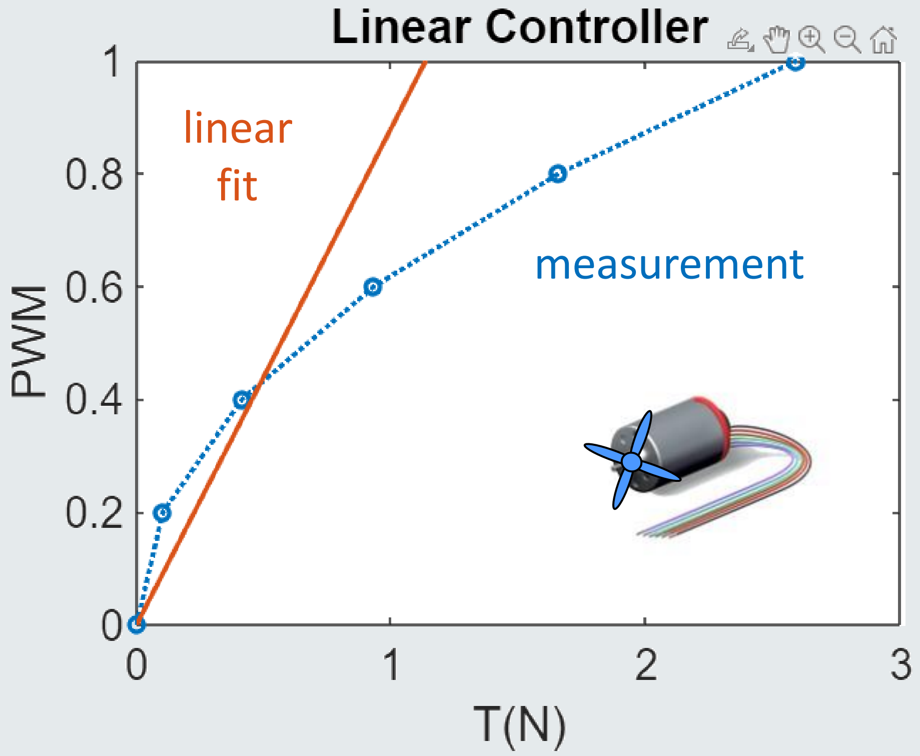
Model-free Proportional Control law $PWM = k\,T_{target}$



Open-loop (Feedforward) Control

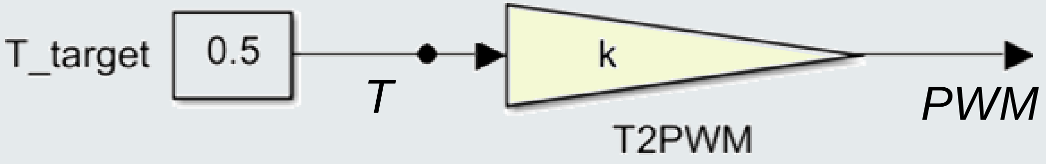
$$PWM = k\,T_{target}$$

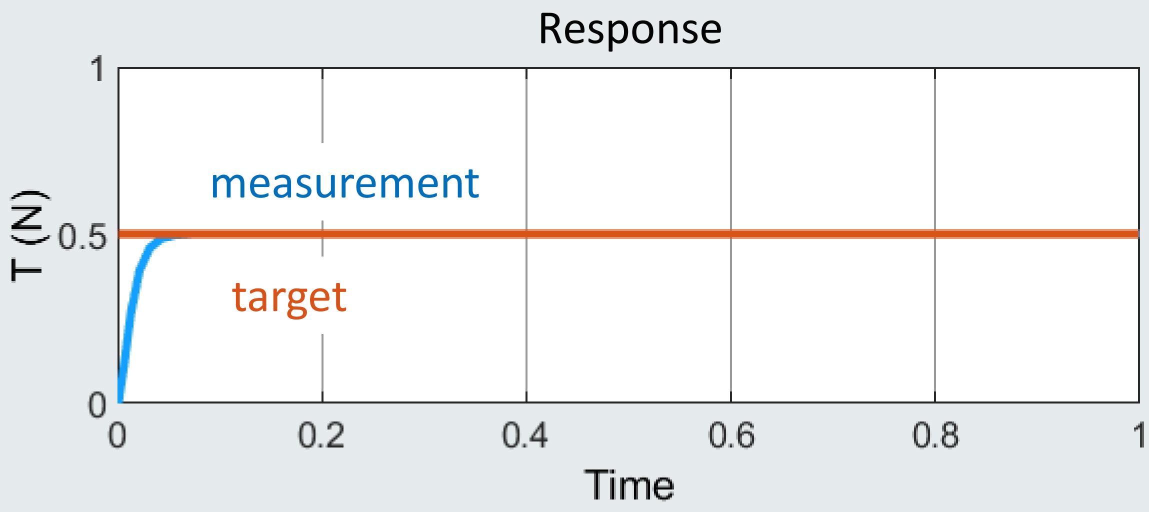


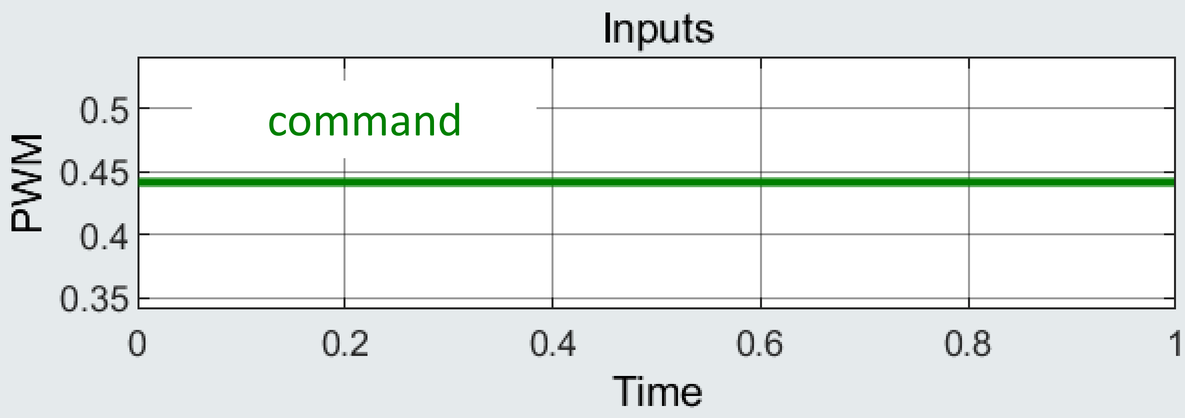


Open-loop (Feedforward) Control

$$PWM = k\,T_{target}$$

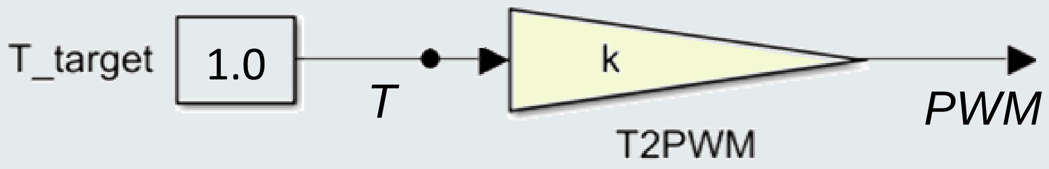


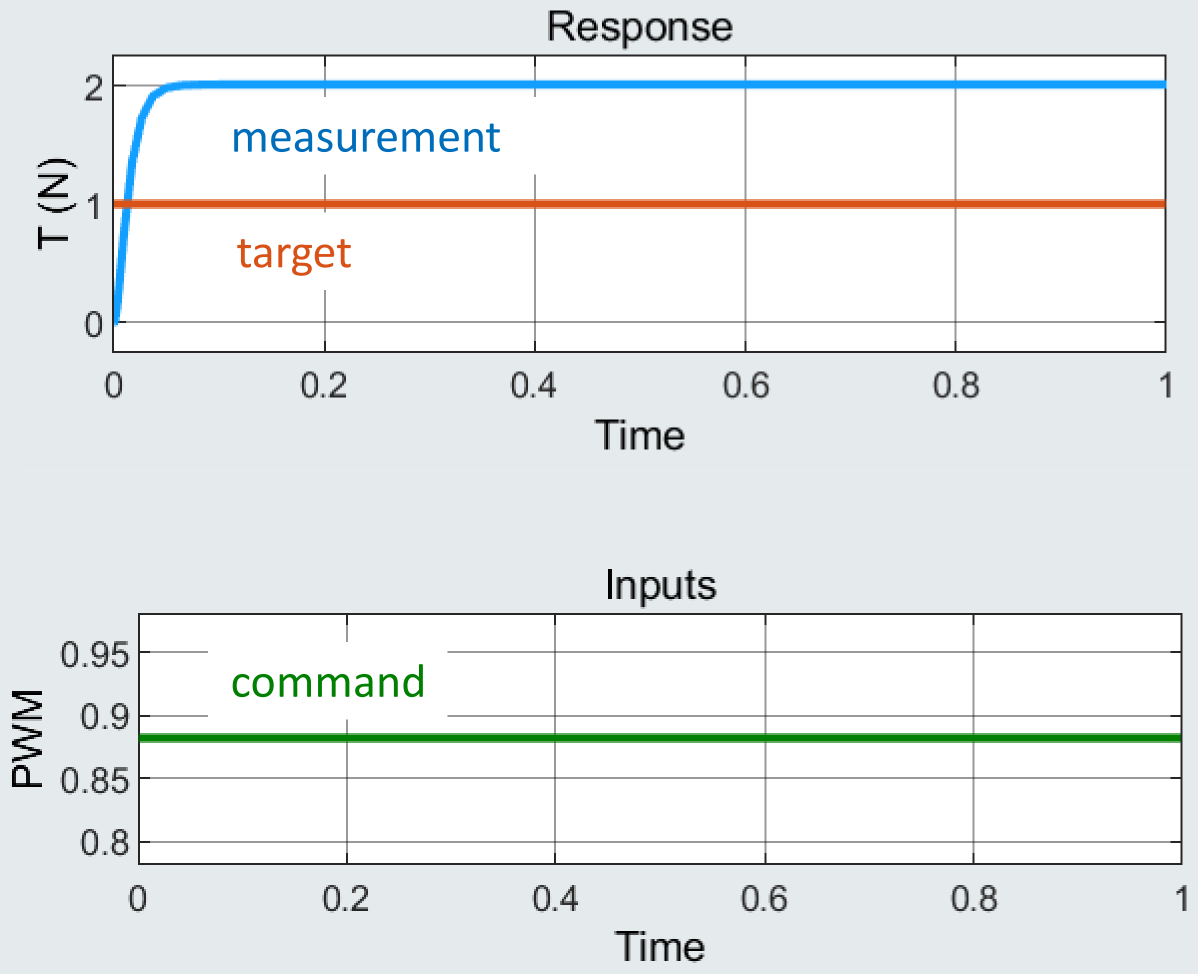




Open-loop (Feedforward) Control

$$PWM = k\,T_{target}$$

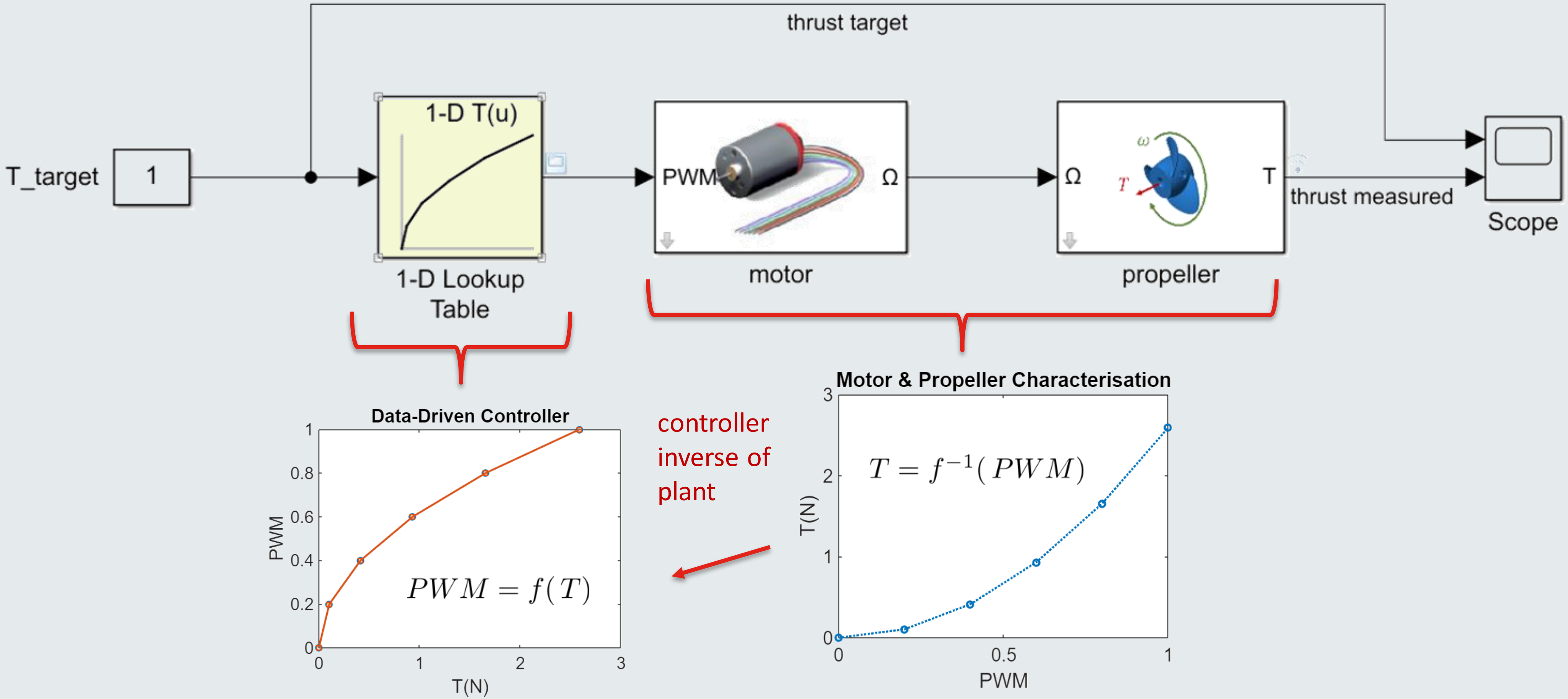




Open-loop (Feedforward) Control

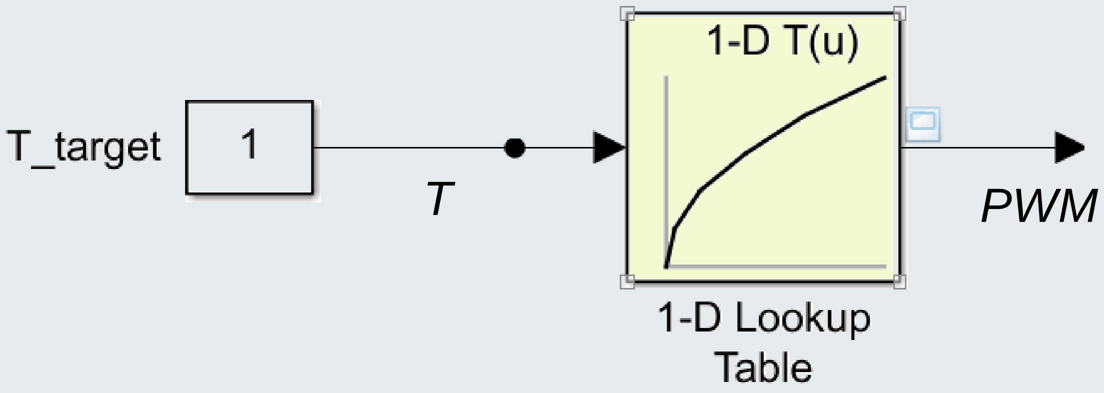
Data-driven Look-up Control law

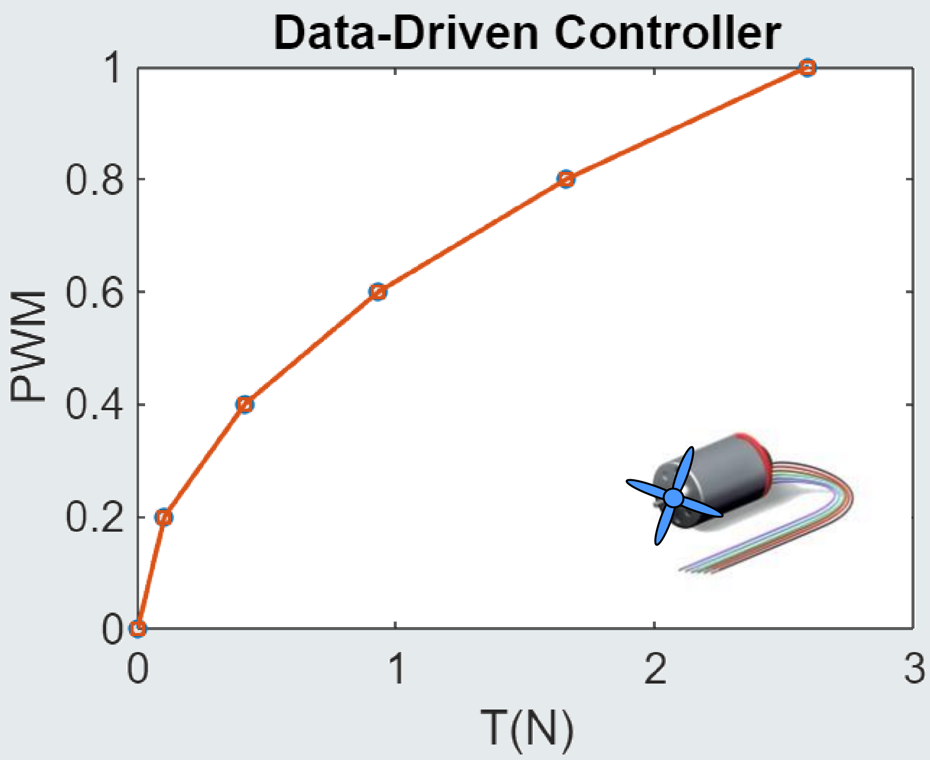
$$PWM = f(T_{target})$$



Open-loop (Feedforward) Control

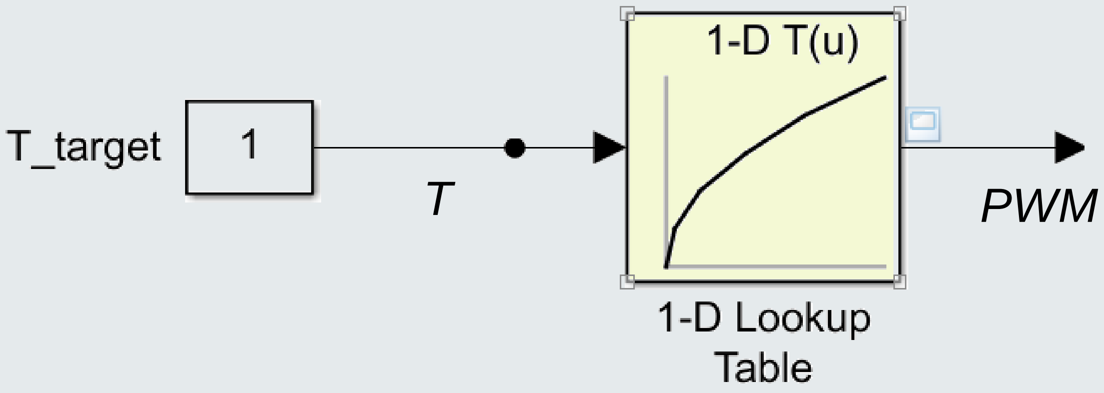
$$PWM = f(T_{target})$$

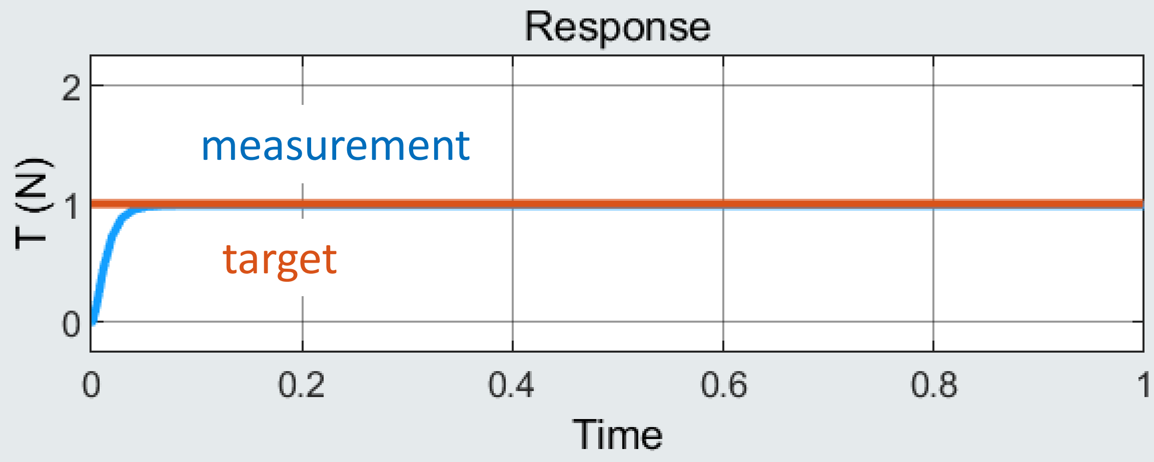


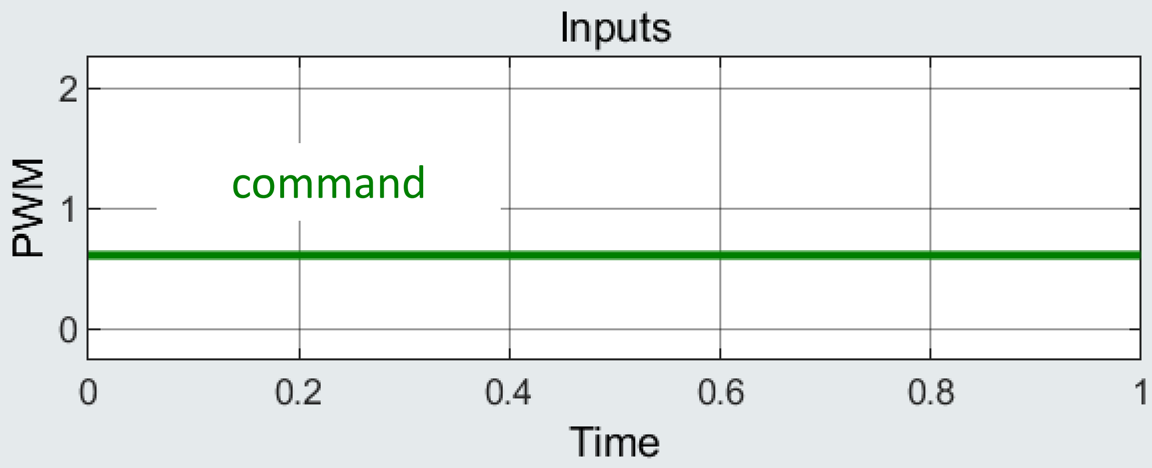


Open-loop (Feedforward) Control

$$PWM = f(T_{target})$$

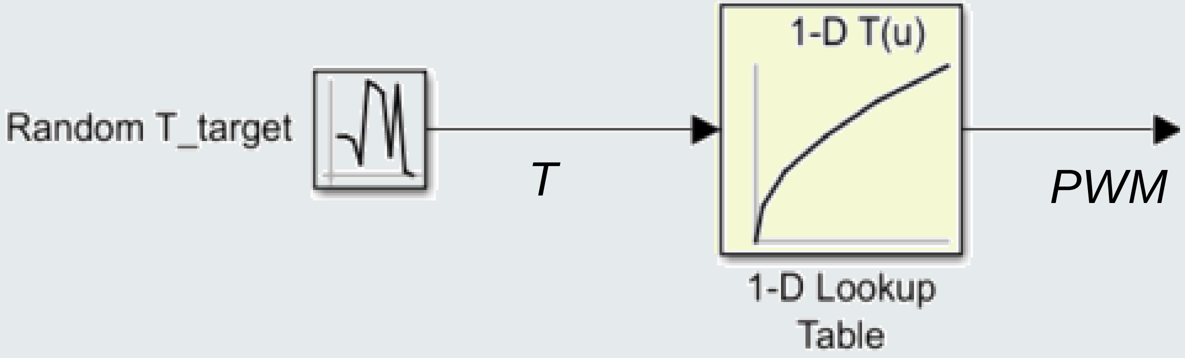


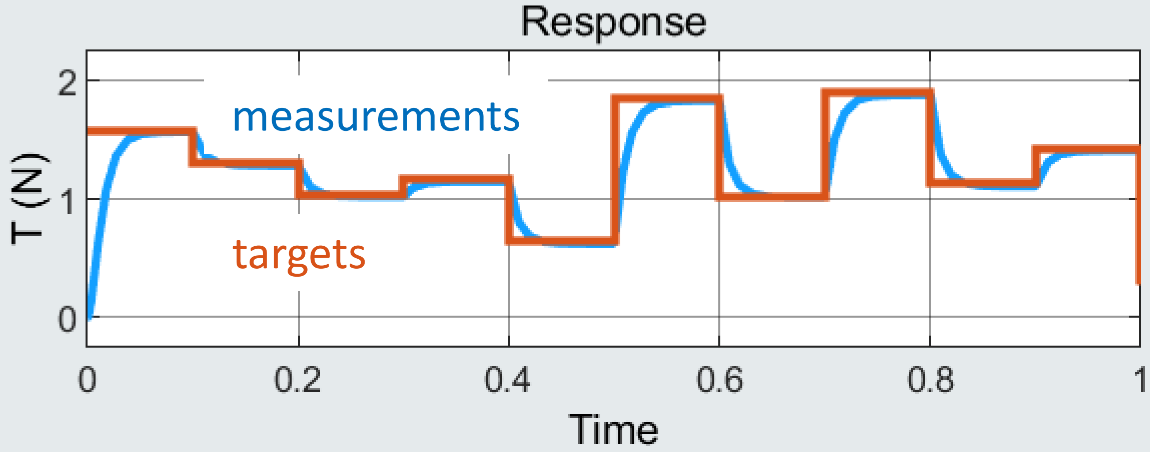


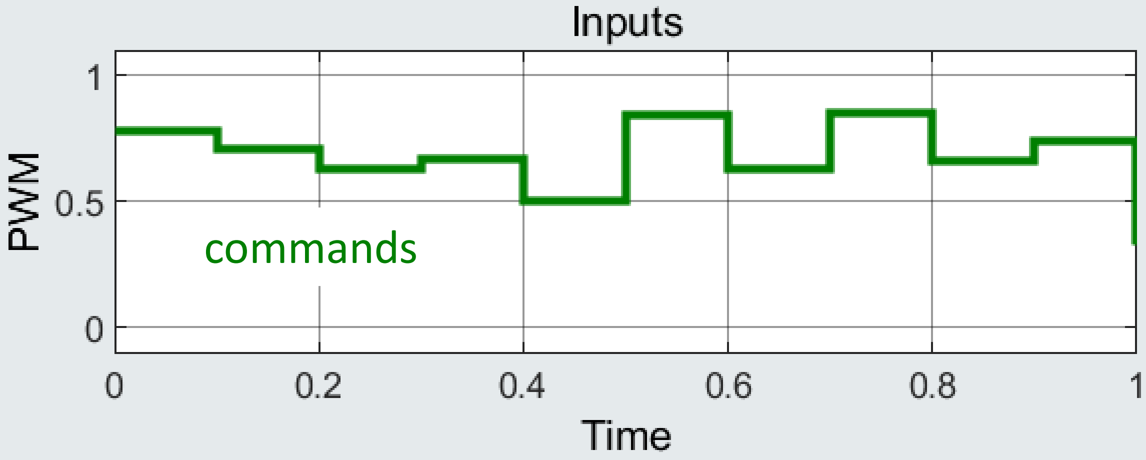


Open-loop (Feedforward) Control

$$PWM = f(T_{target})$$



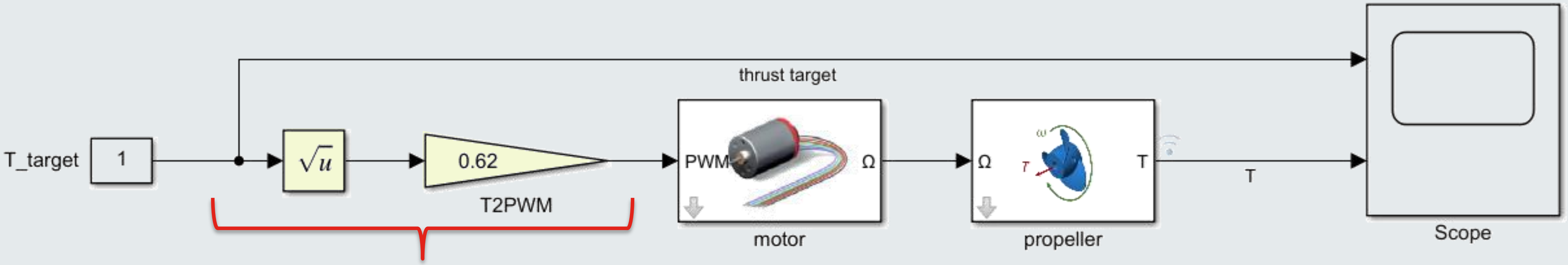




Open-loop (Feedforward) Control

Model-based – “inverse of the plant”

$$PWM = k\sqrt{T_{target}}$$



$$PWM \propto \sqrt{\mathrm{T}}$$

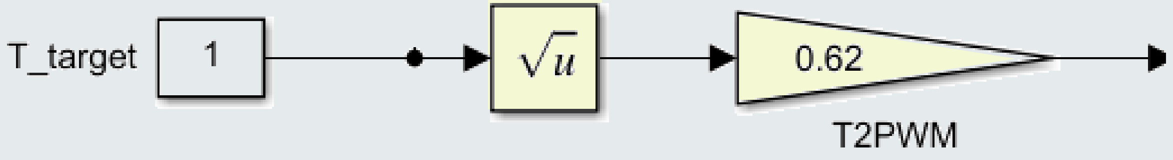
$$\Omega \propto \mathrm{PWM} \qquad T \propto \Omega^2$$

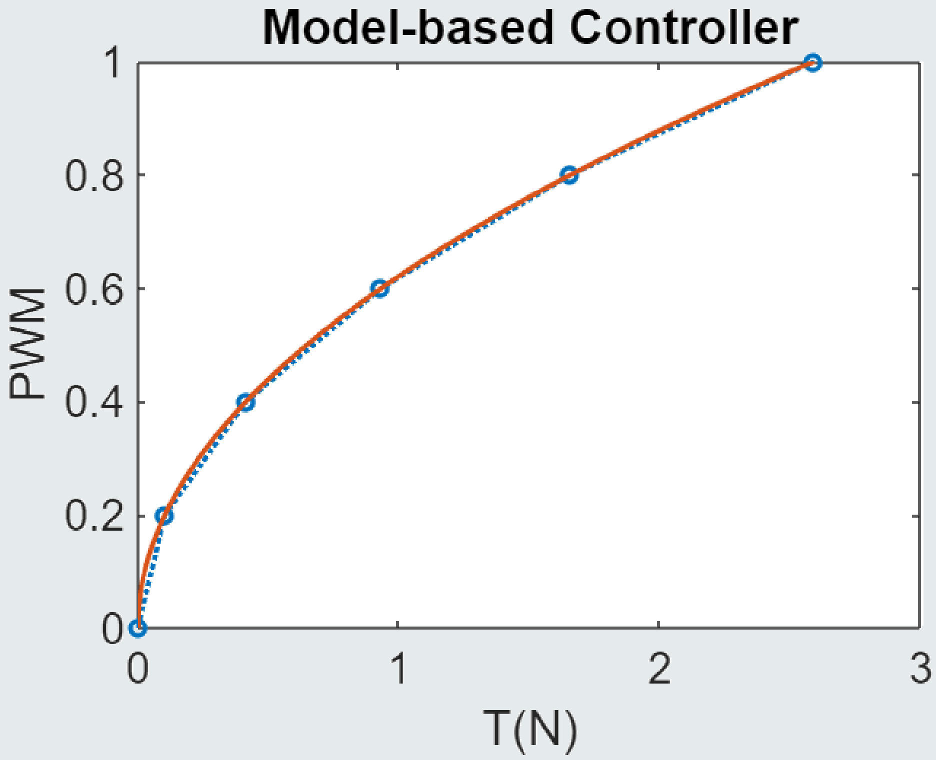
$$T \propto \mathrm{PWM}^2$$

Open-loop (Feedforward) Control

Model-based – “inverse of the plant”

$$PWM = k\sqrt{T_{target}}$$

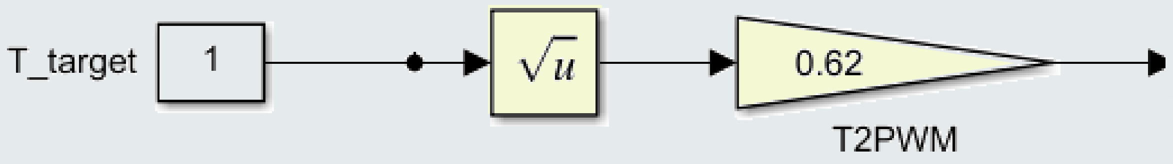


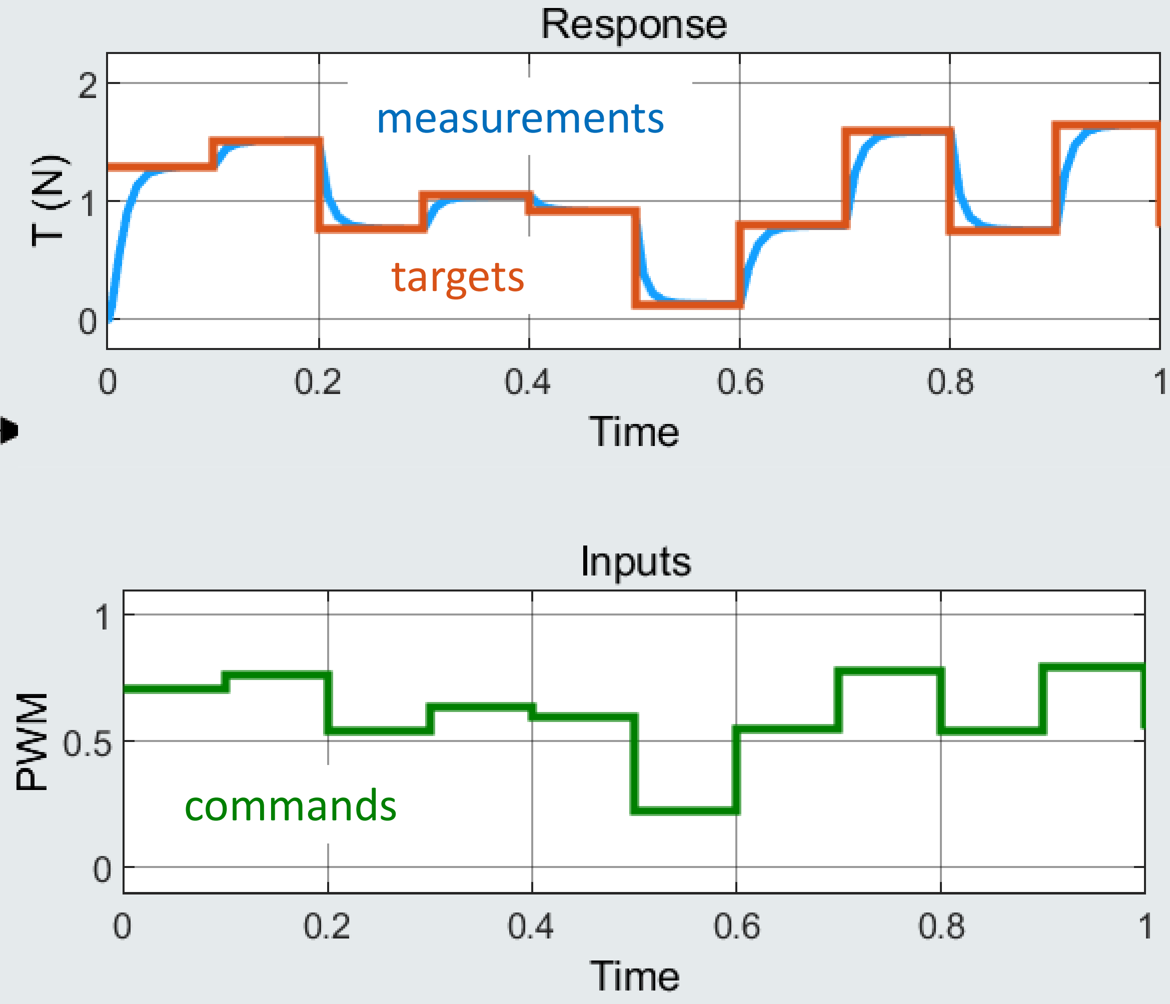


Open-loop (Feedforward) Control

Model-based – “inverse of the plant”

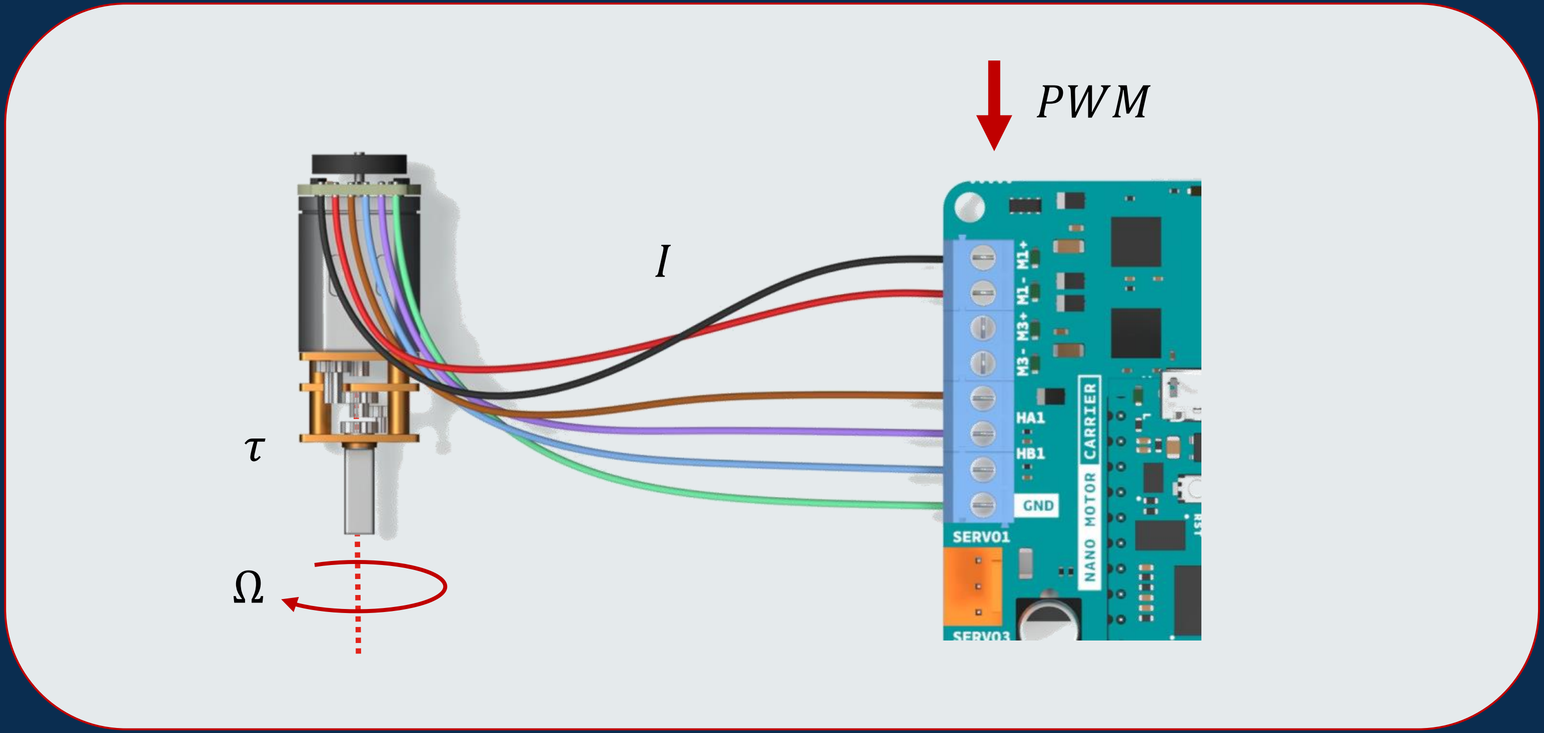
$$PWM = k\sqrt{T_{target}}$$



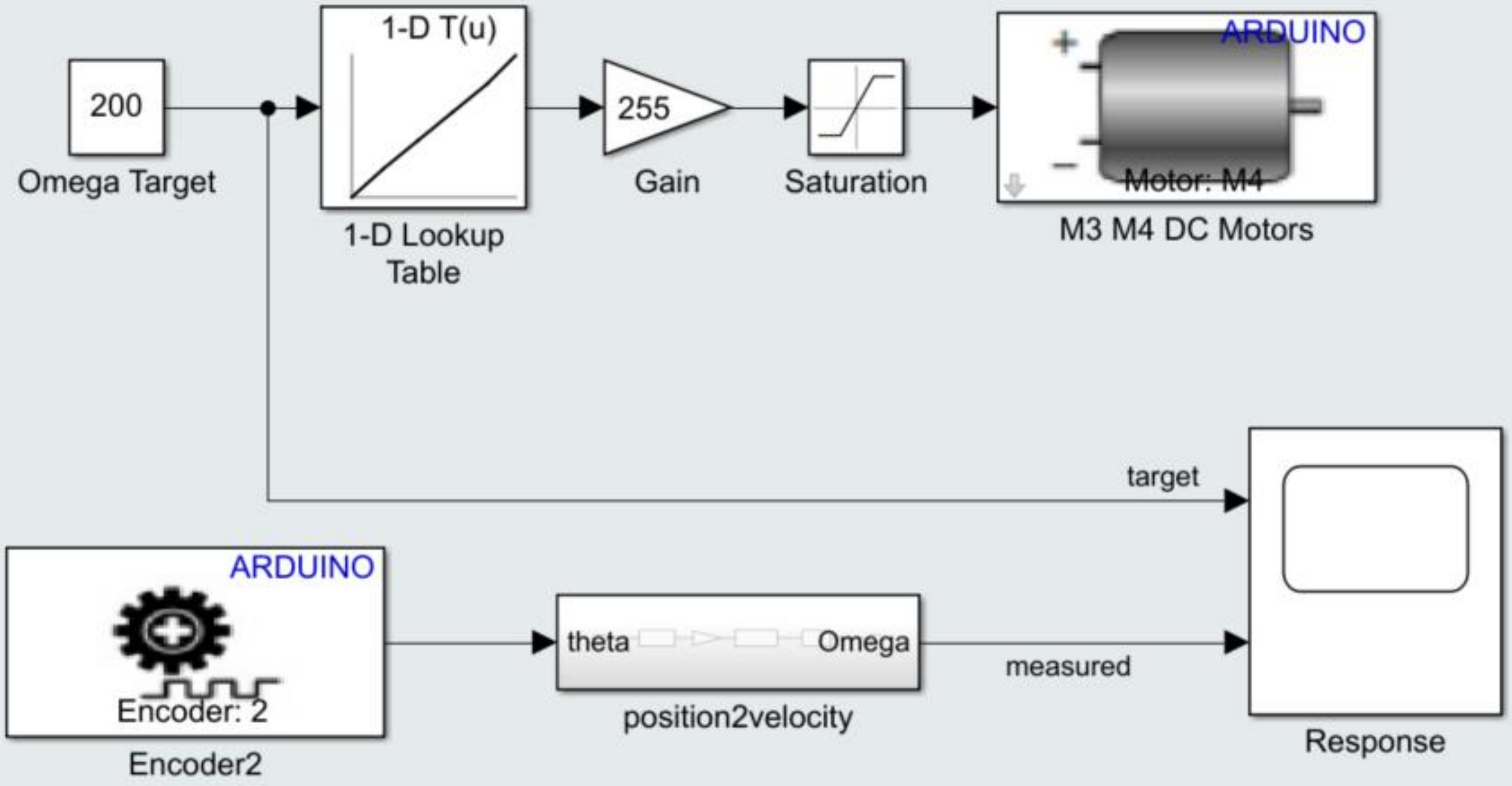


Motor Control

Example: Open-loop control on hardware

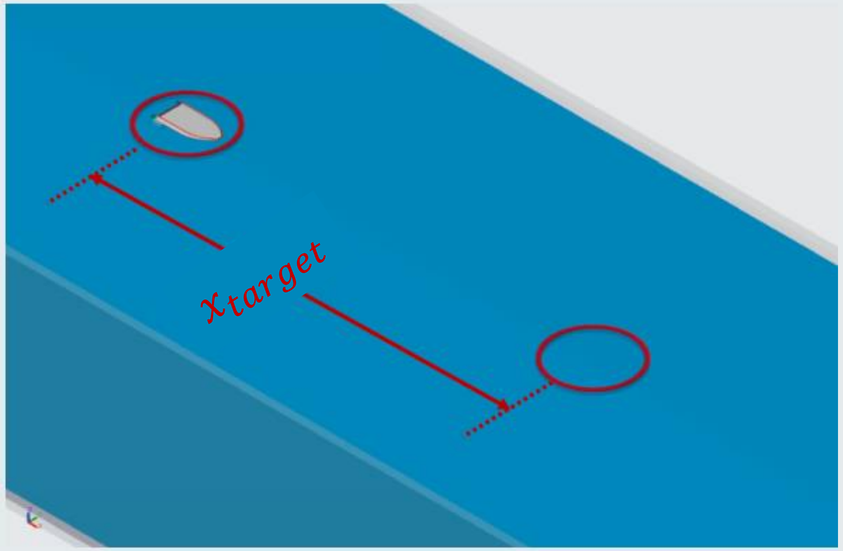


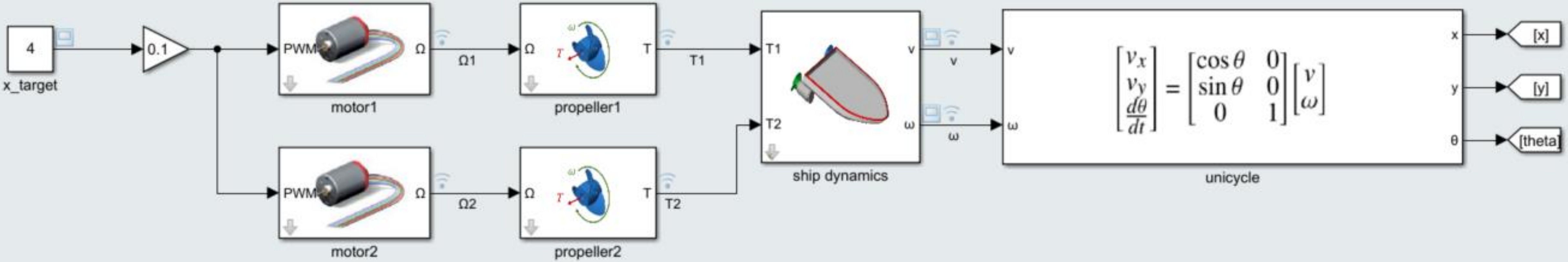
Demonstration: Open-loop Control DC Motor with Encoder



Open-loop (Feedforward) Control

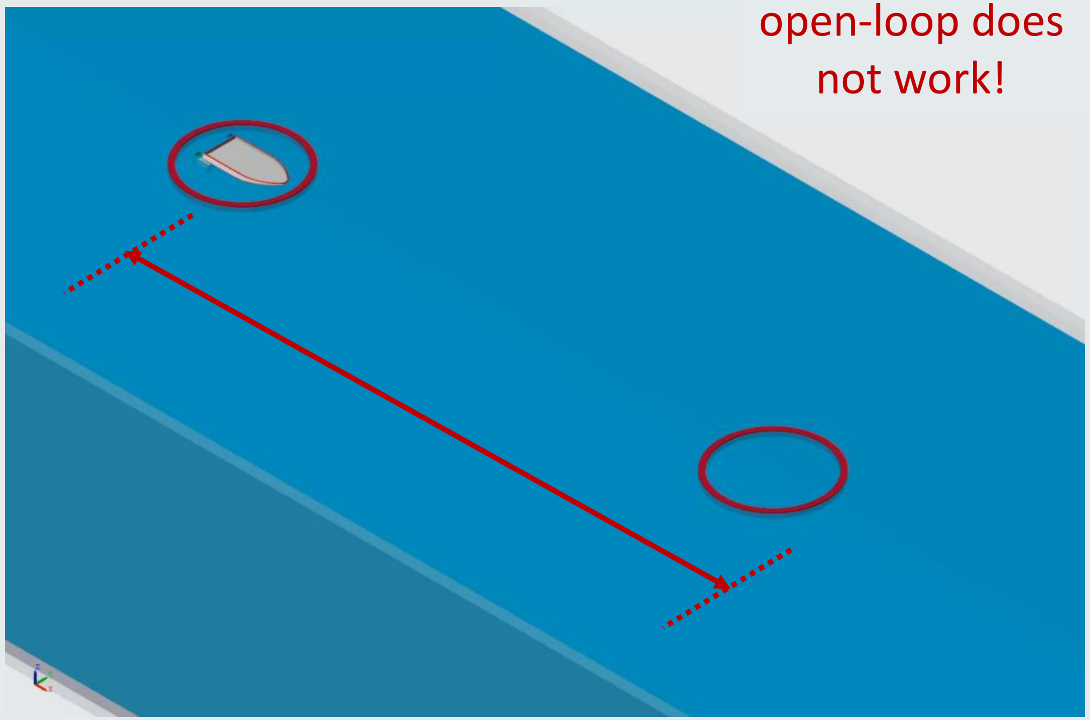
$$PWM = k\,x_{target}$$



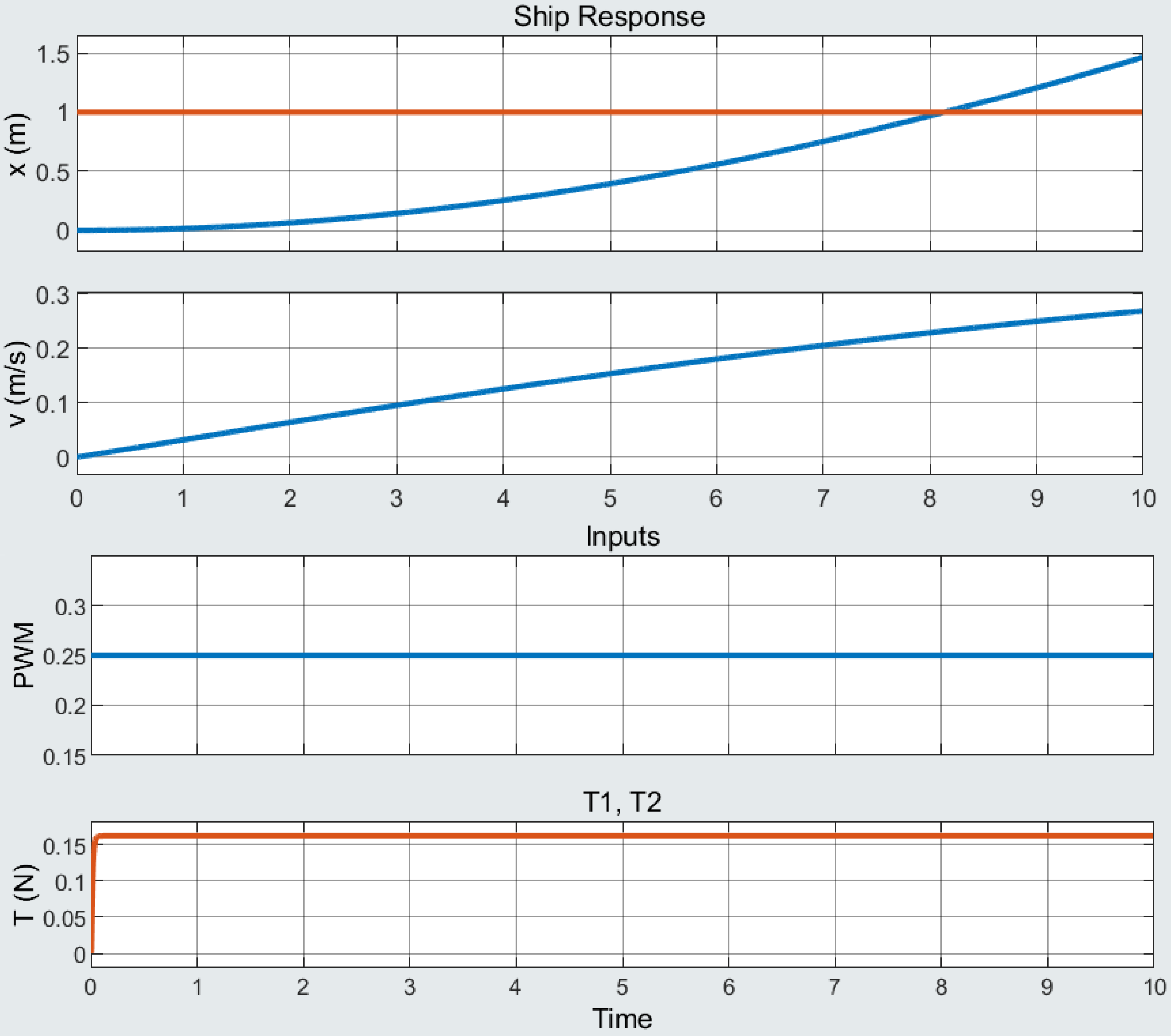


Open-loop (Feedforward) Control

$$PWM = k\,x_{target}$$



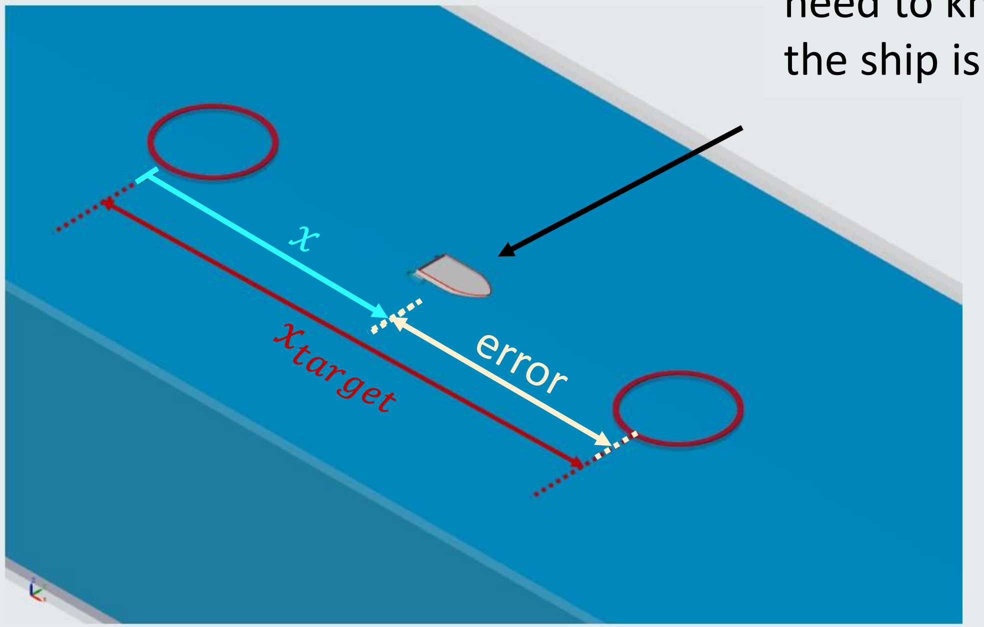
open-loop does not work!



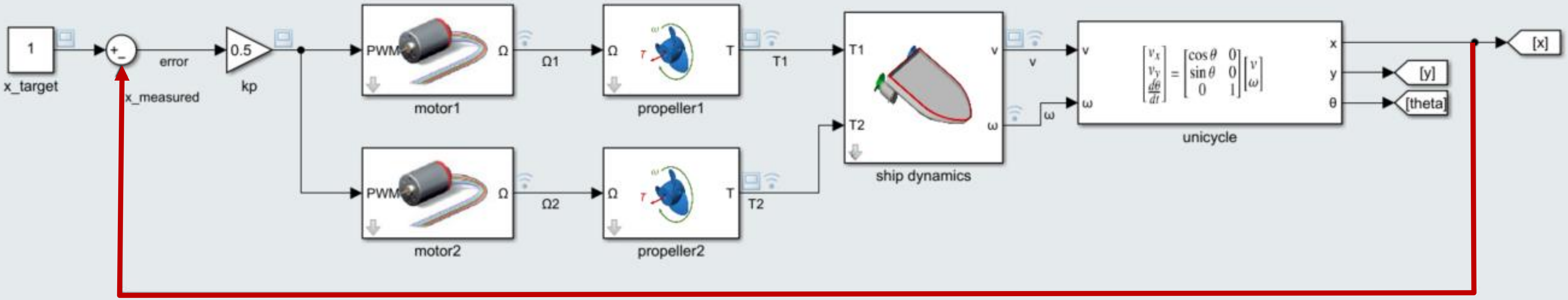
Feedback control

$$error = x_{target} - x$$

$$PWM = k_p\left(x_{target} - x\right)$$



need to know where the ship is to control it

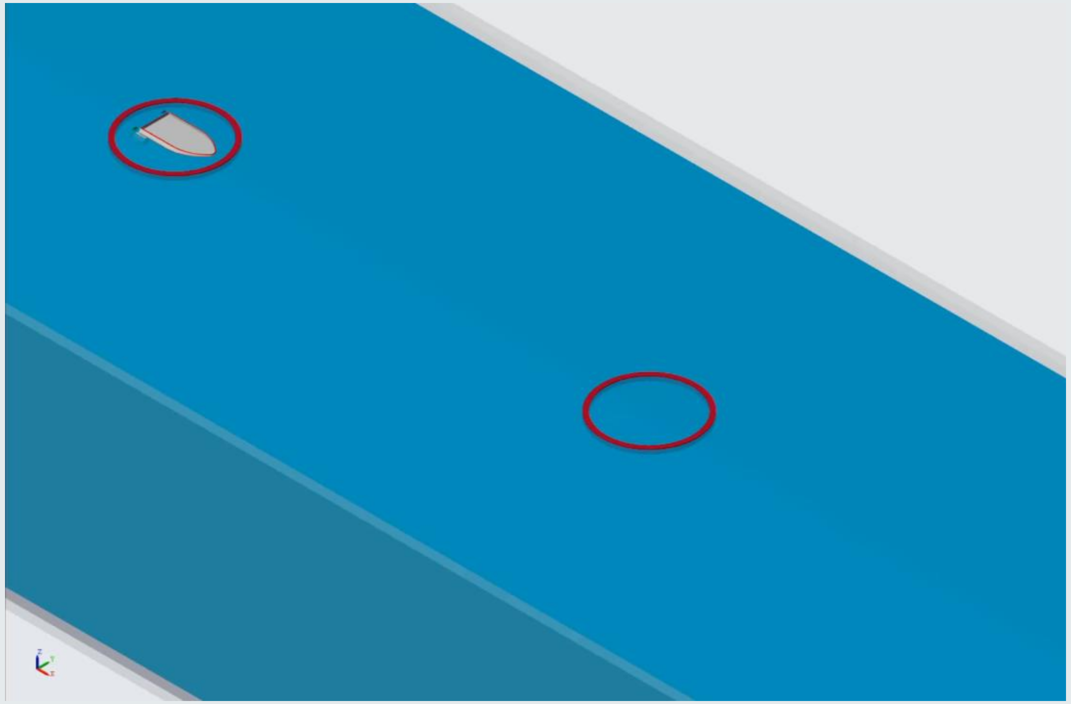


feedback control uses **measurements** of the **plant** state for **control**

Proportional control

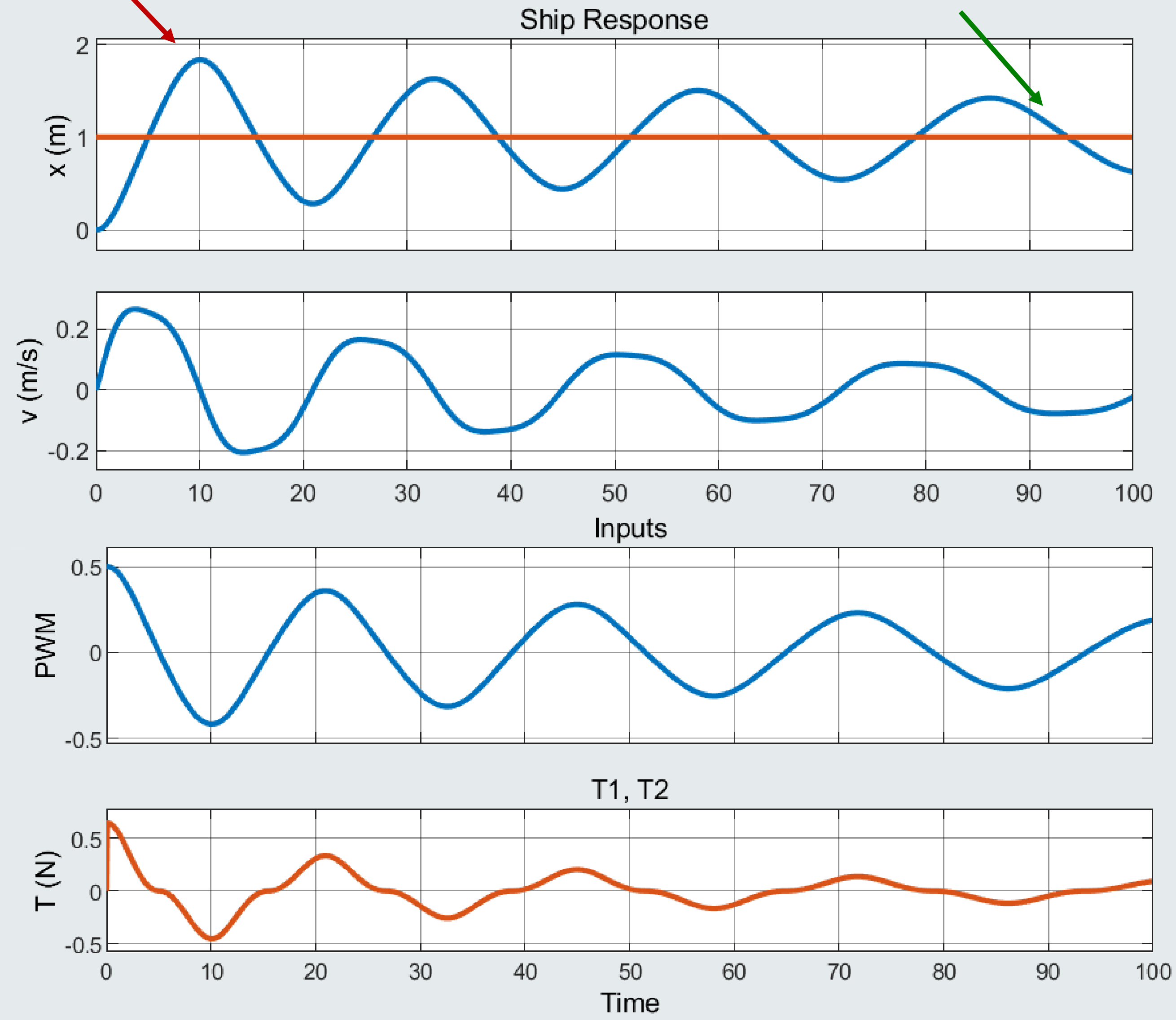
$$error = x_{target} - x$$

$$PWM = k_p\left(x_{target} - x\right)$$



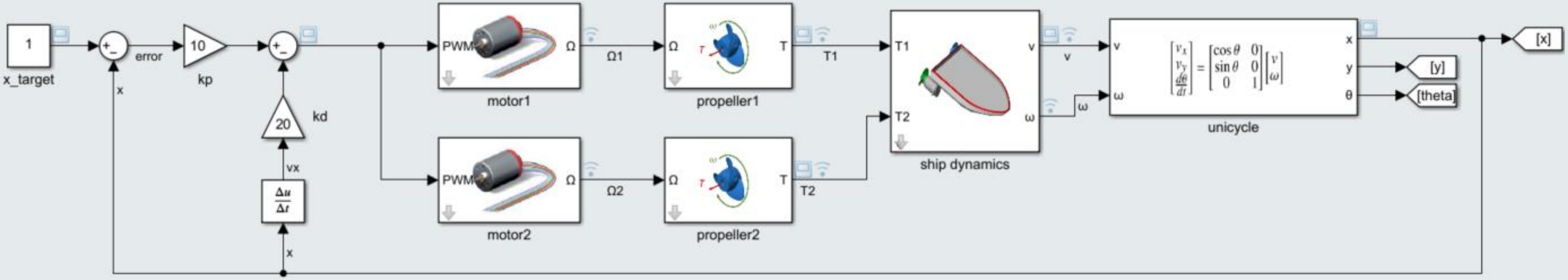
response is underdamped

steady state error approaches zero



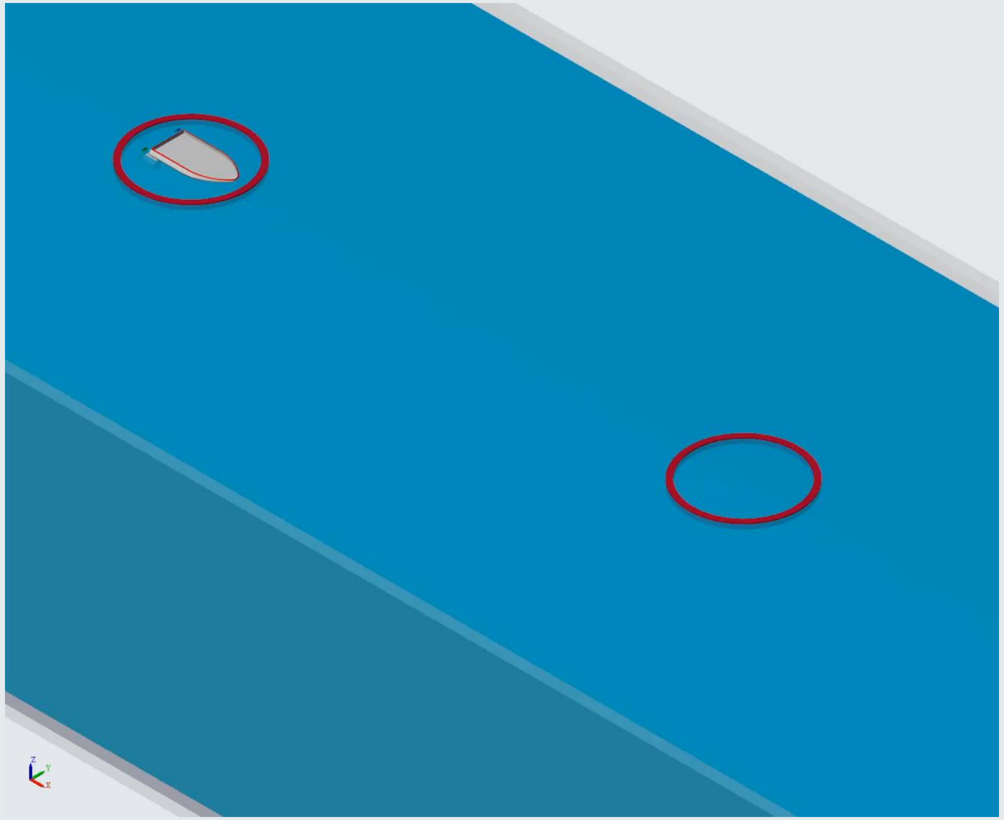
Proportional-derivative control

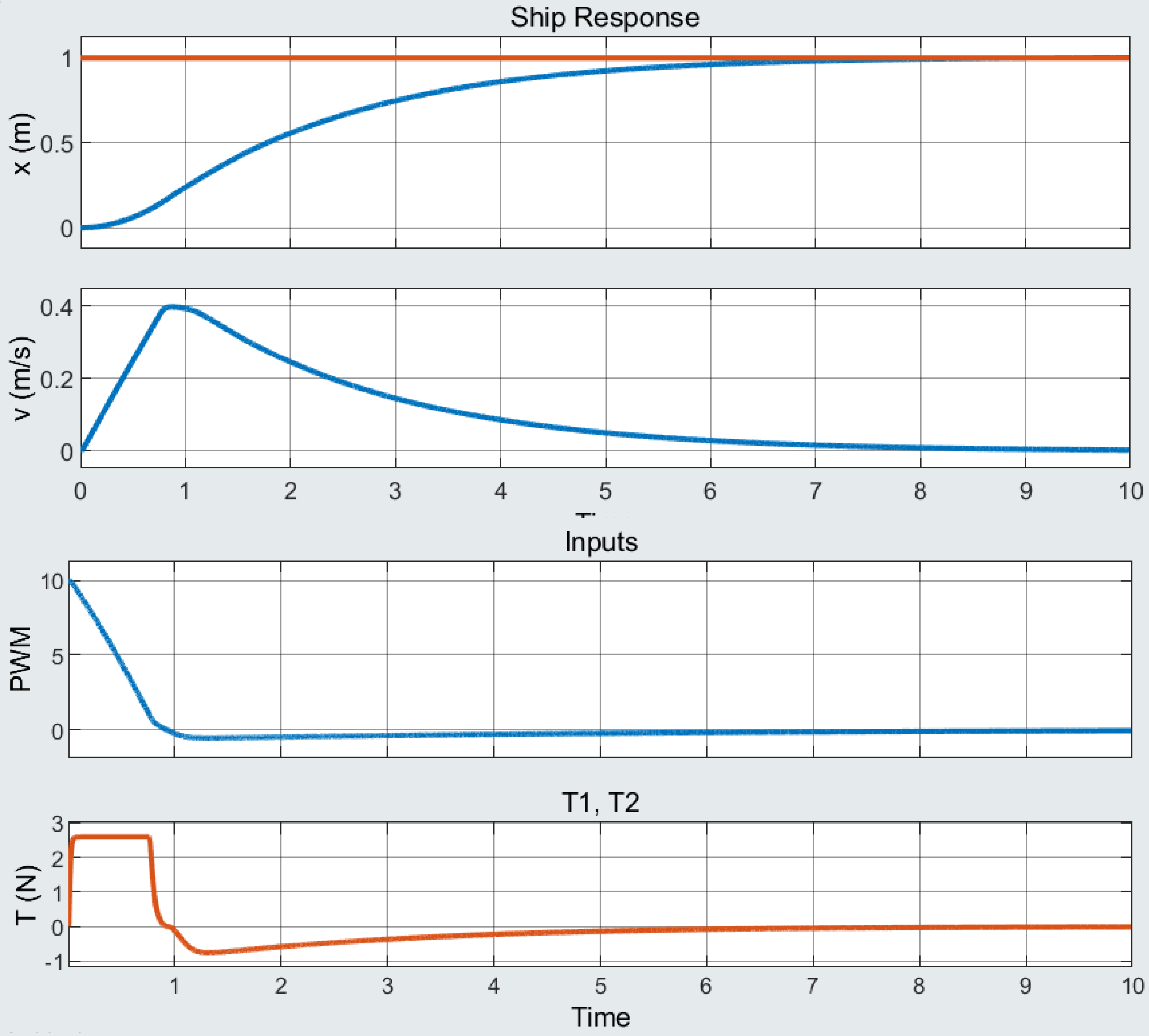
$$PWM = k_p \left(x_{target} - x\right) - k_d \, v$$



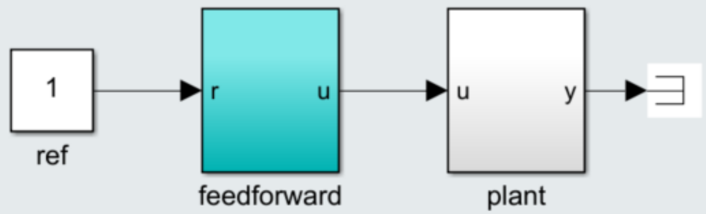
Proportional-derivative control

$$PWM = k_p\,(x_{target} - x) - k_d\,v$$

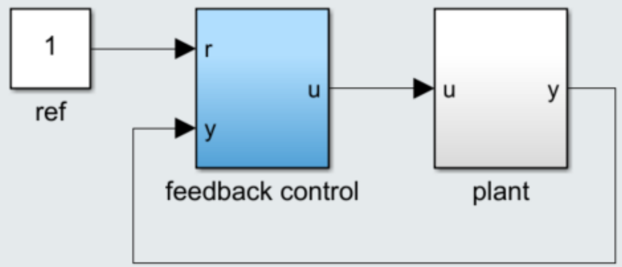




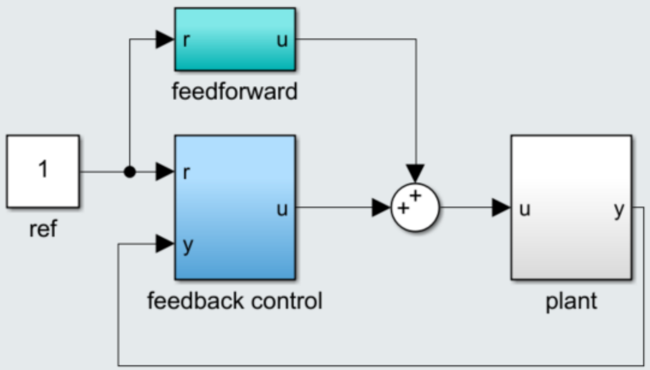
Open vs. Closed-loop Control



Open-loop (feedforward) control
- requires and uses knowledge of the plant
- does not require sensors



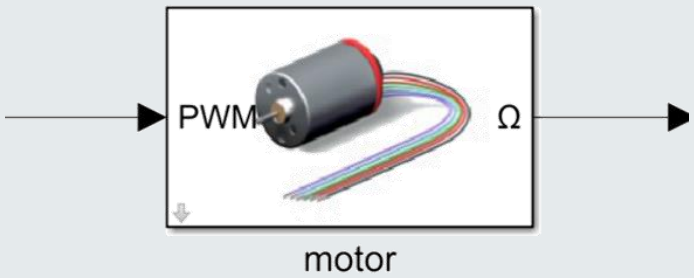
Closed-loop (feedback) control
- requires sensor measurements during operation
- can self-correct and reject disturbances



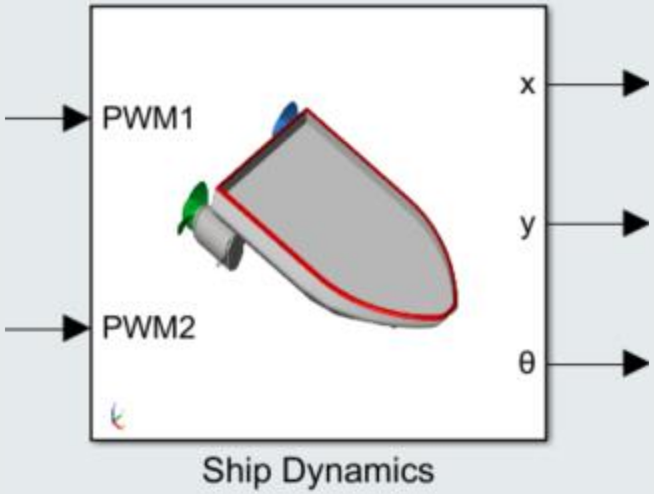
Combined feedback and feedforward architectures can significantly improve the performance of controllers

SISO vs MIMO Systems

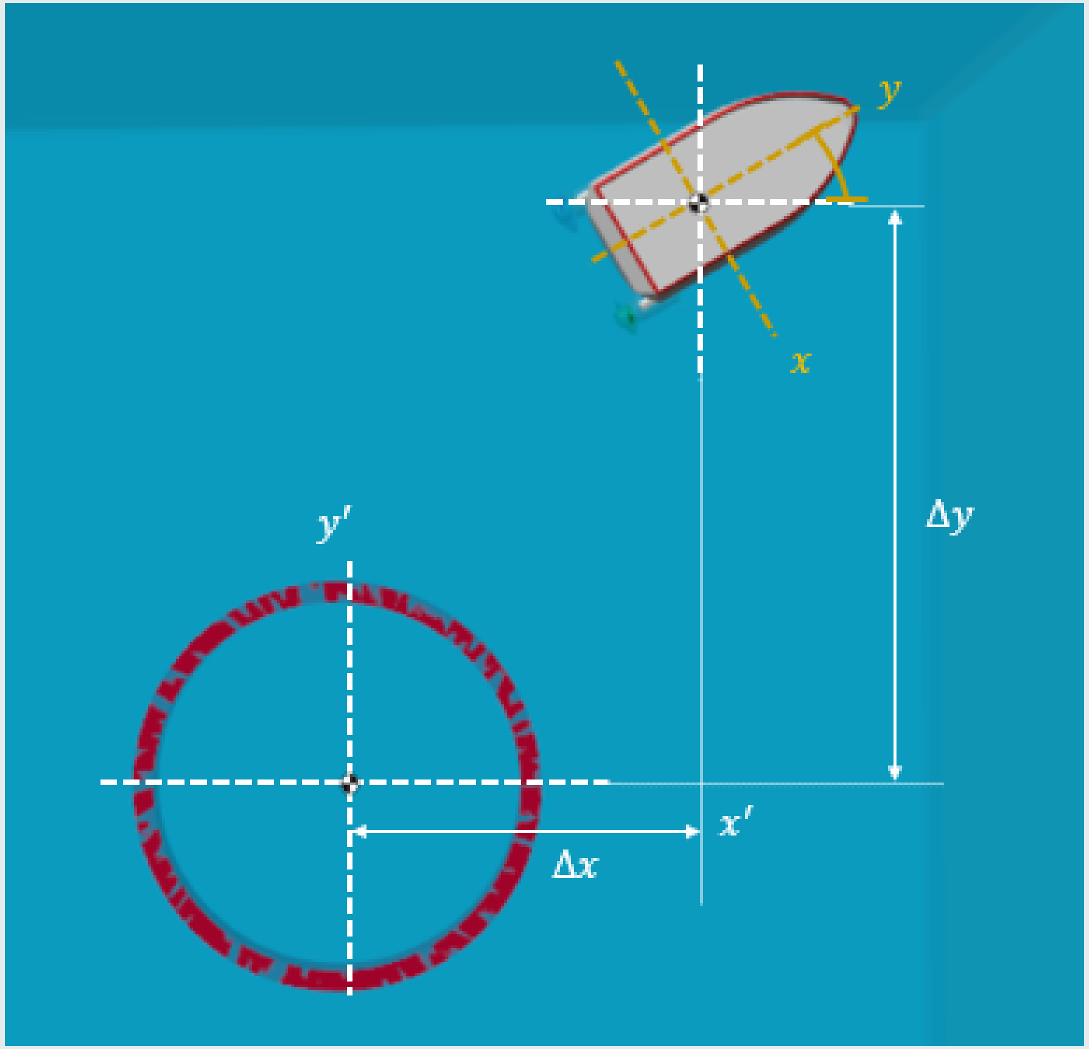
single-input-single-output (SISO) systems



multiple-input-multiple-output (SISO) systems

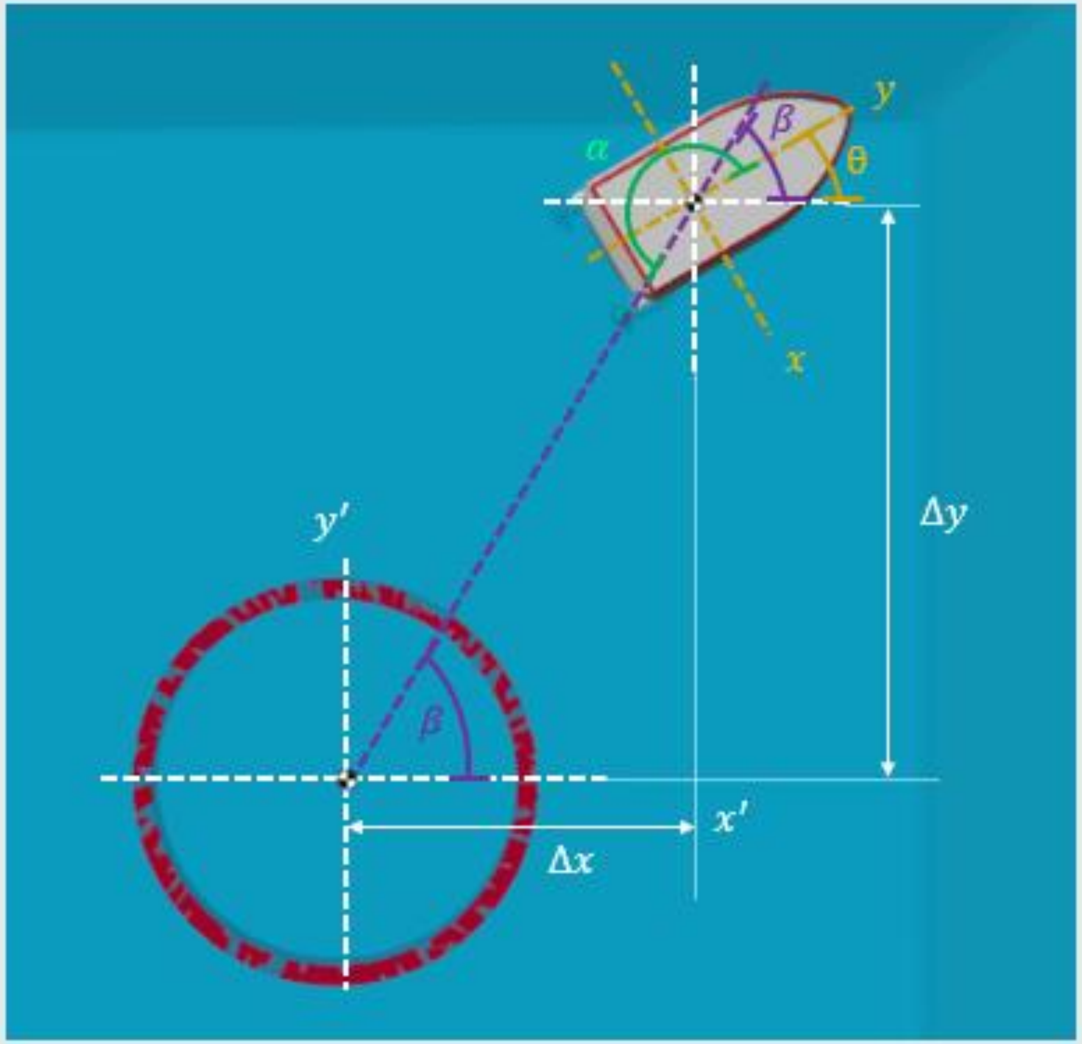


Vehicle Control – Cartesian coordinates



How do we control the ship so that it can move to a target destination (x, y)?

Vehicle Control – Polar coordinates



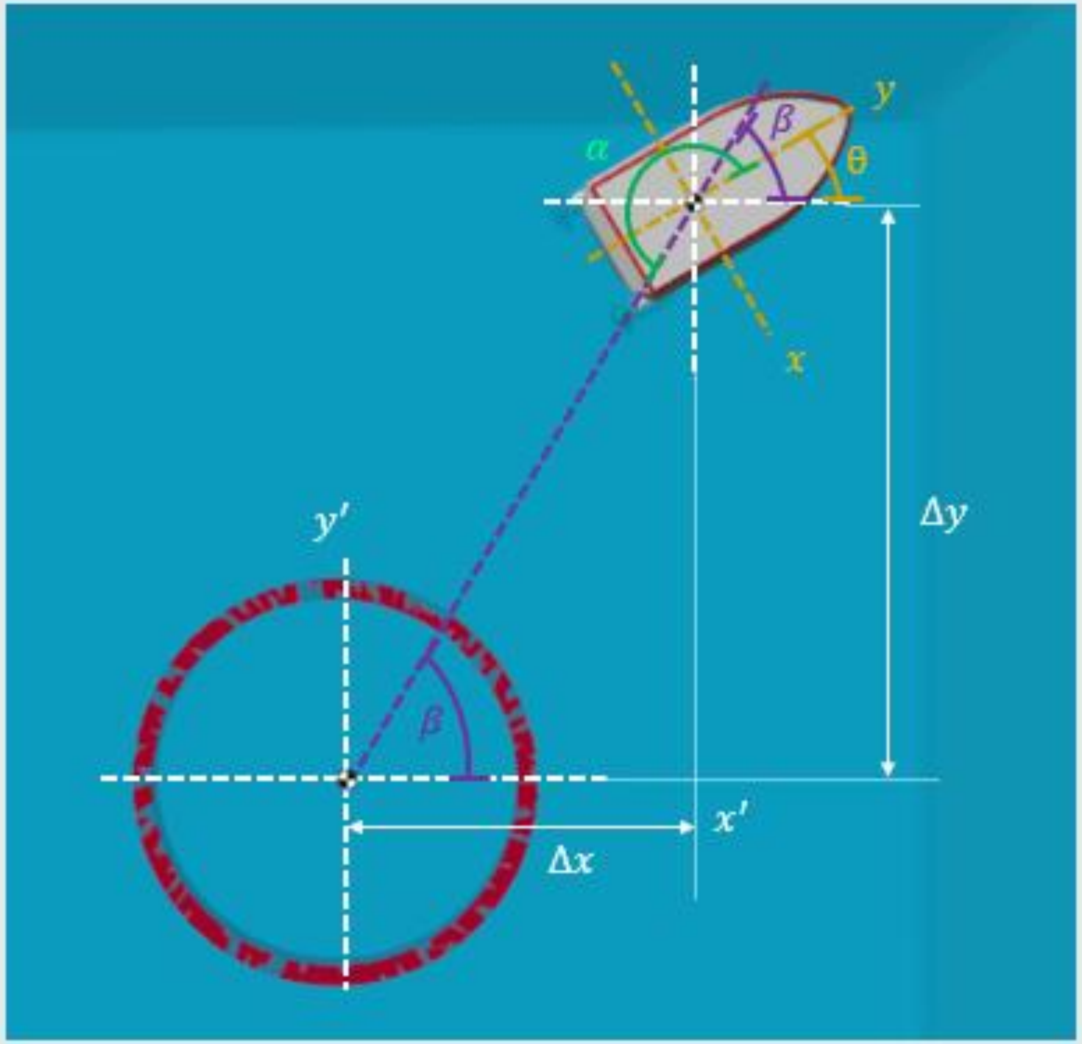
Define polar coordinates relative to goal

$$\rho = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$ (distance)

$$\beta = \operatorname{atan2}\left(\frac{\Delta y}{\Delta x}\right)$$ (angle from horizontal)

$$\alpha = \beta - \theta - \pi$$ (angle from facing goal)

Vehicle Control – MIMO Control law



MIMO kinematic controller

$$v_{in} = k_\rho \, \rho - k_d \, v$$

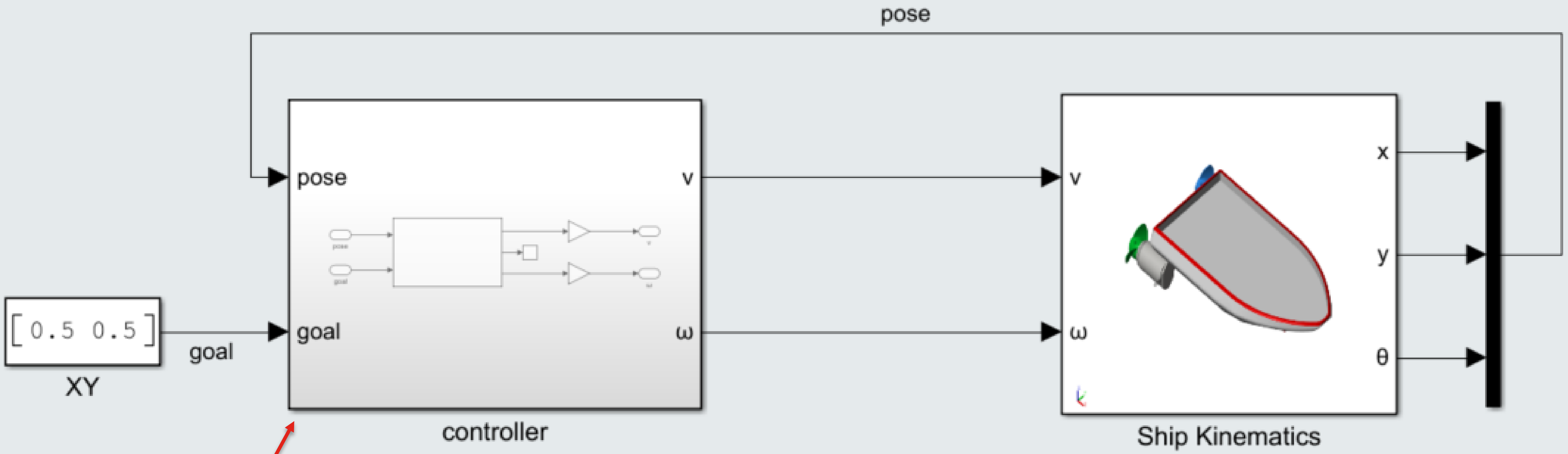
$$\omega_{in} = k_\alpha \, \alpha$$

Take care, there's no guarantee this controller is stable!

As a rule of thumb, ensure $k_\rho < k_\alpha$

Stability proofs are beyond the scope the lecture, derivations can be found in Siegwart (2004) Introduction to Autonomous Mobile Robots

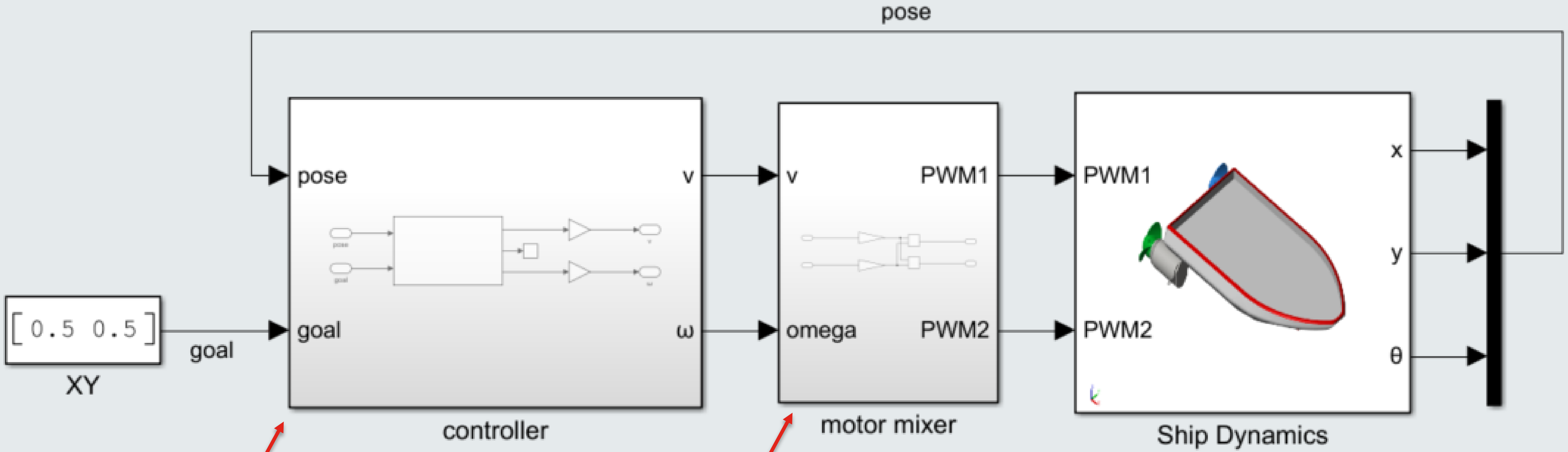
Vehicle Control – MIMO Control law



$$v_{in} = k_\rho \rho - k_d v$$

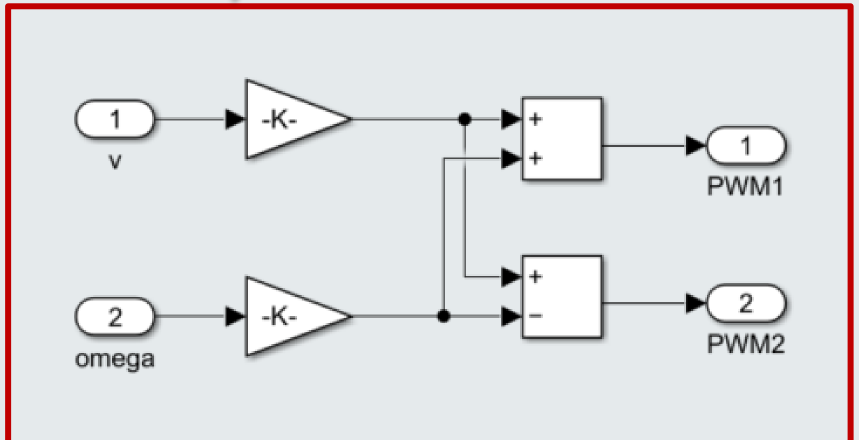
$$\omega_{in} = k_\alpha \alpha$$

Vehicle Control – MIMO Control law

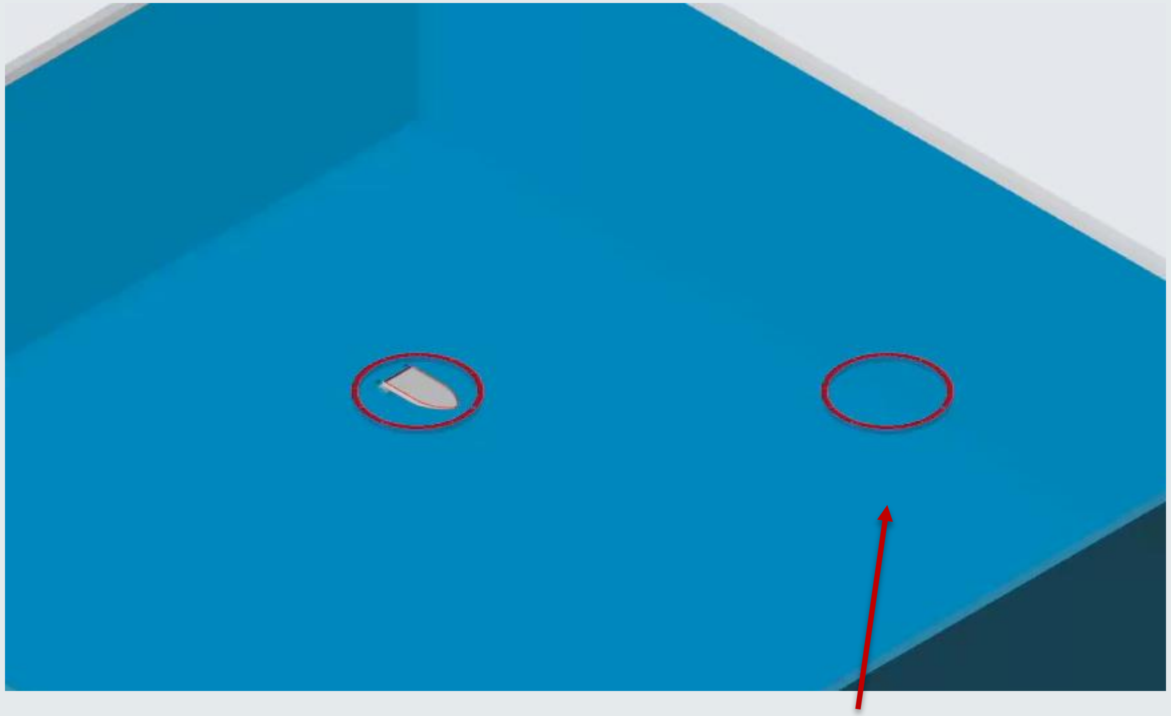


$$v_{in} = k_\rho \rho - k_d v$$

$$\omega_{in} = k_\alpha \alpha$$

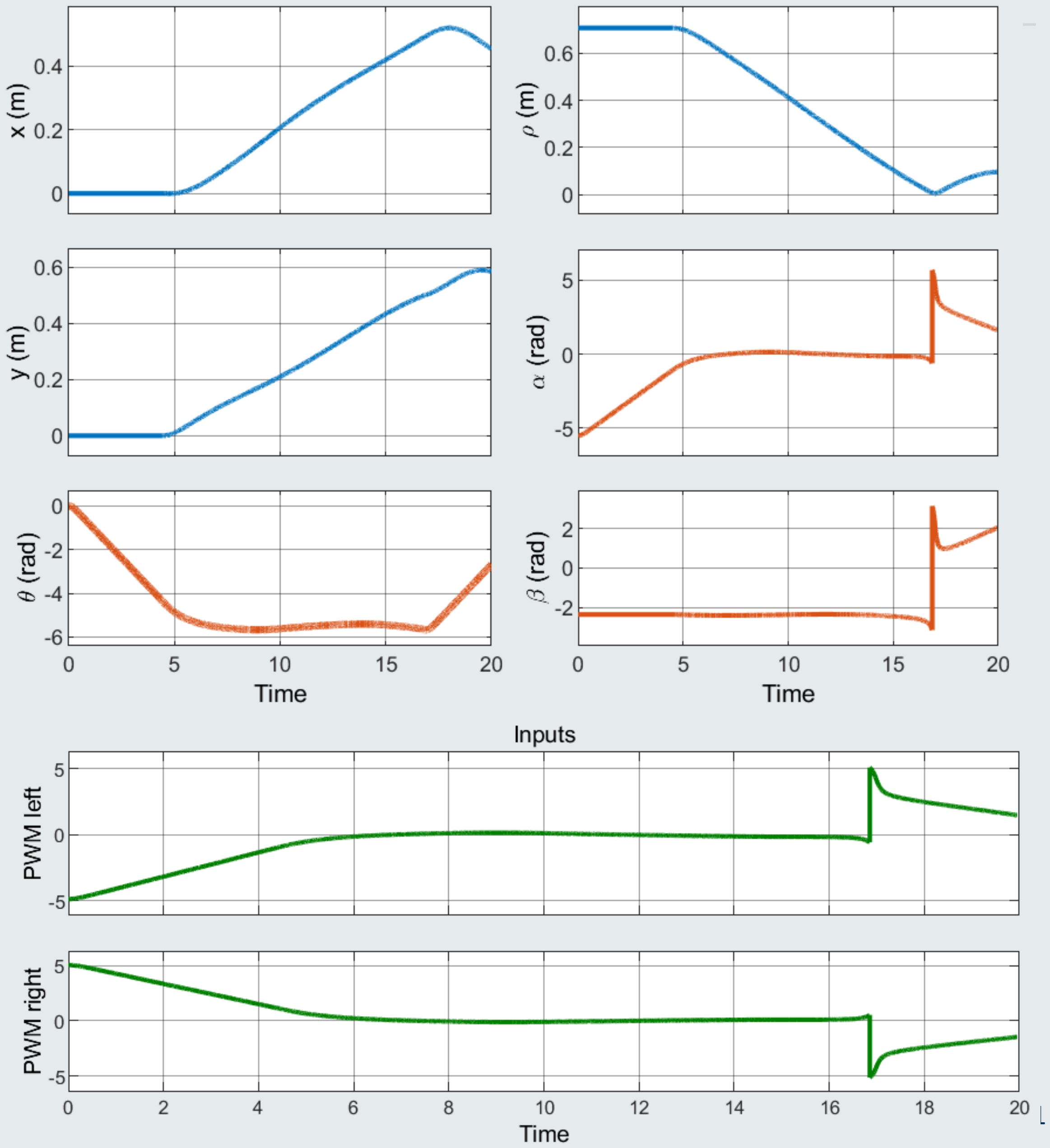


Vehicle Control – MIMO Control law

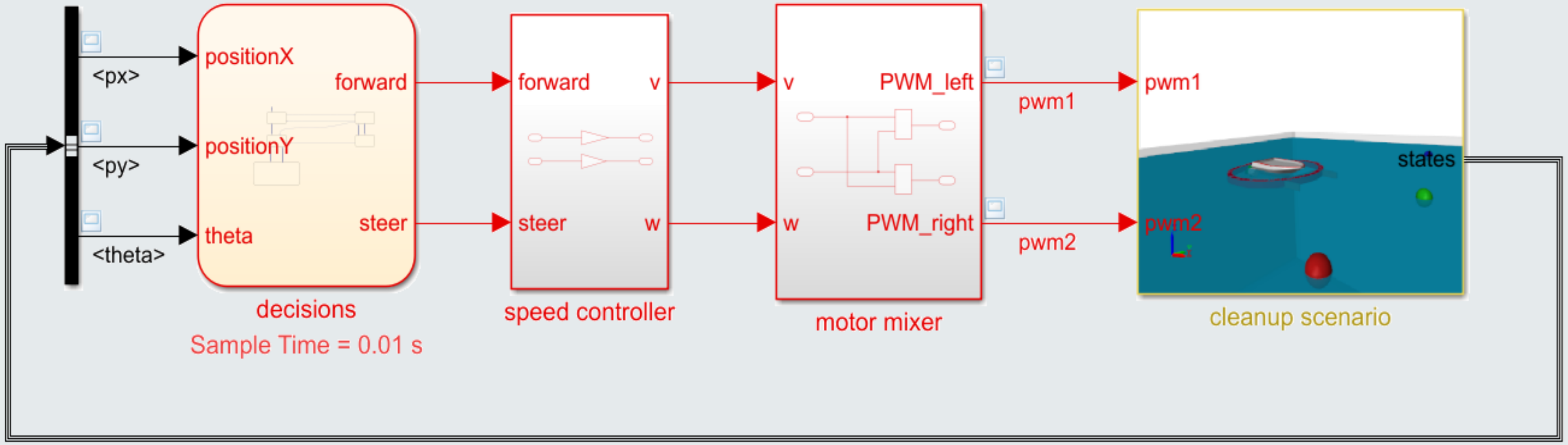


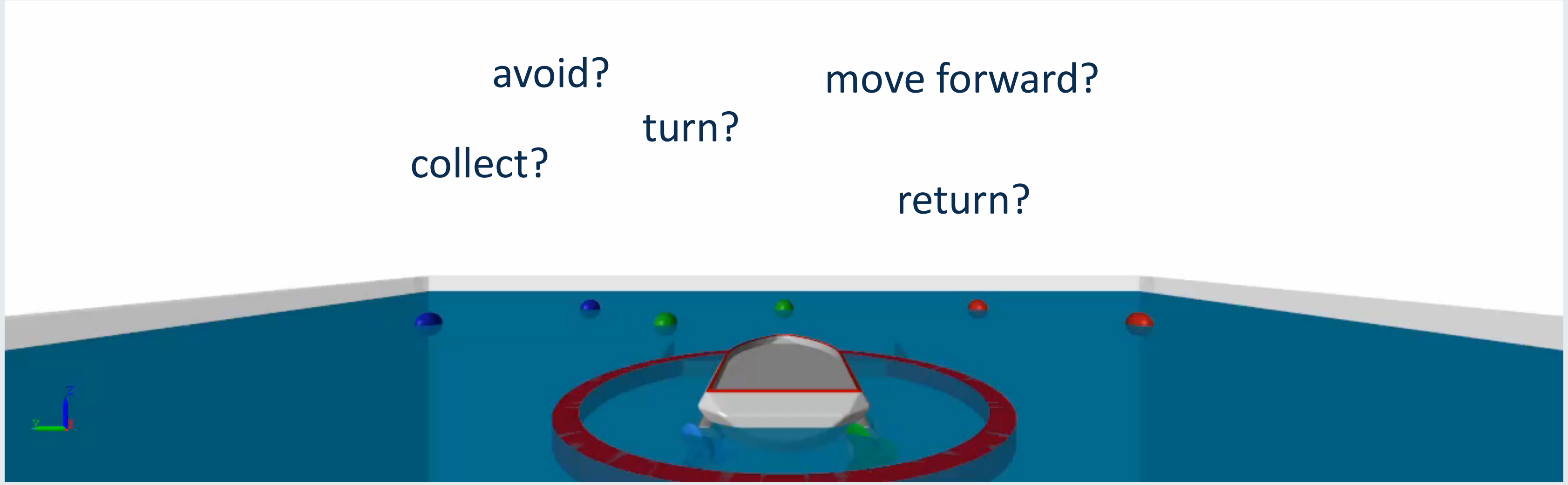
“Goal checking” is typically implemented to shutdown controller when target is reached

Ship Response



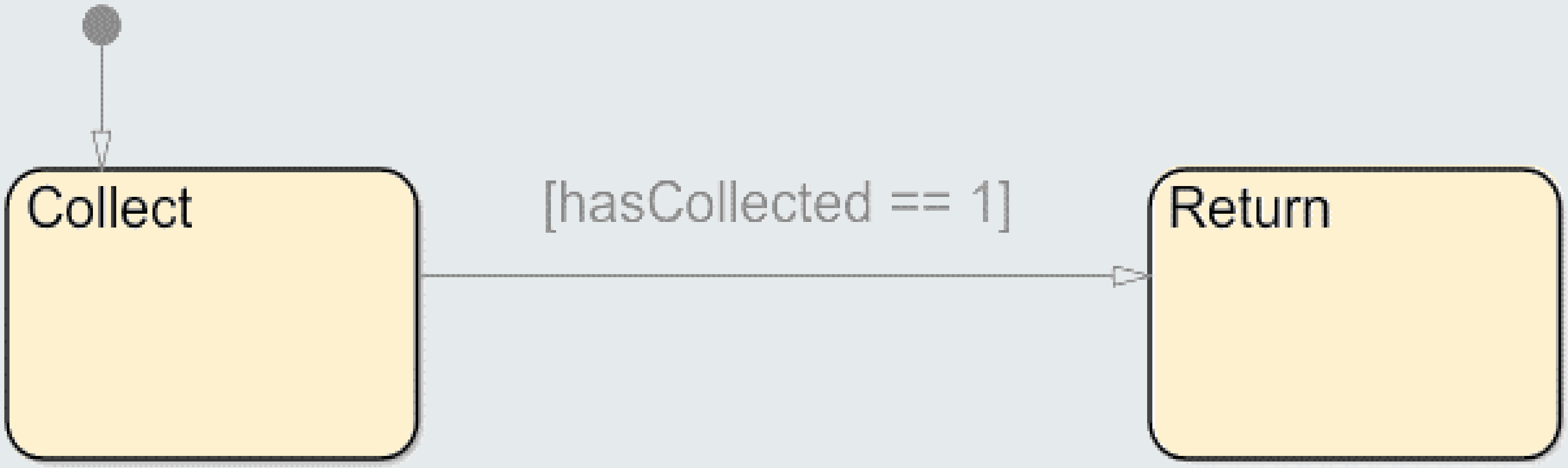
Autonomous Control





Logic-driven control

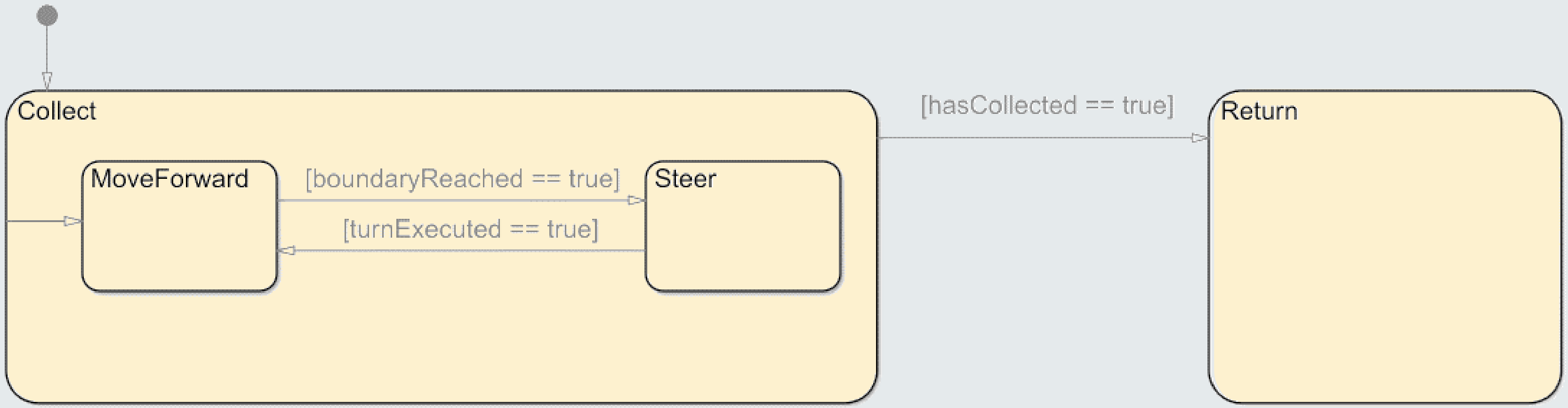
Logic-driven control uses **conditional statements** to switch a machine into different **operating states**



Logic-driven control

Autonomous behaviour is often organised in hierarchies:

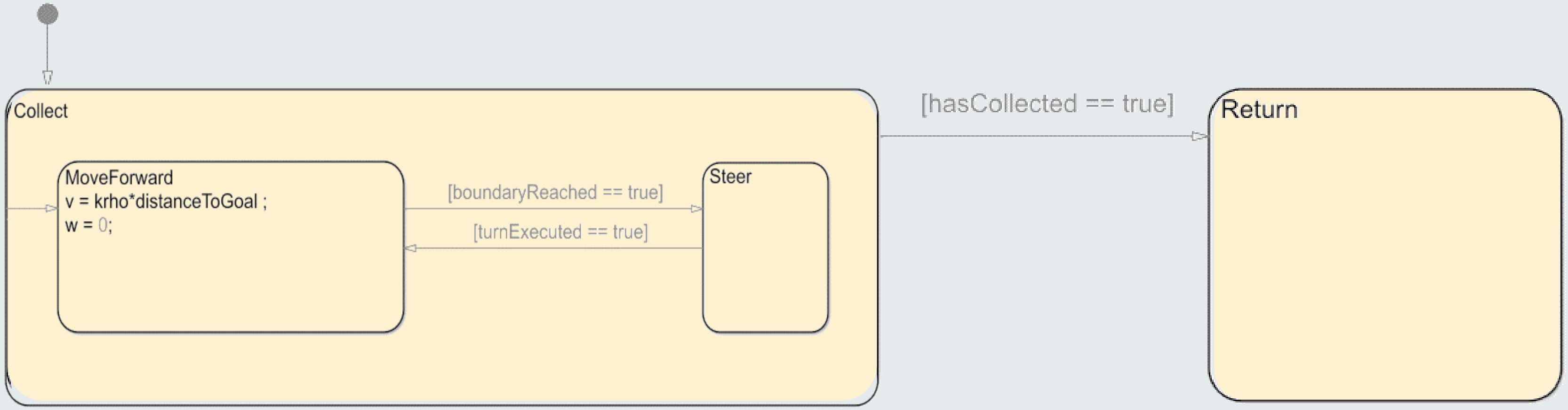
- Task Planning
- Motion Planning
- Motion Following



Logic-driven control

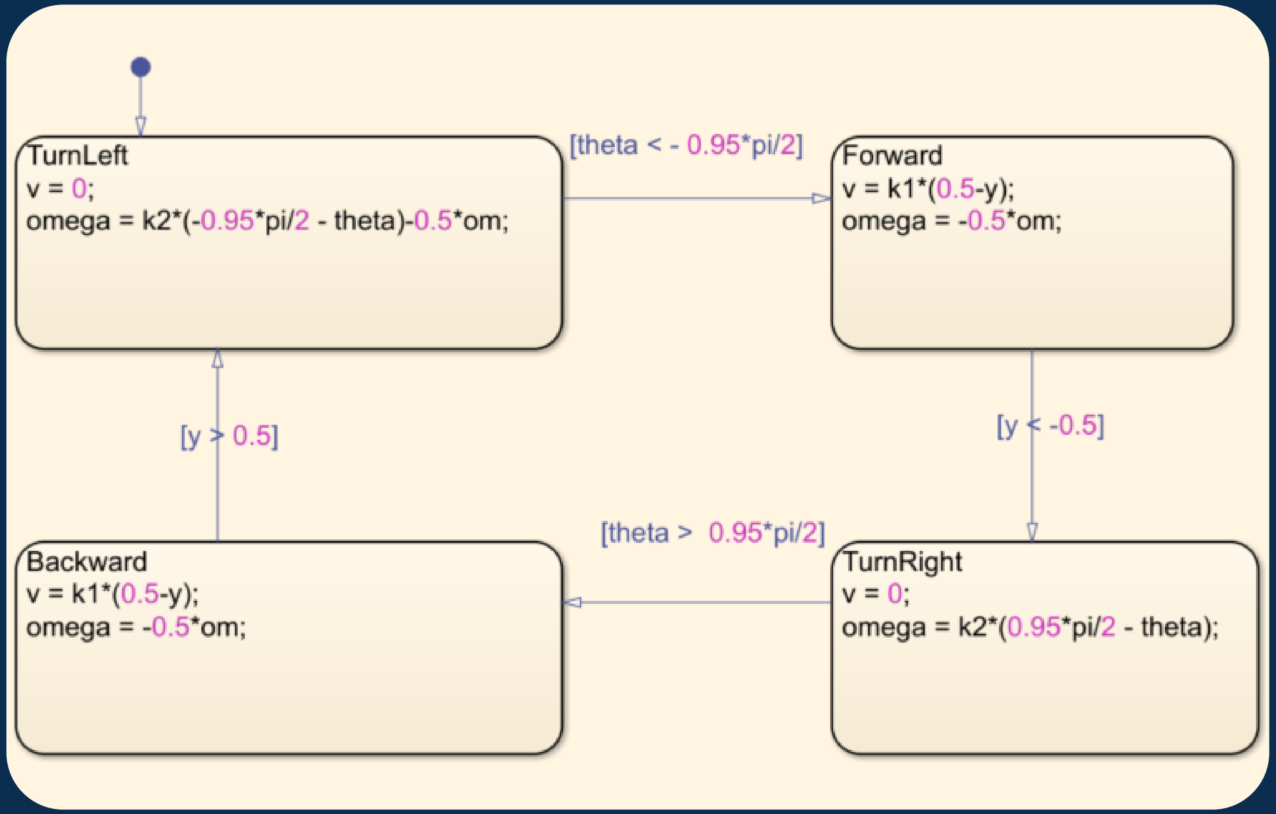
Autonomous behaviour is often organised in hierarchies:

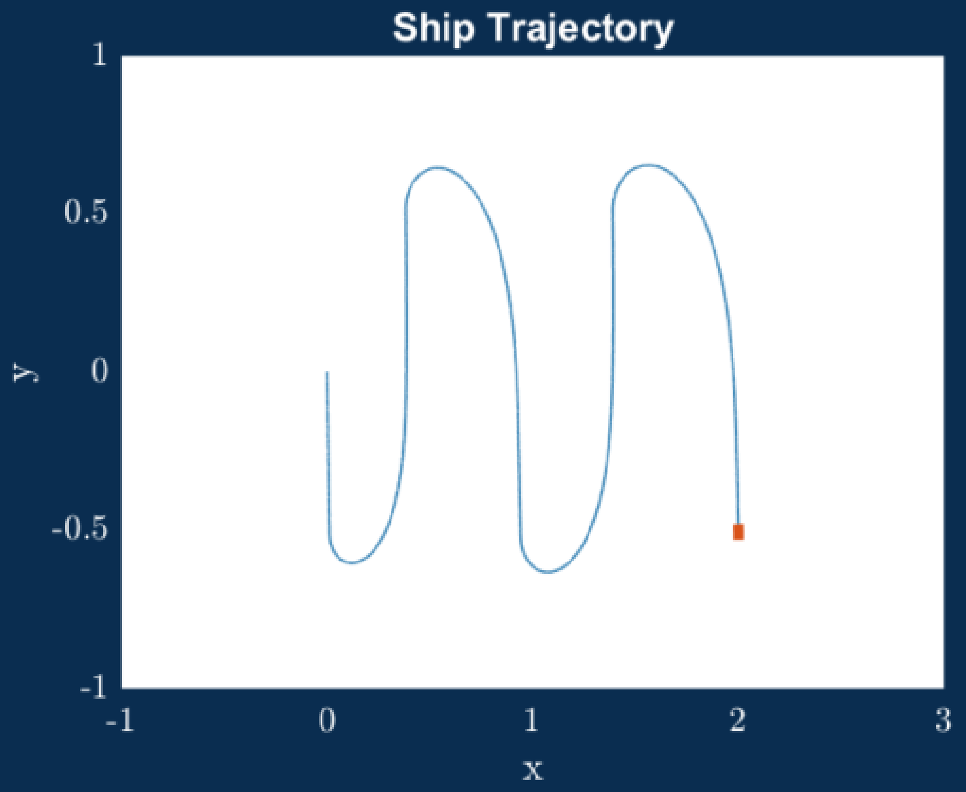
- Task Planning
- Motion Planning
- Motion Following



Logic-driven control

Example: StateFlow demonstration for sweeping actions





Lab Preview

In the third lab, you will use Simulink to control the ship. The control hierarch you will develop can be used to control the ship from the individual coursework exercise.