Hardware and Closed-loop System for Automation

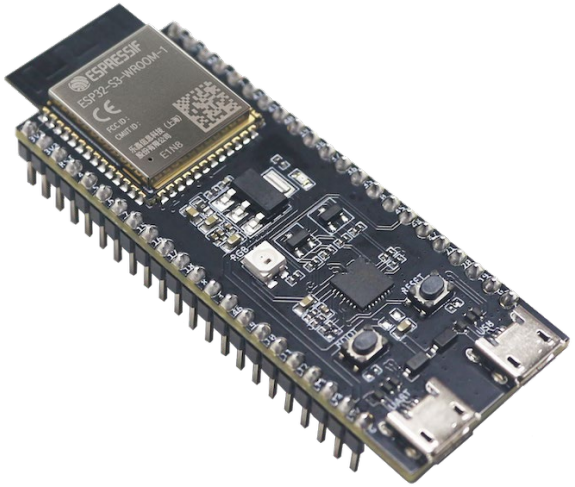
Prof. Jamie Paik
Ekrem Yüksel
Reconfigurable Robotics Laboratory
EPFL, Switzerland

Learning Objectives

- Differentiate between **open-loop** and **closed-loop systems**
- The importance of **closing the loop** for automation and sustainability
- **Identify** the key components of a closed-loop system: reference, controller, actuator, plant, sensor, and feedback path.
- Describe the roles of the proportional, integral, and derivative terms in a PID controller
- **Understand** the impact of noise on sensor signals and why filtering is critical for reliable closed-loop performance.
- Role of filtering and how to implement it
- **Analyze** how feedback improves energy efficiency, reliability, and adaptability in sustainable automation systems
- **Recognize** real-world examples of closed-loop systems across domains like energy, mobility, and biology.

Introduction to Hardware

- **Hardware list**
  - **Already Provided**
    - Microcontroller: ESP32
      - Support Arduino ecosystem;
      - Bluetooth + WIFI communication;
      - Duo-core CPU;
    - Servo: 5V DC motor servo x 2
      - Read position;
      - Move using position/velocity commands;
    - Power supply
      - 5V USB power supply
      - Battery optional
  - **To be selected**
    - Sensors: at least 2 kind



Open Loop vs Closed Loop

- **Open loop system:** A system whose output/behavior is **not regulated**. It is the cheapest type of control.

- **Examples:**
  - Running water irrigation system every day for 10 minutes
  - Running a conveyor belt at a constant speed
  - Heating the building/apartment
  - Simple torque control

Open Loop vs Closed Loop

- **Closed loop System:** A system whose output/behavior **is regulated** by **measuring**, and **processing** its **current state** to maintain a **desired state**.

- **Examples:**
  - Soil moisture sensor controls the water irrigation timing
  - Heating is kept between certain thresholds

Why Closing the Loop?

Aspect	Open-Loop (OL)	Closed-Loop (CL)
Definition	No feedback; output is not measured or corrected	Feedback present; output is measured and used to adjust input
Complexity	Simple design, fewer components	More complex: requires sensors, feedback processing, tuning
Cost	Cheaper (no sensors or feedback hardware)	More expensive (sensors, ADCs, controllers needed)
Setup/Implementation	Easy to implement	Requires tuning and calibration
Accuracy	Low; performance depends on exact system modeling	High; adapts to changes and maintains target values
Robustness	Poor at handling disturbances or system changes	Corrects for load changes, wear, environmental shifts
Stability	Always stable (if system itself is stable)	Can become unstable if feedback loop is poorly tuned
Responsiveness	No dynamic correction capability	Continuously adjusts to track desired behavior
Energy Efficiency	May waste energy due to over/undercompensation	Can be optimized to use minimal energy for task
Use Cases	Simple timing circuits, basic heating systems, open-loop pumps	Precision motion control, robotics, industrial automation
Sustainability Impact	Less efficient, prone to wear/waste	More sustainable via precision, reduced rework, and longevity

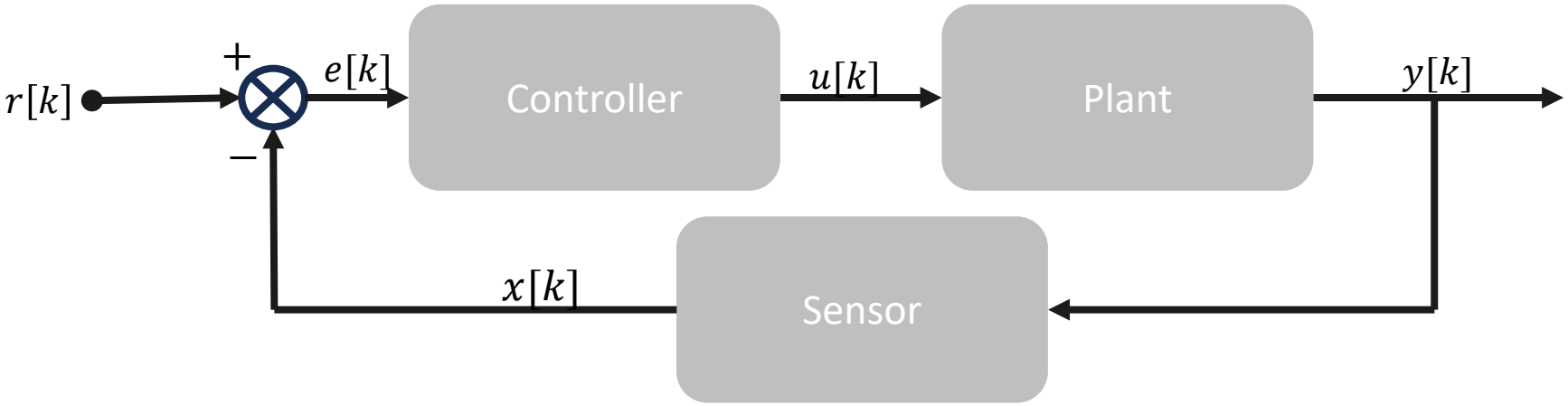
Why Closing the Loop?

- Automatic and Sustainable
  - **Reducing human intervention**: accomplishing pre-set task automatically;
  - **Efficiency optimization**: advance control methods permit optimizations on energy usage and working conditions;
- Robust and Adaptable
  - **Disturbance Rejection**;
  - **Environment Adaptation**: automatically adjust to adapt environment changes and internal system malfunctions;
- Accurate and Rapid Response
  - **Reducing errors:** estimate error in real time and regulate;
- Safe
  - **Fail-Safe Operations**: force, speed limits;

Closed loop System

Essential hardware:

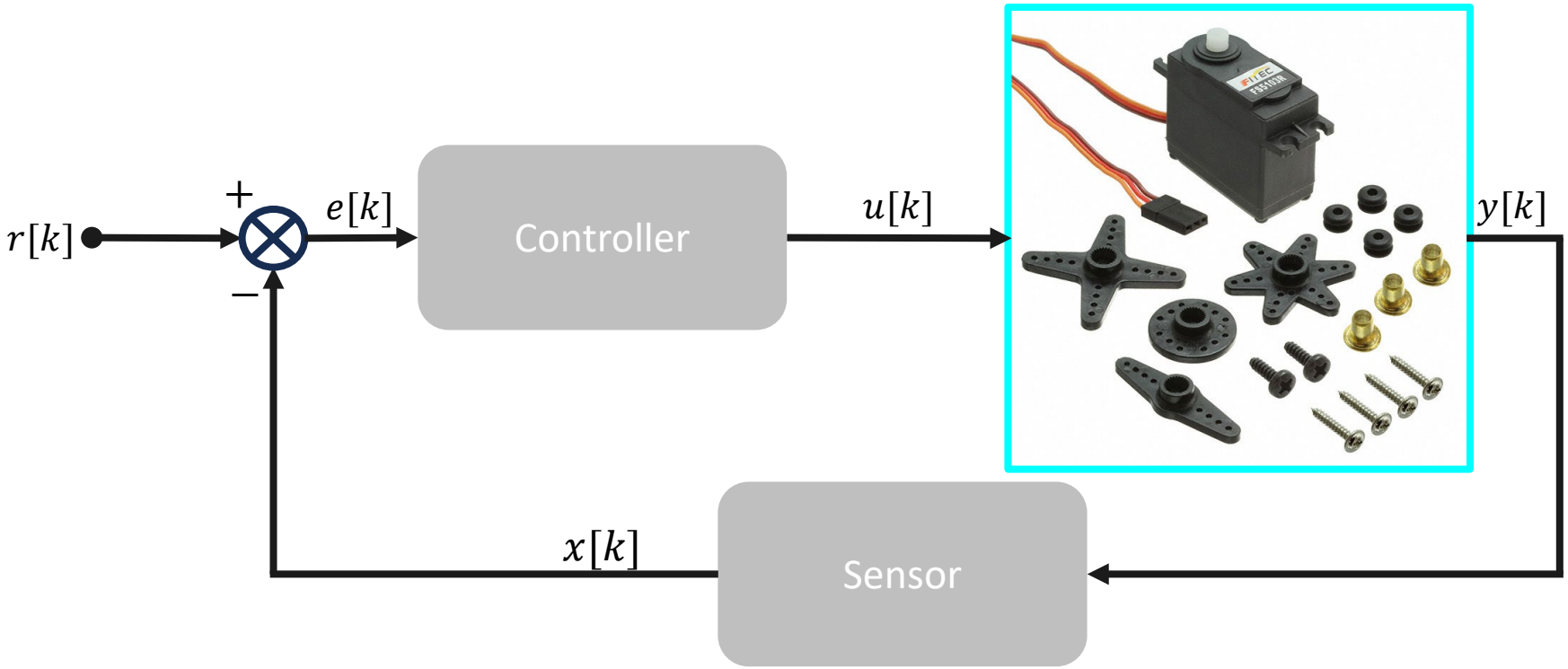
- **Sensors**: reading states
- **Controllers**: computing and processing
- **Actuators**: taking command from the controller, regulating states



Closed loop System

Essential hardware:

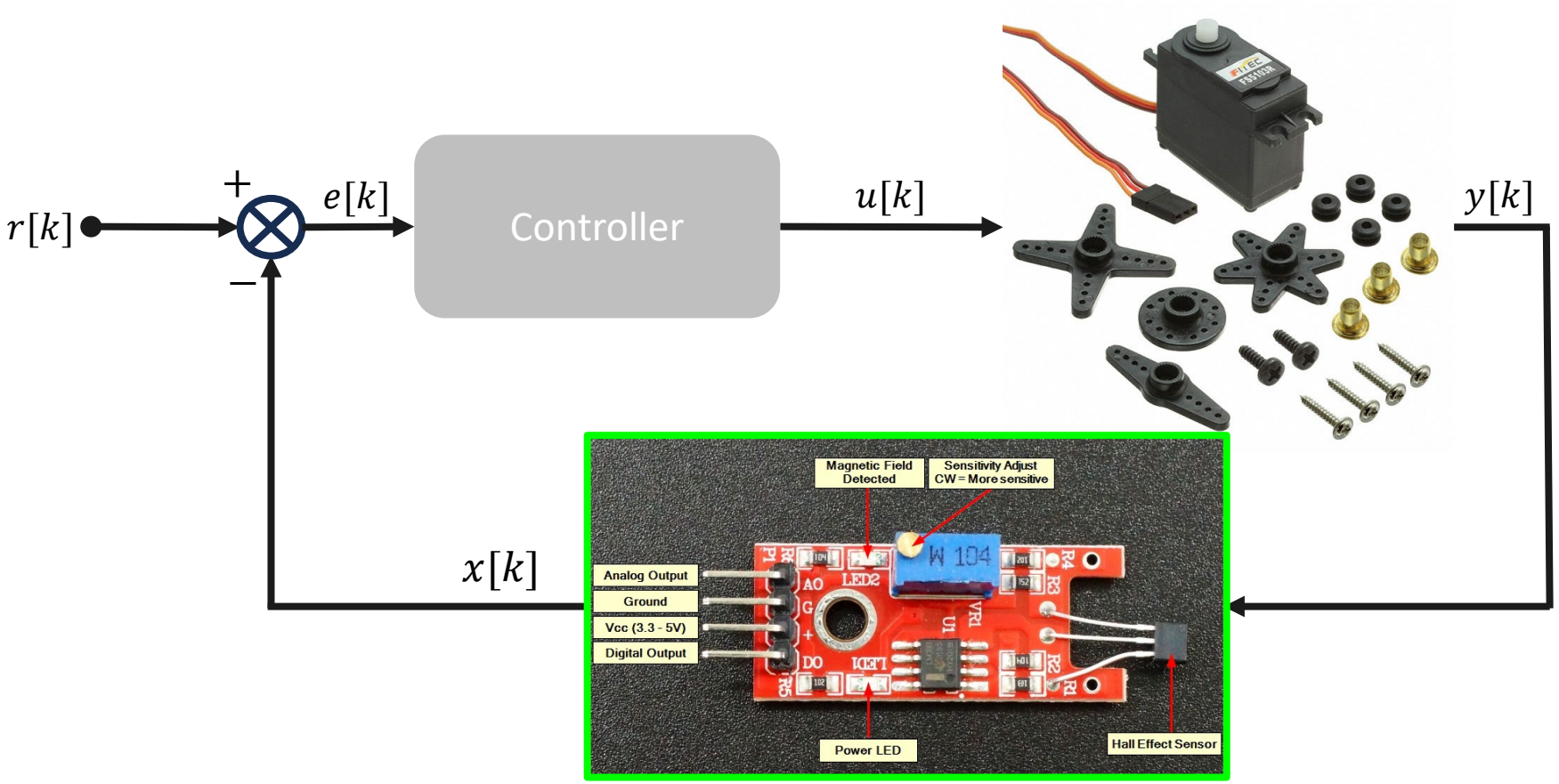
- **Sensors**: reading states
- **Controllers**: computing and processing
- **Actuators**: taking command from the controller, regulating states



Closed loop System

Essential hardware:

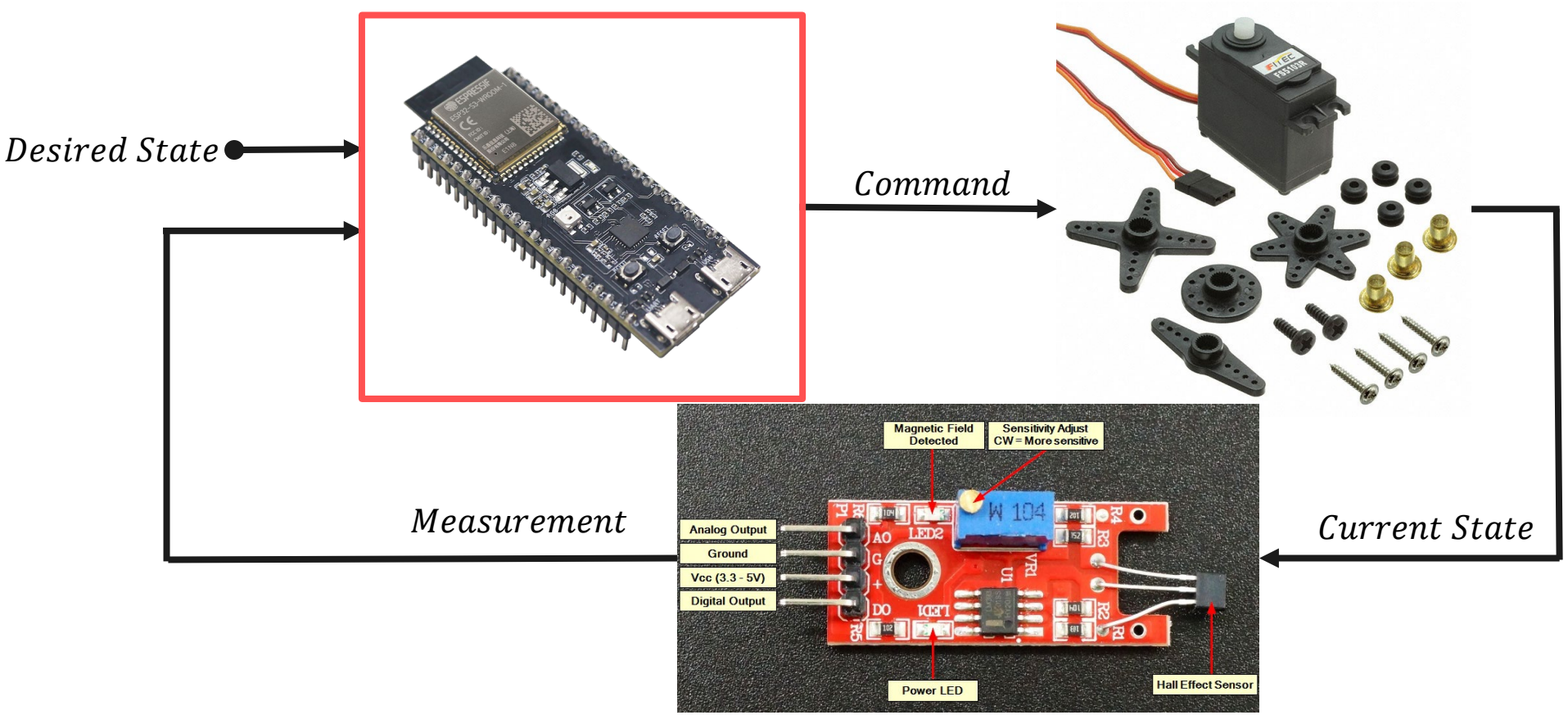
- **Sensors**: reading states
- **Controllers**: computing and processing
- **Actuators**: taking command from the controller, regulating states



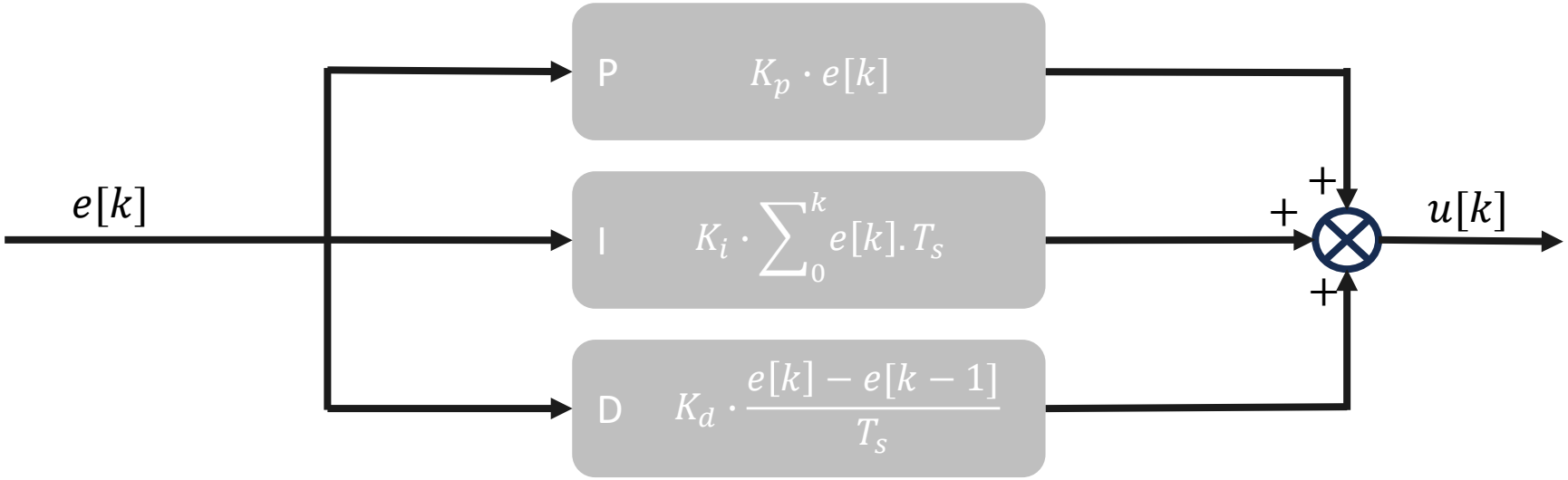
Closed loop System

Essential hardware:

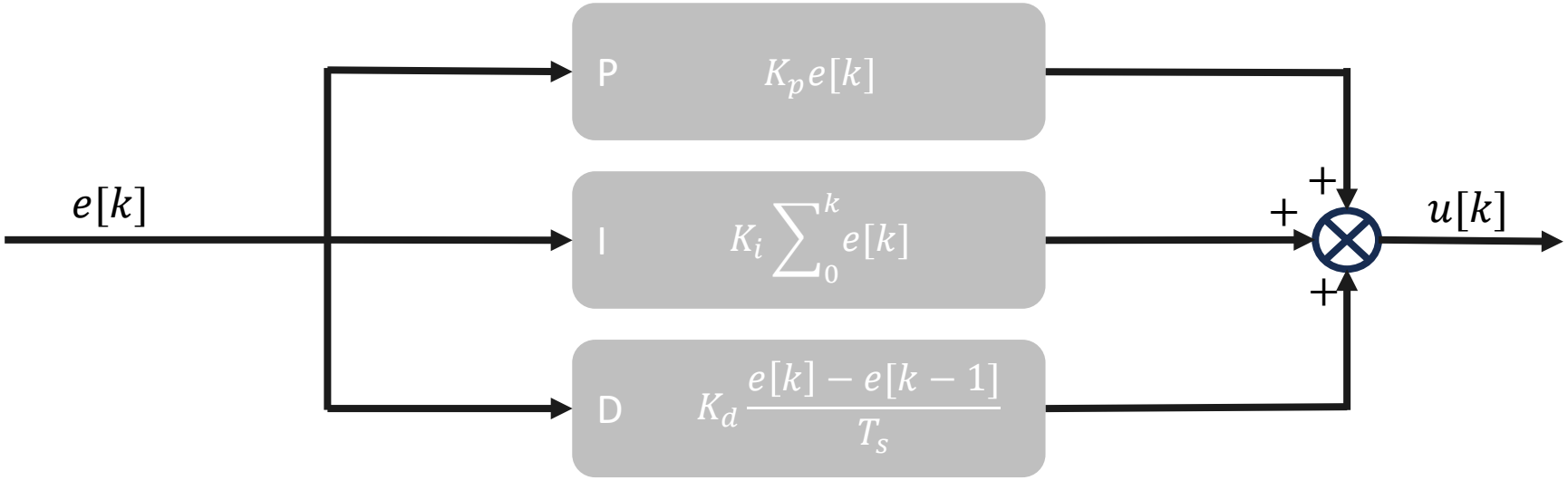
- **Sensors**: reading states
- **Controllers**: computing and processing
- **Actuators**: taking command from the controller, regulating states



Simple PID Implementation



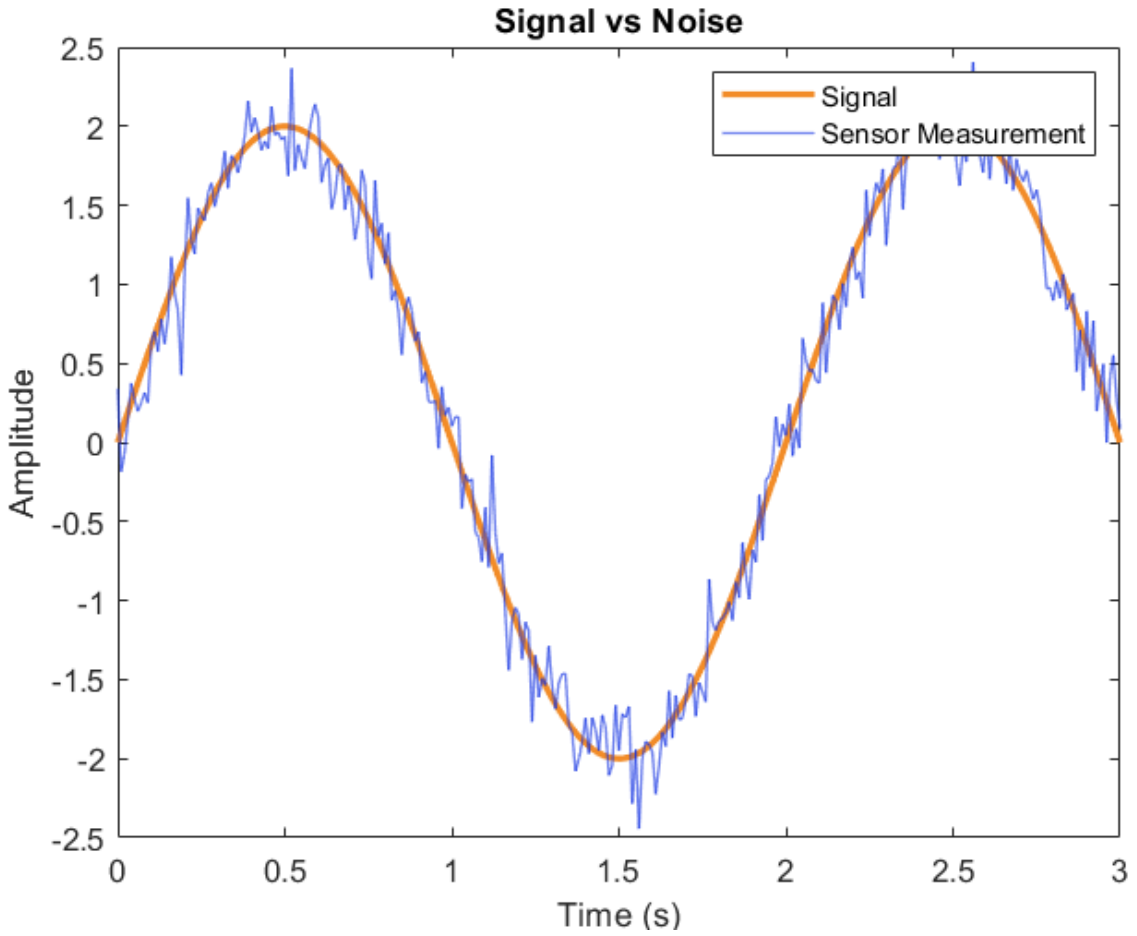
Simple PID Implementation




```python
previous_error = 0.0
error_sum = 0.0
while True:
    measurement = read_measurements()
    error = desired - measurement
    error_sum += error * Ts
    control_input = Kp * error + Ki * error_sum + Kd * (error - previous_error) / Ts
    send_control_signal(control_input)
    previous_error = error
```

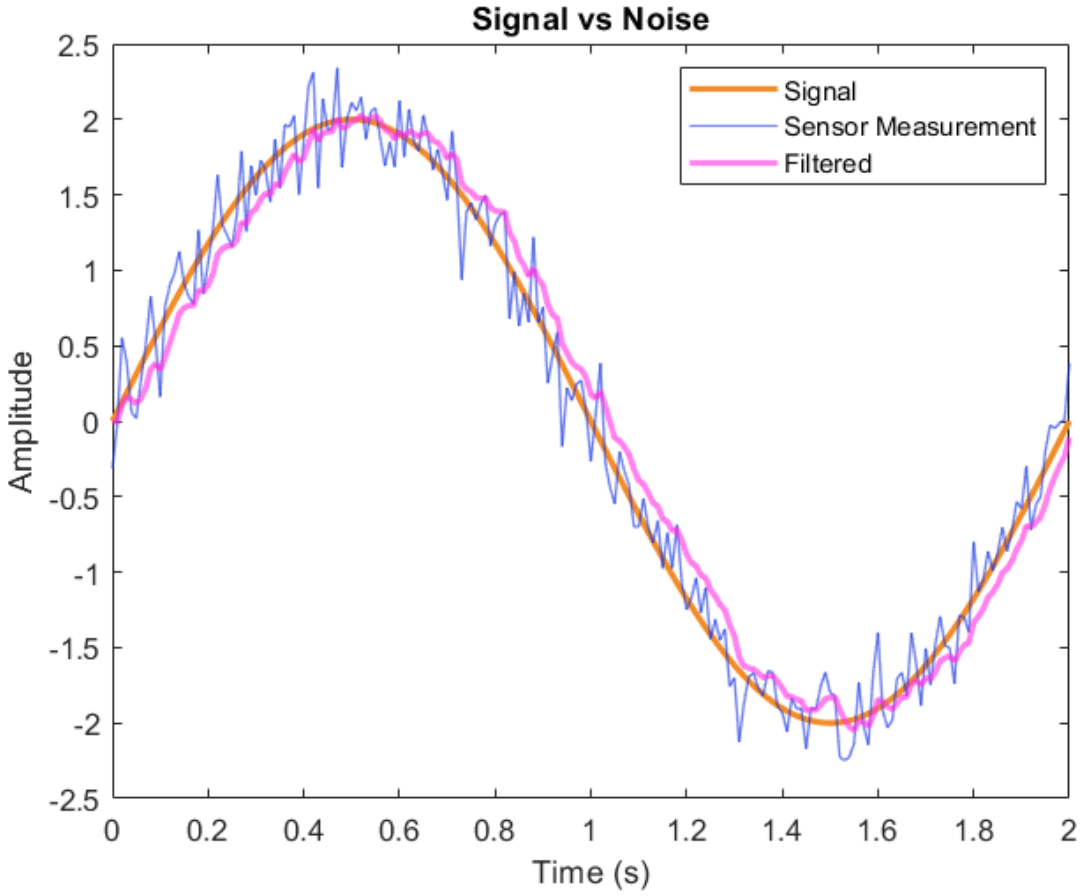
Filtering

- Real-World sensor signals are noisy
  - Electrical interference
  - Quantization errors (Analog-to-digital)
  - Mechanical vibrations or backlash
- Especially Derivative term in PID amplifies the noise
  - Can drive system to instability
  - Can cause jittery motion

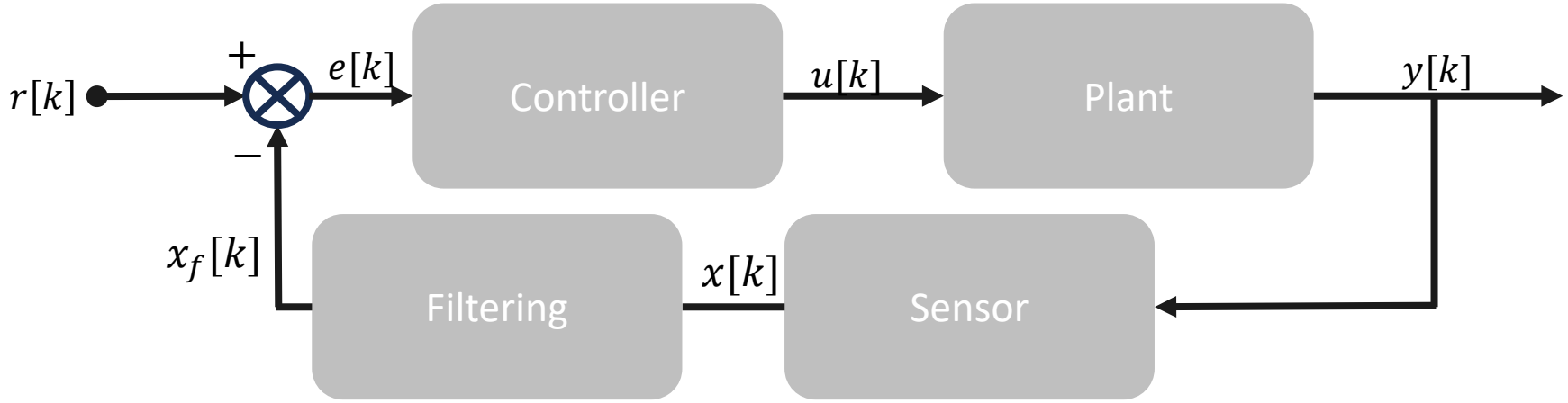


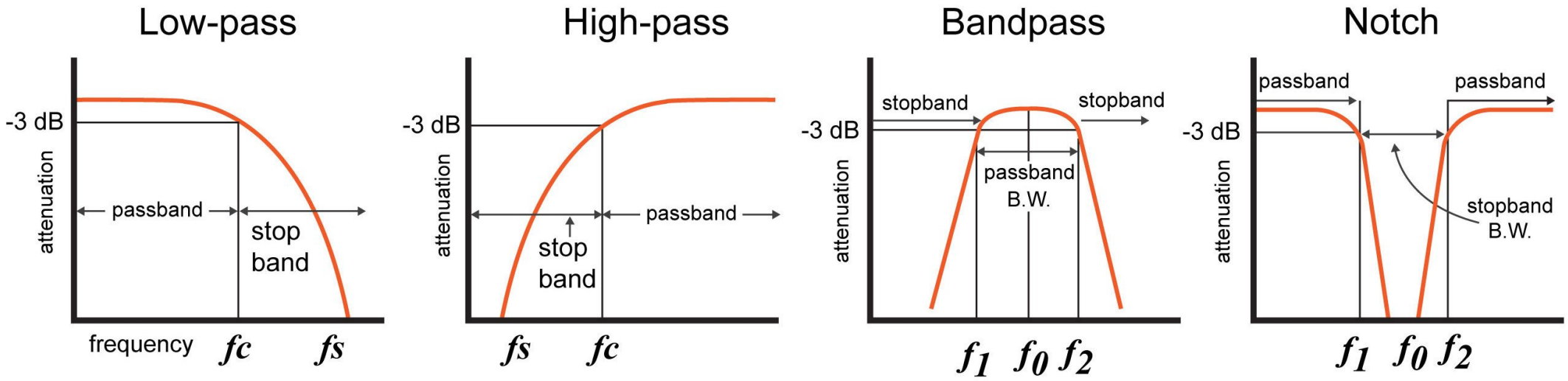
Filtering

- Why should we add a filter?
  - Unfiltered noise can cause jittery motion and actuator to overwork (rapid changes)
  - Rapid changes in signal cause more control effort
- So filtering noisy signal
  - Reduces unnecessary actuator activity
  - Reduces the energy and maintenance cost
  - Extends system lifetime
- **Note: Filters can introduce delay to the system, which may cause instability. Avoid over-filtering**



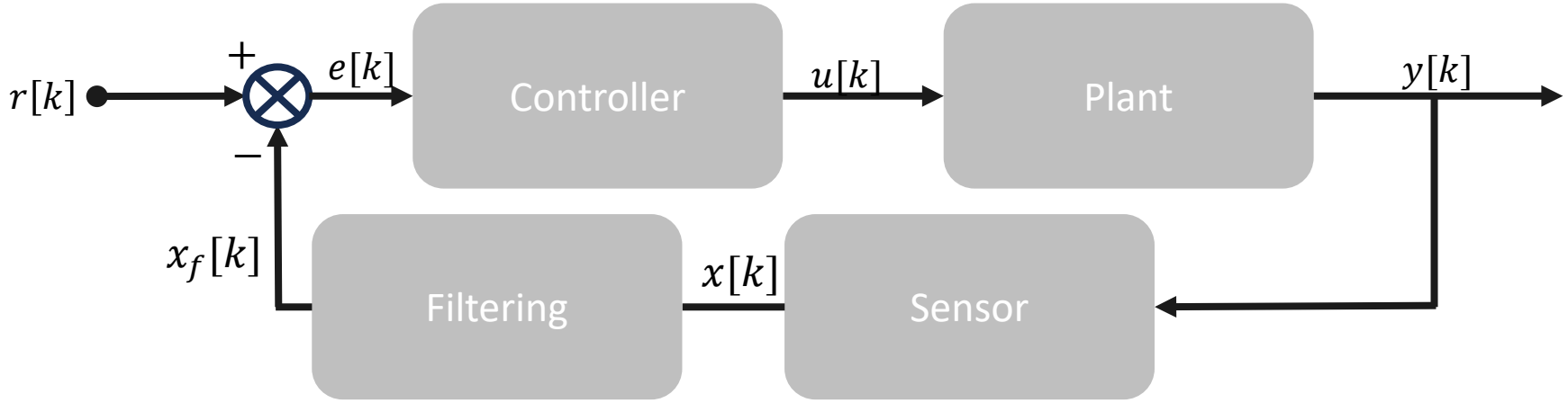
Filtering





https://www.allaboutcircuits.com/technical-articles/an-introduction-to-filters/

Filtering

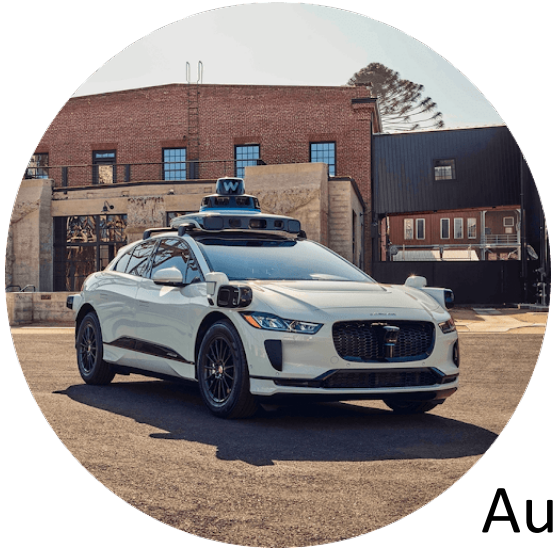


1. $Initialize$
2. $raw[k] \leftarrow read\_measurements()$
3. $filtered[k] = \alpha \cdot raw[k] + (1 - \alpha) \cdot filtered[k-1], \quad \alpha \in [0, 1]$

- α close to 1: faster response, less smoothing
- α close to 0: slower response, more smoothing

Examples

Autonomous Car



Smart Home

Closed-loop Systems

Autonomous Factory

Biology

Examples

Goal

- Autonomous cruising
  - Maintain speed
  - Stay in the lane
  - Avoid collision

Controllable Objects

- Engine / Motor throttle
- Steering
- Brake

Information

- Current location
- Map
- Speed
- Power state
- Road situation
- Traffic Info

Control Input

- Goal speed
- Goal heading

Control Output

- Power output
- Steering angle
- Brake

Cost Factors

- Fuel efficiency
- Time
- Comfort
- Safety

Control Feedback State

- True speed
- True heading

Examples

Goal

- Maintain room temperature and humidity
  - Save energy
  - Comfortable, healthy
  - Economic

Controllable Objects

- AC system
  - Air flow rate
  - Air temperature
- Window, door
- Window shades

Information

- Outdoor climate
- Outdoor air condition
- Indoor air condition
- AC system state
- Electricity price
- Indoor people activity

Control Input

- Goal room temperature and humidity

Control Output

- AC air flow rate
- AC air temperature
- Window, door
- Window shades

Control Feedback State

- True room temperature and humidity

Cost Factor

- Economy
- Comfort
- Healthy

Examples

Goal

- Blood glucose regulation in human body
  - Maintain blood glucose level

Controllable Objects

- Pancreas
  - Insulin secretion
    - Reduce blood sugar
  - Glucagon secretion
    - Signal the liver to produce glucose
- Brain
  - Eat!
  - No more food!

Information

- Pancreatic Beta Cell
  - Glucose sensors
- Diet condition

Control Input

- Normal/healthy blood glucose level

Control Output

- Secretion speed of Insulin
- Secretion speed of Glucagon

Control Feedback State

- True blood glucose level

Cost Factor

- Diet condition