Introduction to Hardware and Closed-loop System

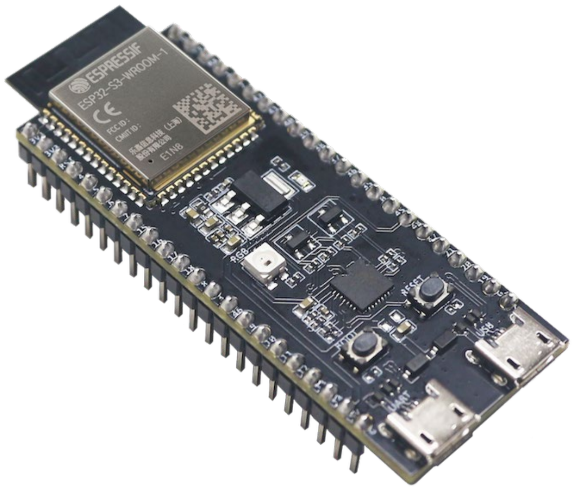
Prof. Jamie Paik
Ekrem Yüksel
Reconfigurable Robotics Laboratory
EPFL, Switzerland

Learning Objectives

- Differentiate between open-loop and closed-loop systems
- The importance of closing the loop
- Describe the structure and components of a PID controller
- How noise and disturbance affect the system
- Role of filtering
- Additional strategies on how to minimize external disturbances
- What is human-in-the-loop (HITL) control works

Introduction to Hardware

- **Sensors –** what do you want to measure?
- **Actuators –** what action will your system have?
- **Microcontrollers –** how much processing power do you need? E.g., ESP32, Arduino
- **Power –** how will you excite your system? Will it be electric or pneumatic? Does your system need mobility (battery), or will it be tethered?

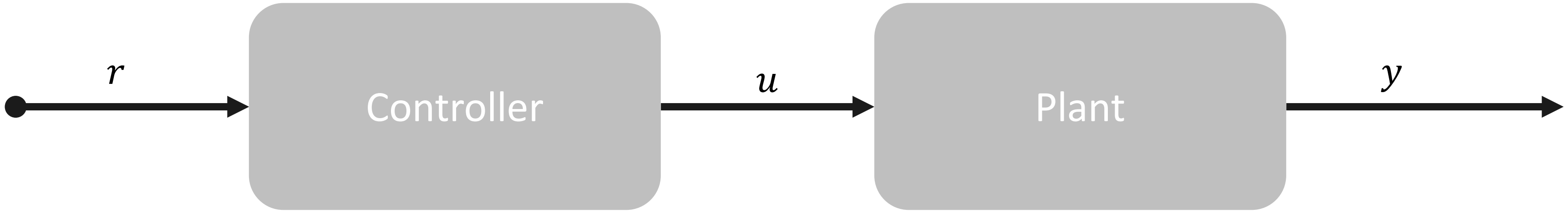


Key questions to ask:

- What variable do I want to control?
- What do I need to measure to control it?
- Can I sense and act fast enough to matter?

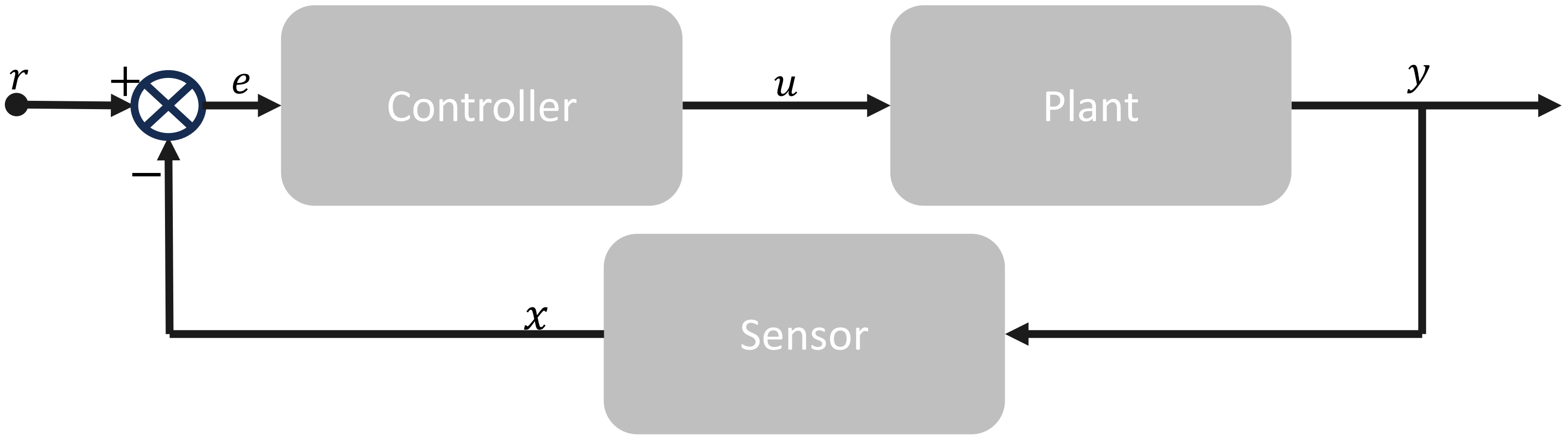
Control Systems

- **Open loop System:**
  - System without any feedback
  - No direct control of the output/behavior of the system
  - Predictable only if the plant is well-known, no disturbance rejection
  - Examples: Turning up the volume, toaster, gas stove



Control Systems

- **Closed loop System:**
  - System with state feedback
  - Reads, processes, and regulates the system to maintain the desired state
  - Low to high disturbance rejection
  - Examples: Exoskeleton gait assistance, head tracking VR headset,



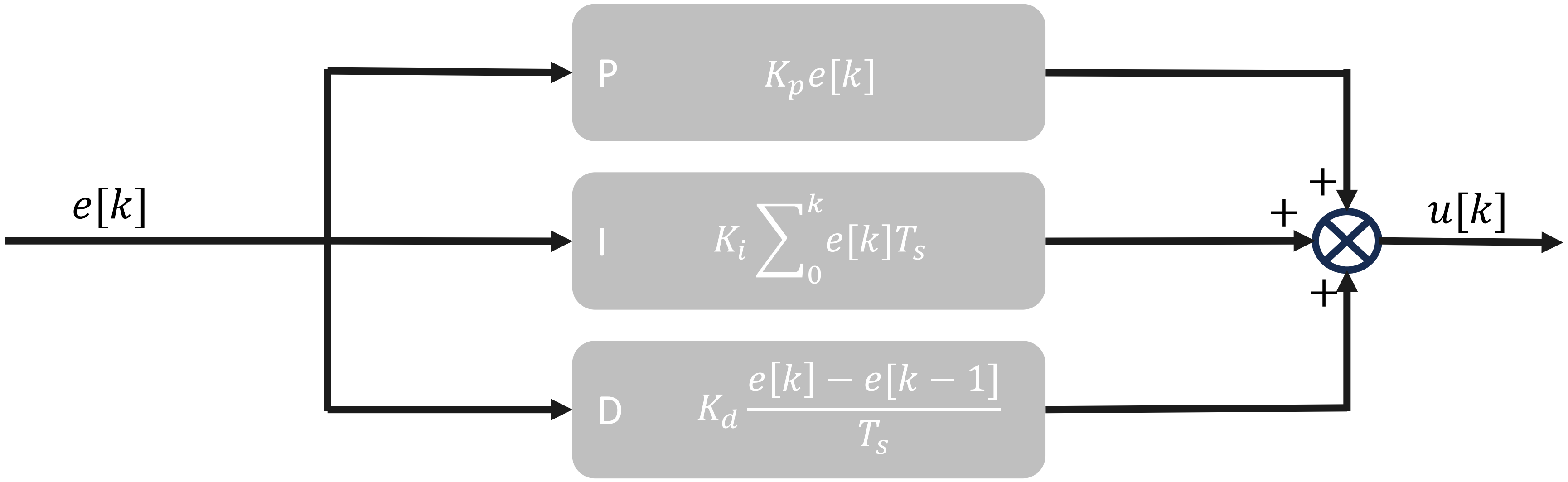
Open Loop vs Closed Loop

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 (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, otherwise unstable	Can stabilize systems that are unstable in open-loop — but it can also become unstable itself if the controller (e.g., PID) 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

Why Closing the Loop?

- Personalization and adaptability (varying human parameters)
  - Fixed commands in OL can’t adapt across users
  - Human variability (height, weight, length of limbs, etc.)
  - Adaptability to changes in the environment
- Consistent and reliable performance (output regulation)
- Safety for human interaction
- Energy efficiency, especially if it runs on battery

Simple PID Implementation

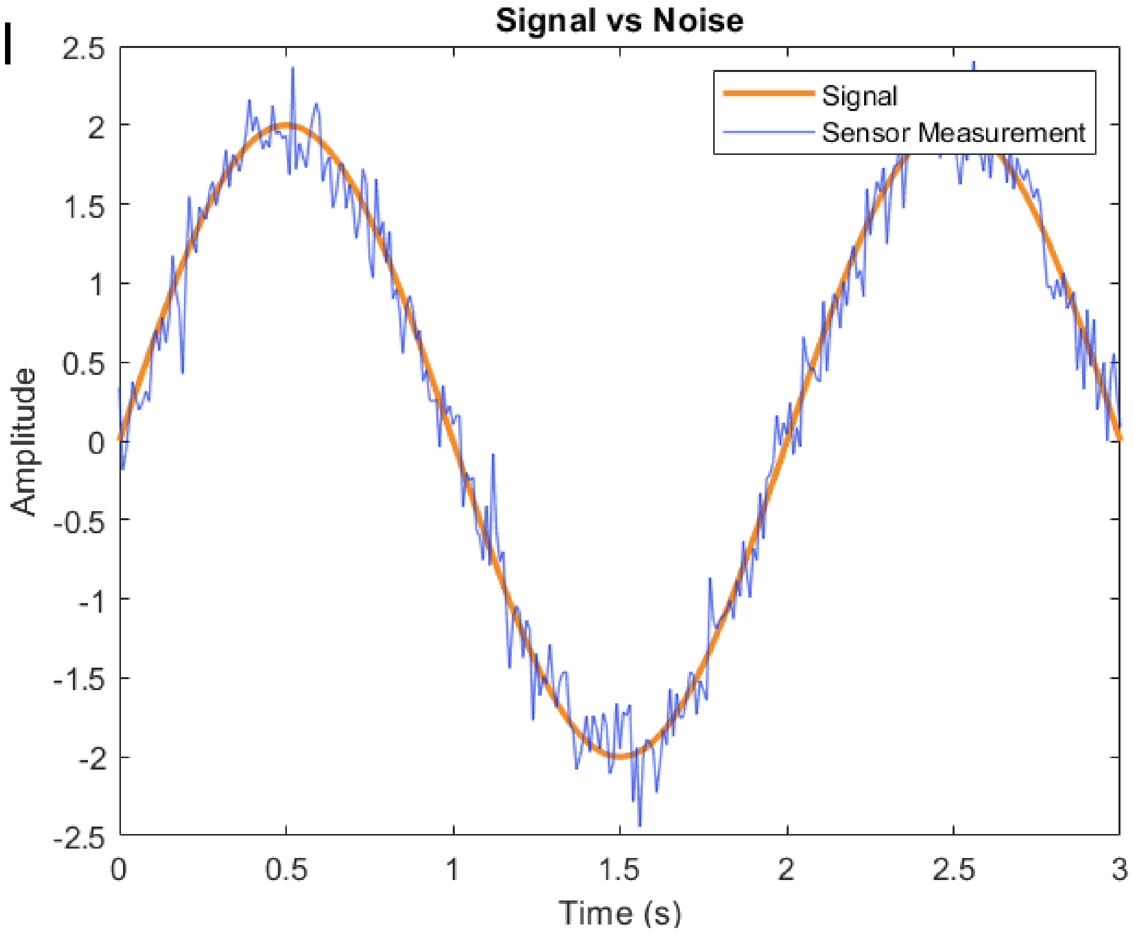



```python
previous_error = 0.0
error_sum = 0.0
while True:
    measurement = read_measurements()
    error = desired - measurement
    error_sum += error * Ts
    control_input = Kp * error + Ki * error_sum + Kd * (error - previous_error) / Ts
    send_control_signal(control_input)
    previous_error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

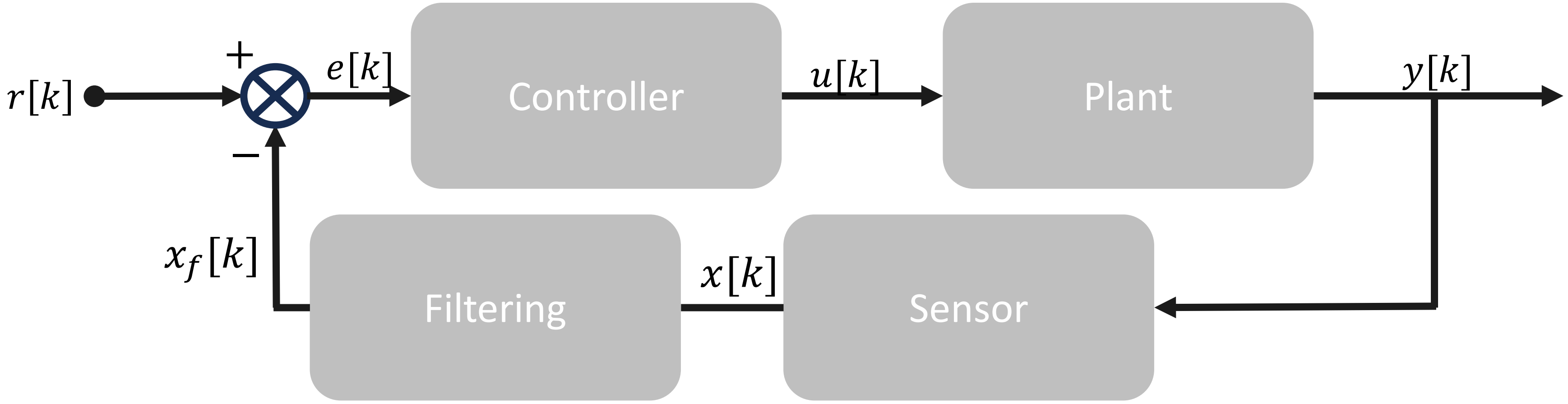
Noise

Noise: Undesired and unpredictable signal that interferes with the measurement or the behavior of the system. It is not part of the desired input/output.

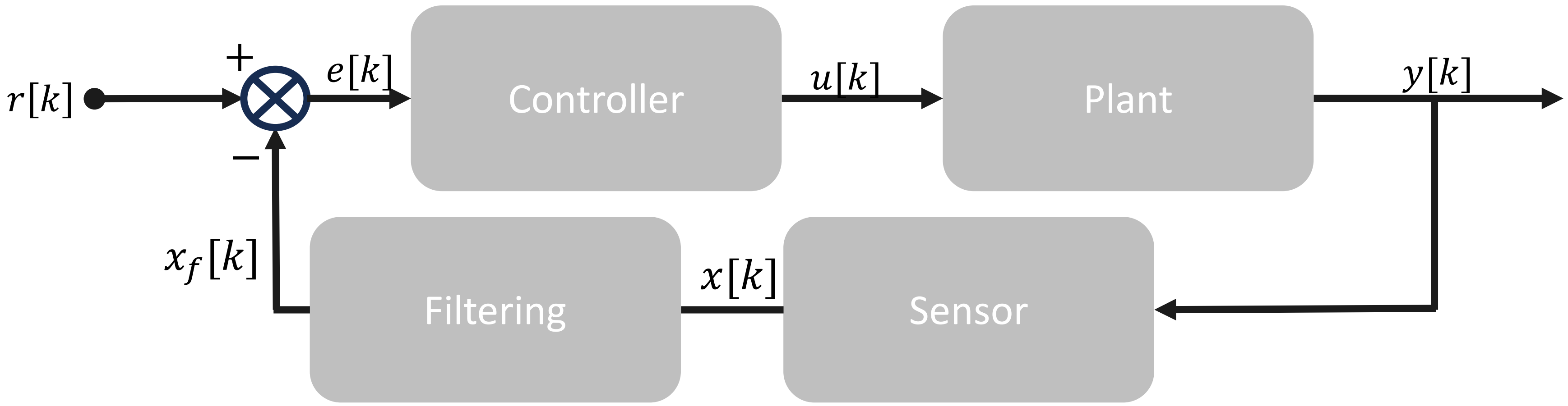
- Electronic sensor signals are noisy
  - Electrical/Electronic:
    - Electrical interference
    - Quantization errors (Analog-to-digital)
  - Mechanical
    - Gravity
    - Mechanical vibrations & backlash
- The derivative term in PID amplifies the noise
  - Can drive system to instability
  - Can cause jittery motion



Filtering



Filtering



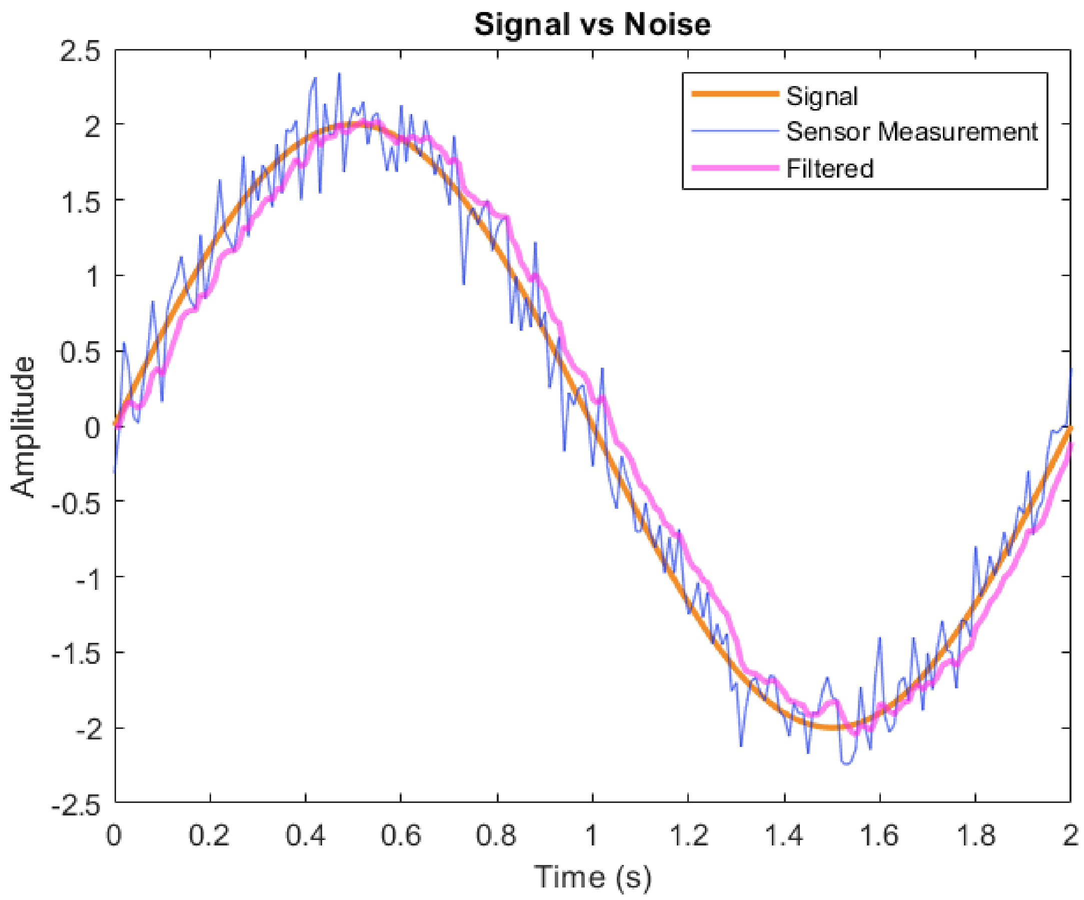

```
raw = read_sensor()
filtered = [0]
alpha = 0.005
k = 0
while True:
    k += 1
    raw[k] = read_sensor()
    filtered[k] = alpha * raw[k] + (1 - alpha) * filtered[k - 1]
    send_data(filtered[k])
```

EPFL

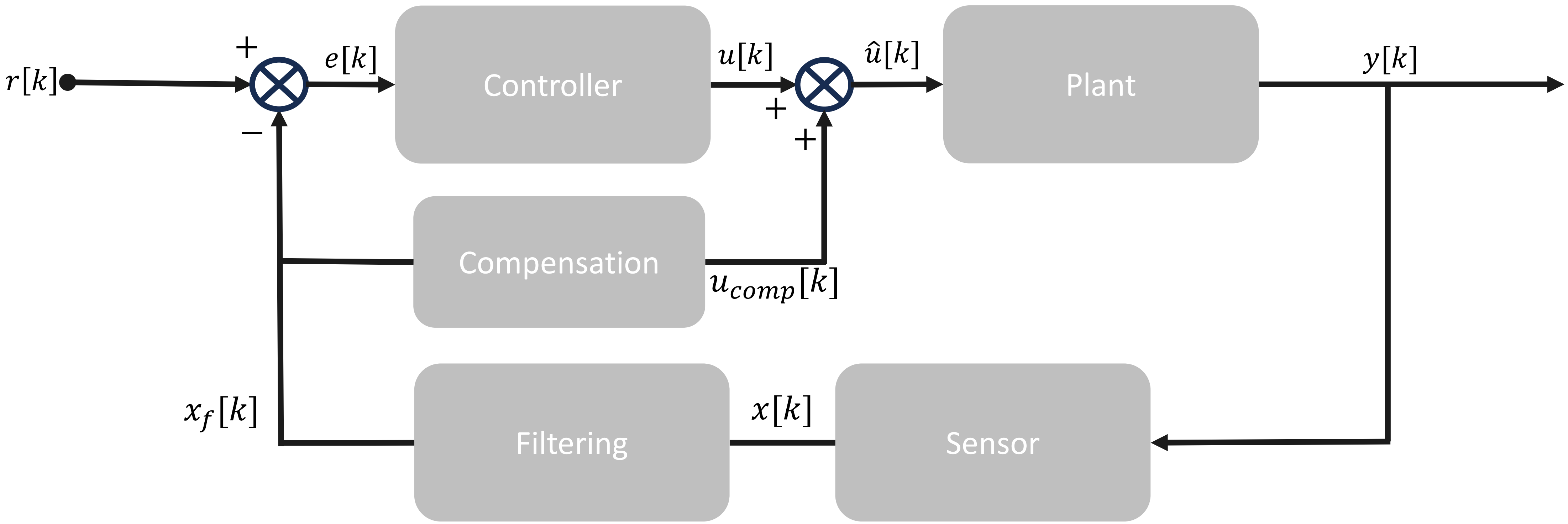
Filtering

- 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

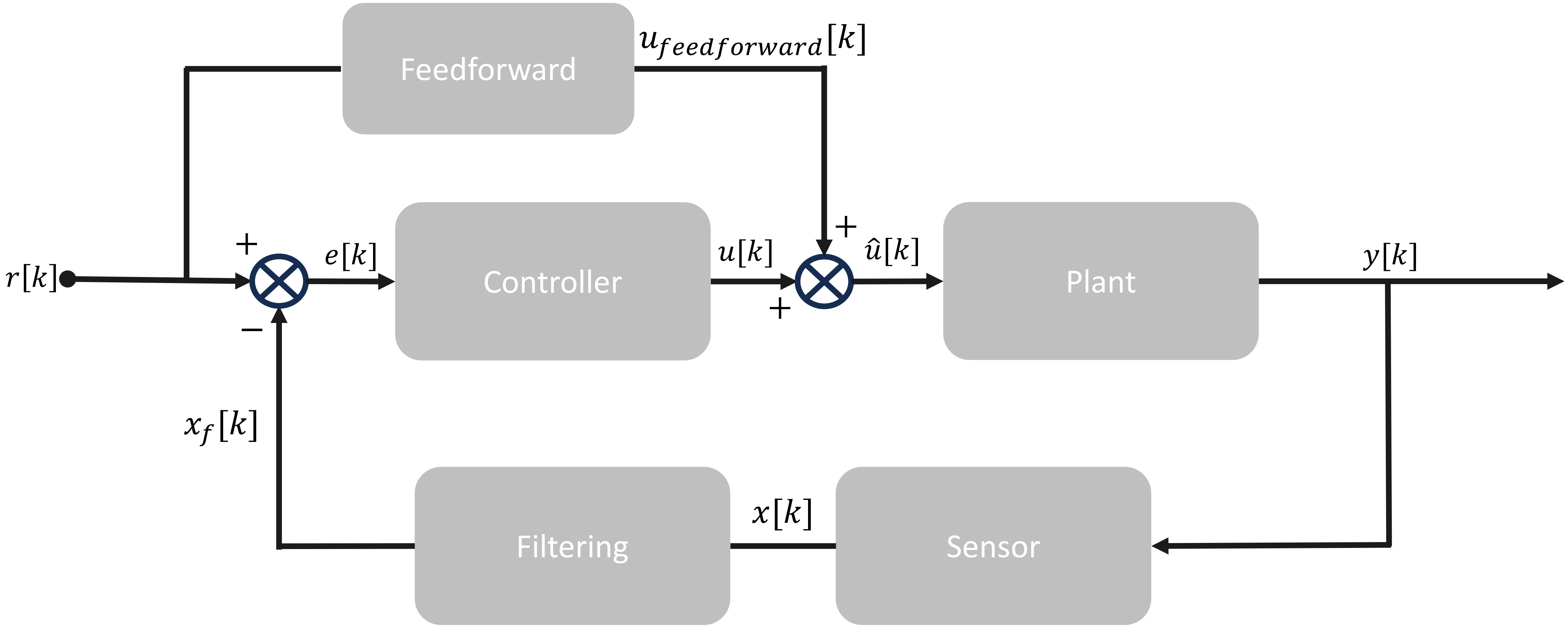
- **Note: Filters can introduce delay to the system, which may cause instability. Avoid over-filtering**



Compensation



Feedforward Control



Human in the Loop (HITL)

- **Output/behavior of the system depends on the Human**
  - The body is part of the control loop

Core Requirements:

1. **User state is measured.**
   - Examples: joint angles, muscle activity (EMG), posture, intention.
2. **Measured state is used to compute control actions.**
   - The controller uses this input to decide what the system should do next.
3. **The system acts back on the human.**
   - Through forces, motion, pressure, feedback — closing the loop.

Human in the Loop (HITL)

System	Senses Human?	Adapts to Human Behavior?	Provides Feedback to Human?	Closed-Loop Control?	Human-in-the-Loop?	Notes
VR goggles (head tracking)	☑ (IMU/camera)	☑ (scene updates in real time)	☑ (visual)	☑ Perceptual closed-loop	☑ (perceptual loop)	Feedback alters what you see
VR goggles (video only)	✗	✗	☑ (visual)	✗	✗	Static playback
Haptic glove (sensing)	✗	✗	☑ (tactile)	✗	✗	Feedback only, t adaptive
Haptic glove (with sensors)	☑ (IMU, flex)	☑ (force adjusts with interaction)	☑	☑	☑	Example: adaptive grip force
Controller (button-based)	✗ (discrete input)	✗	✗ Optional	✗	✗	Command-only interaction
Joystick (analog control)	☑ Partially	☑ With continuous mapping	✗ Optional	◍ Semi (open-loop from system's view)	◍ Maybe (depends on system response)	Depends on whether system adjusts
Exoskeleton (preprogrammed)	✗	✗	✗ Optional	✗	✗	Fixed motion, sensing
Exoskeleton (sensor-driven)	☑ (EMG, IMU)	☑	☑ (haptic, assistive)	☑	☑	Example: assistive knee brace
Posture vest (vibrates every 30 min)	✗	✗	☑	✗	✗	Time-triggered feedback
Smart compression sleeve (sensor-based)	☑ (pressure sensors)	☑	☑ (fit/comfort)	☑	☑	Adjusts to maintain safe pressure

Human in the Loop (HITL)

Autonomyo – SCI-STI-MB - EPFL

REHAssist

TWIICE – SCI-STI-MB - EPFL

Human in the Loop (HITL)

- **Adaptation to each user**
  - Comfort and ergonomics
  - User intent & timing
  - Sensors for event detection, e.g. force sensor for gait detection
  - Closed-loop for assistive behavior
- **Fighting against the user**
  - Feels unnatural
  - Increases cognitive and/or physical effort
  - Reduces the effectiveness