Controls

CSCI 420-04 Robotics

WILLIAM & MARY

CHARTERED 1693

Performing Actions in the World

- Sensors are noisy
  - Do we really know the world?
- Actuators are noisy
  - Did the motor do what we asked?
- The world is messy
  - Different environment conditions mean different actions: wind pushes drone, slick road for AV, …

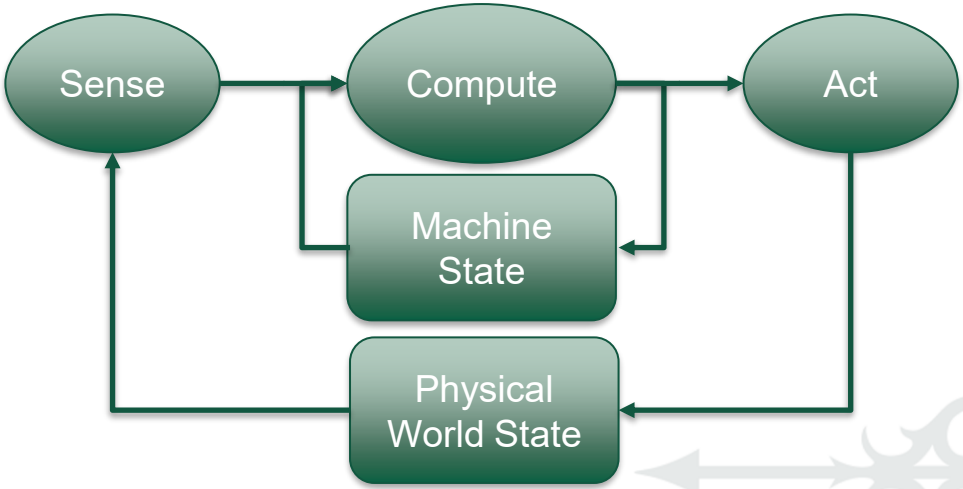
Examples

- Keep the A/C at the desired temperature

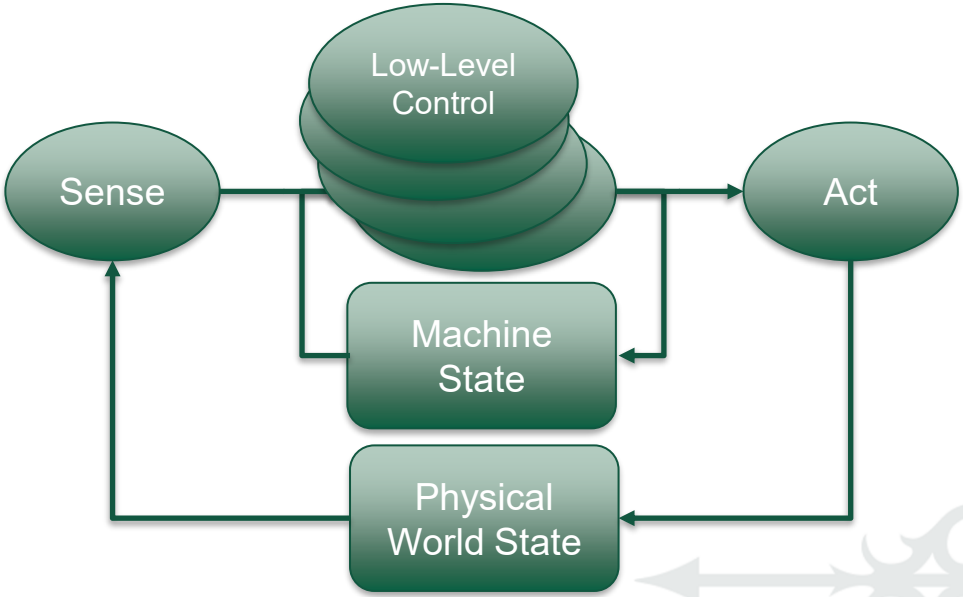

- Maintain steady cruise control speed

- Fly at a constant altitude

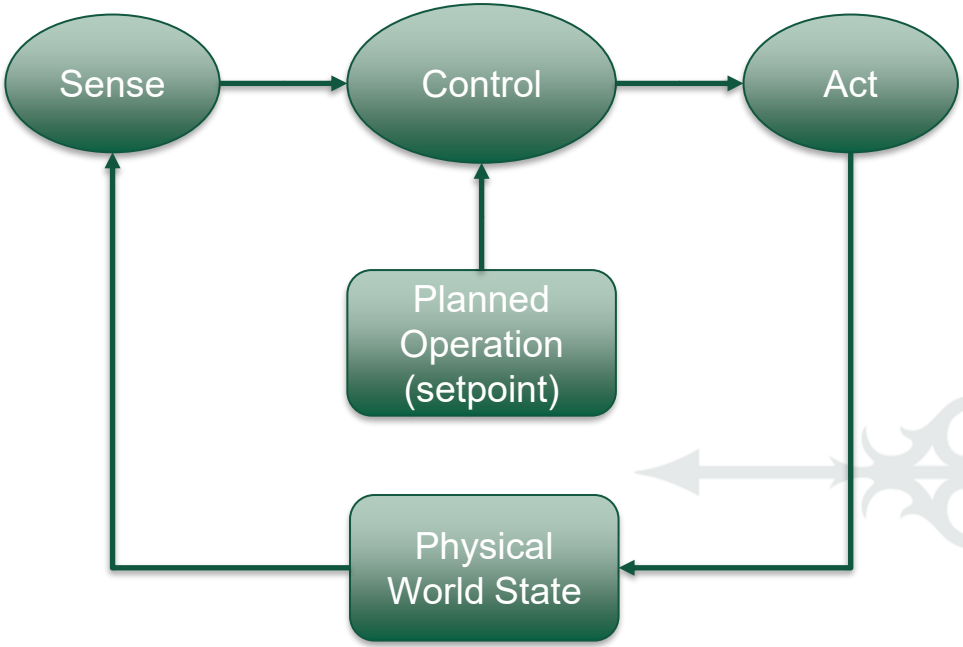
How do we choose the action?



How do we choose the action?



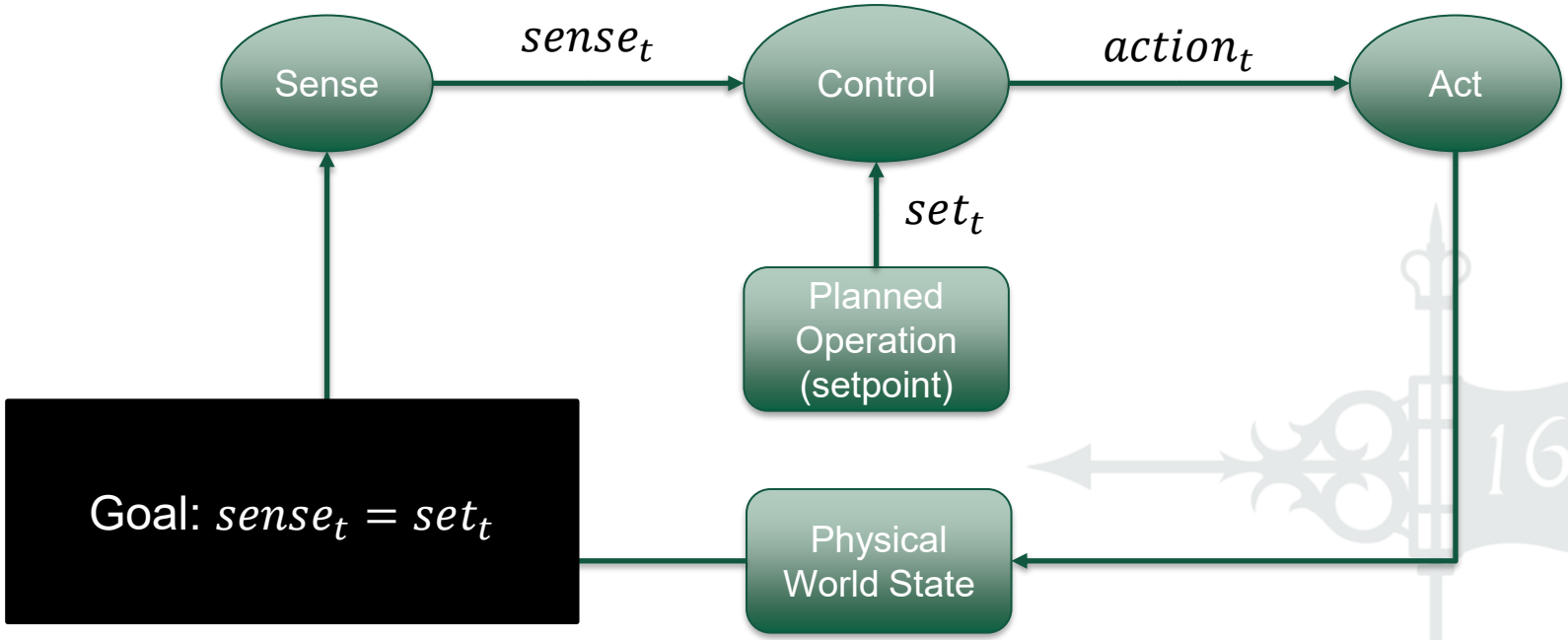
How do we choose the action?



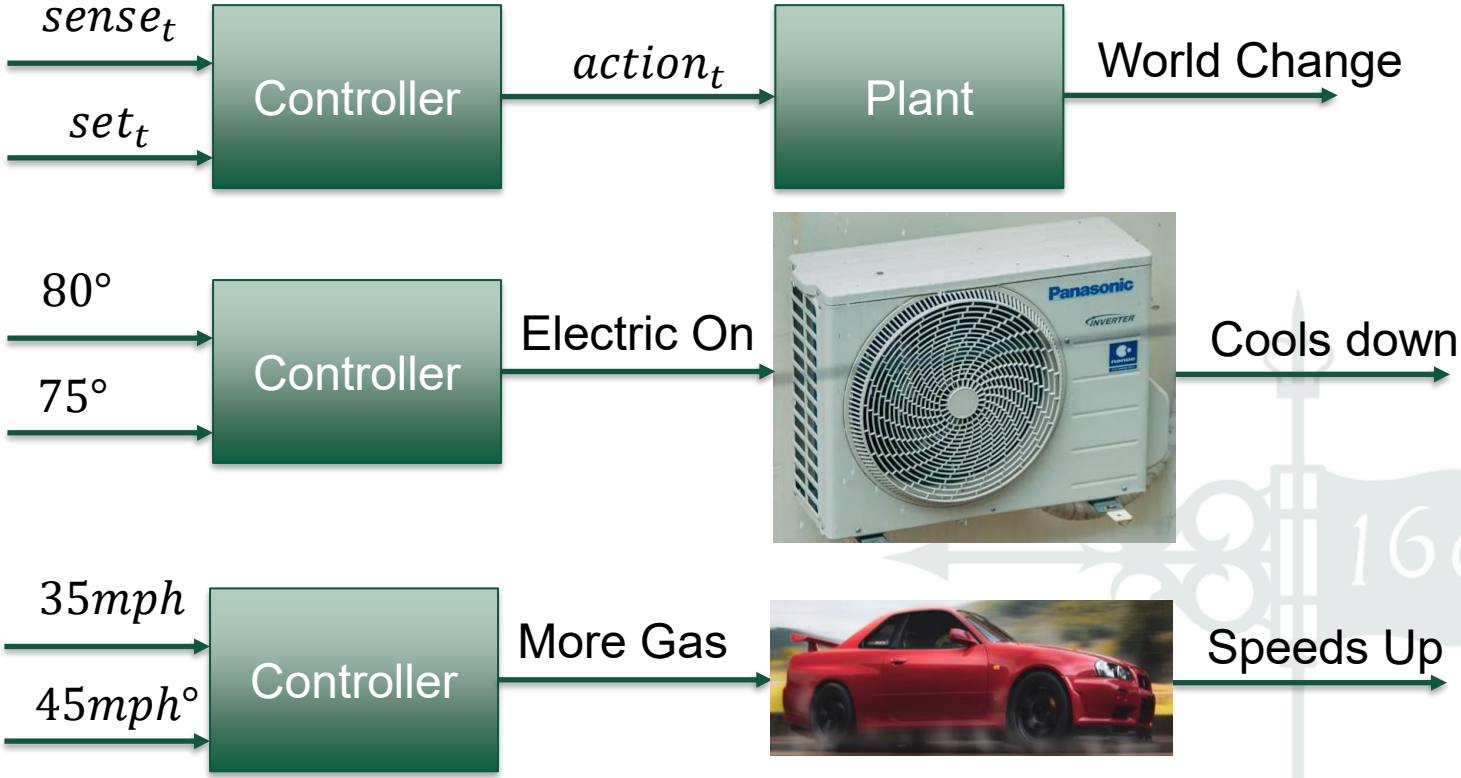
Setpoint is Desired Outcome

- The desired temperature of the A/C
- The speed of the car
- The altitude of the drone

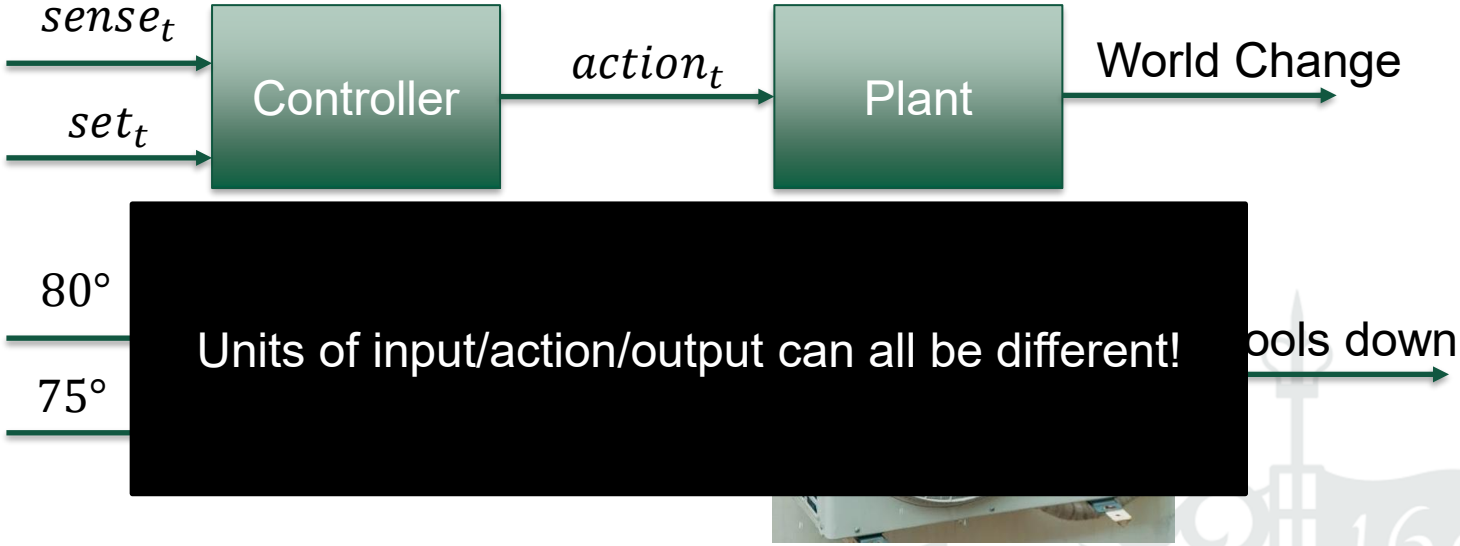
How do we meet the setpoint?



Focus on Controls



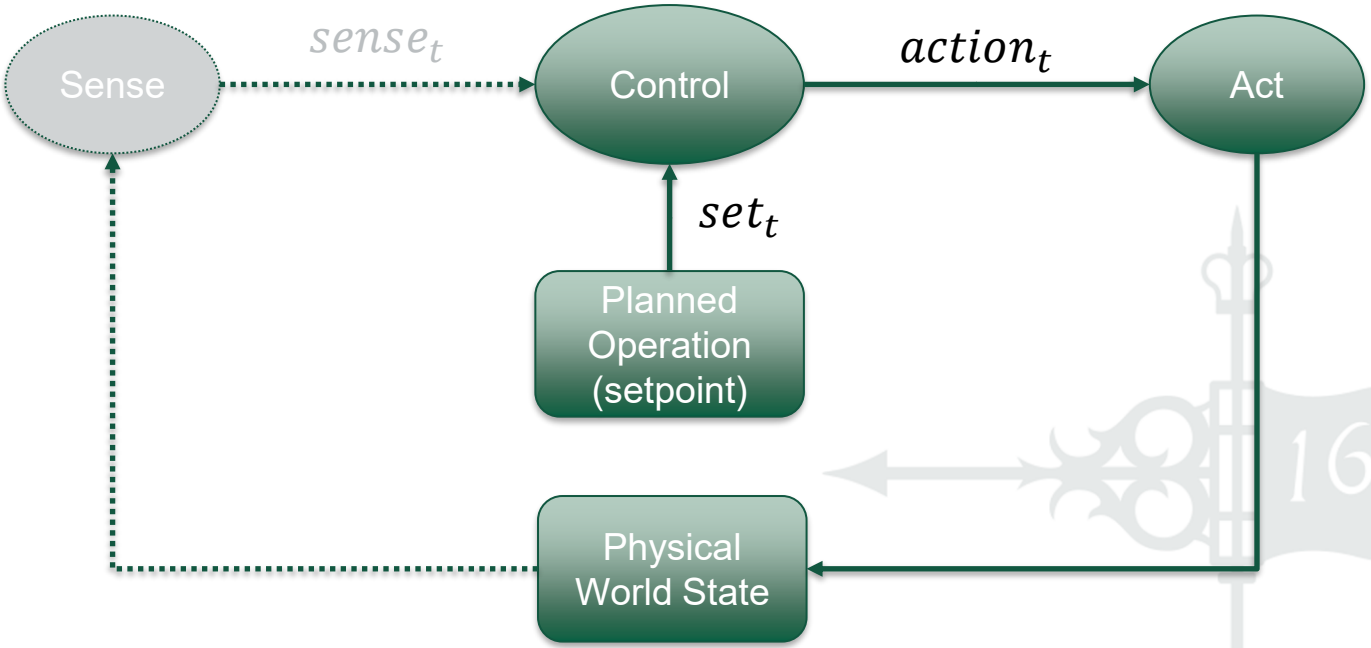
Focus on Controls



Types of Controllers

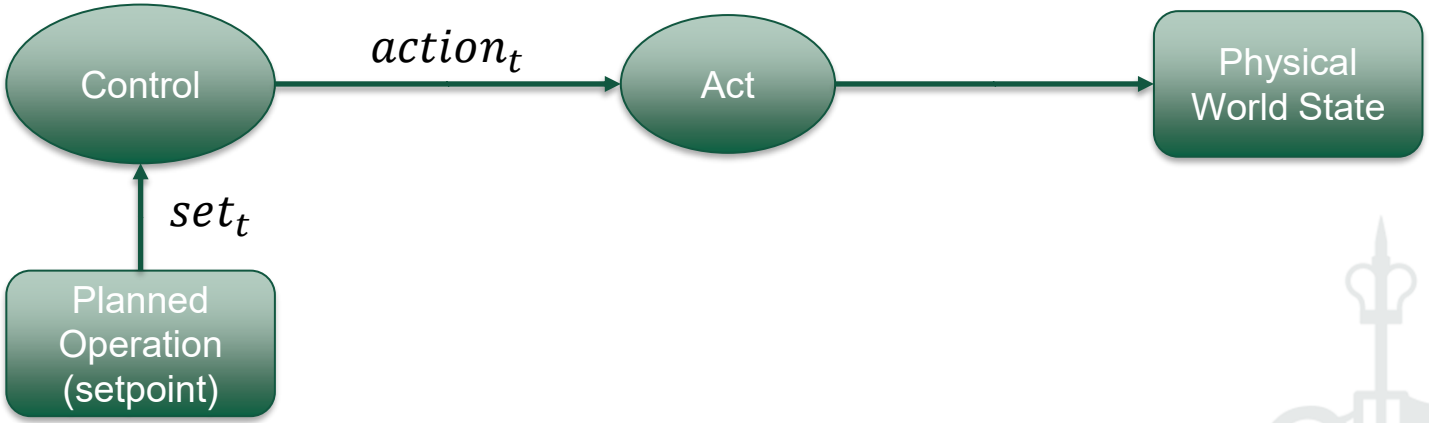
- Open-loop
  - No feedback from the environment
- Closed-loop
  - Feedback from the environment

Open Loop Controller



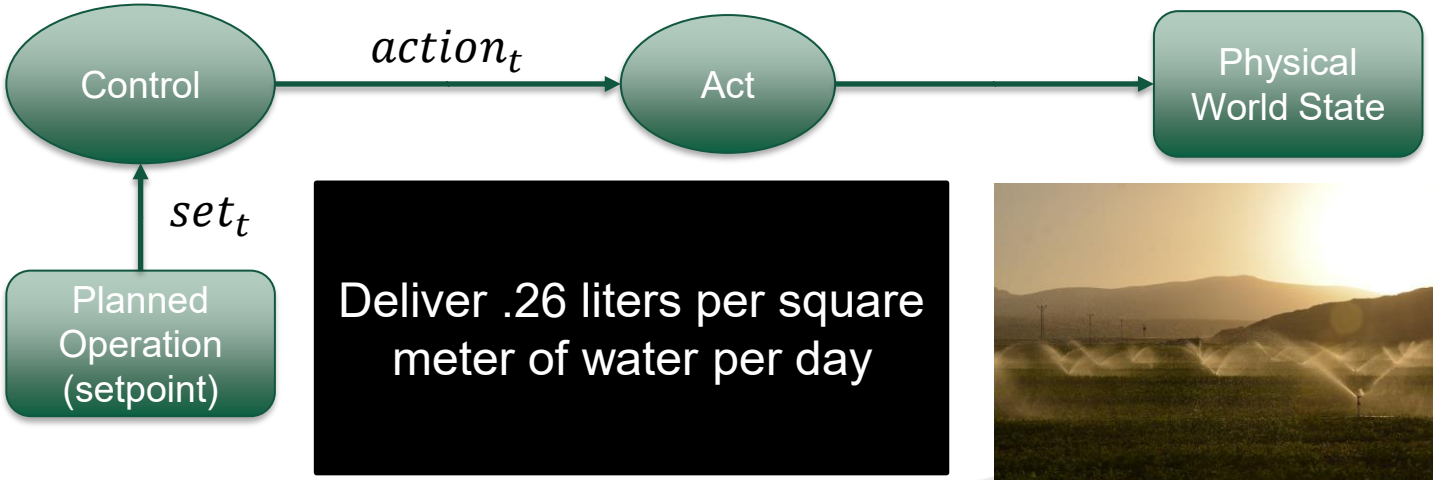
Open Loop Controller

$$action_t = F(set_t)$$



Open Loop Controller

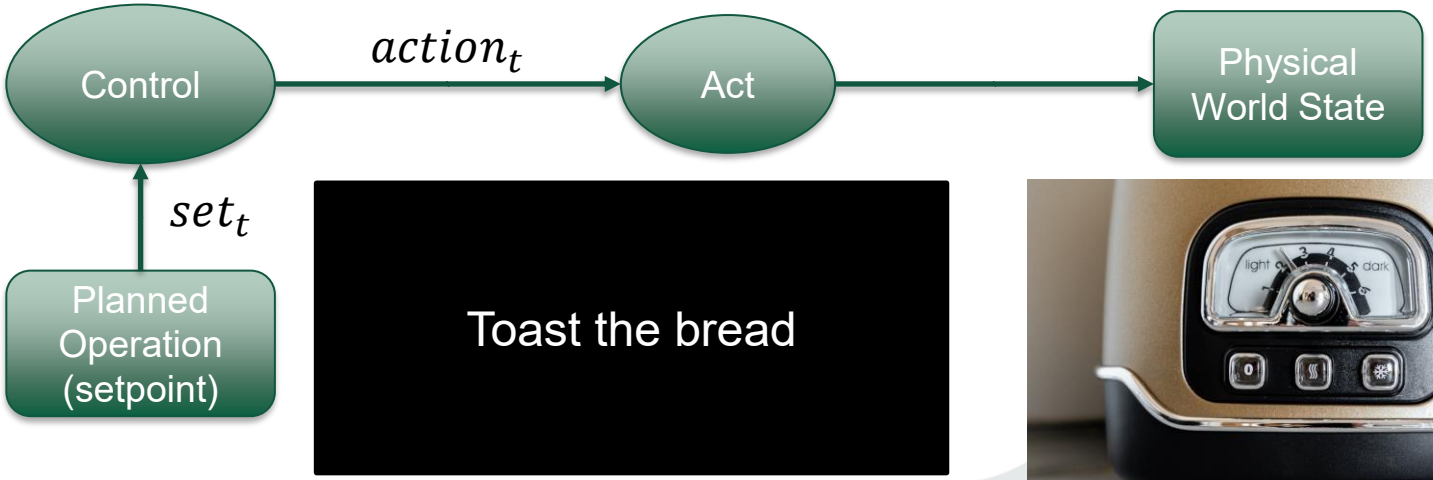
$$action_t = F(set_t)$$



Set of .26 liters per square meter, flow rate of 0.5 liters per minute over an area of 100 square meters means run for 52 minutes/day

Open Loop Controller

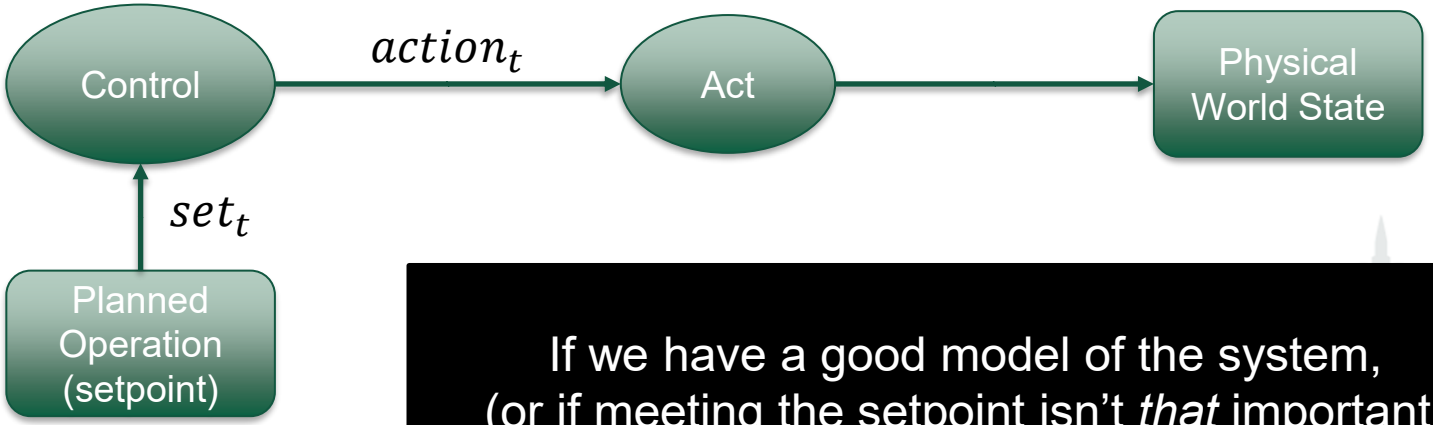
$$action_t = F(set_t)$$



What does it mean to be “level 2 toast”?

Open Loop Controller

$$action_t = F(set_t)$$



If we have a good model of the system,
(or if meeting the setpoint isn't *that* important),
we can skip the feedback

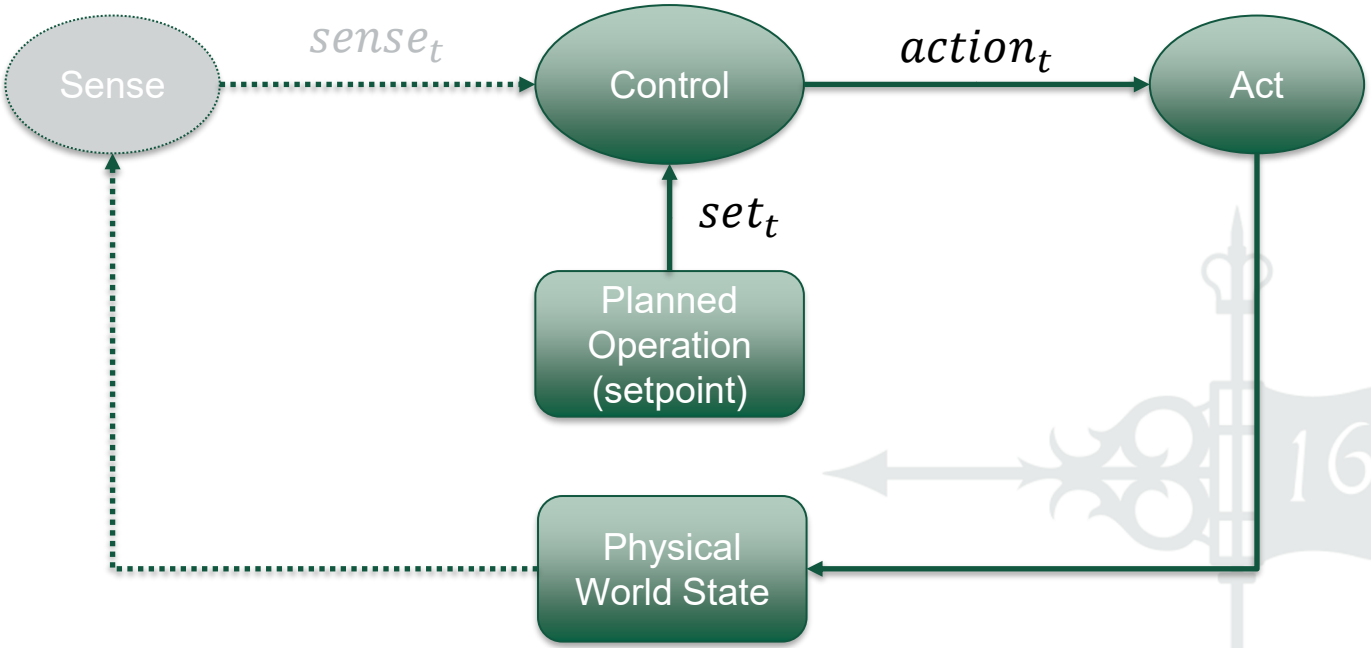
Open Loop Control is Hard!

- Close your eyes
- Rotate 5 times in place
- Walk 3 steps
- Repeat 3 times
- You should be back where you started
- Are you?

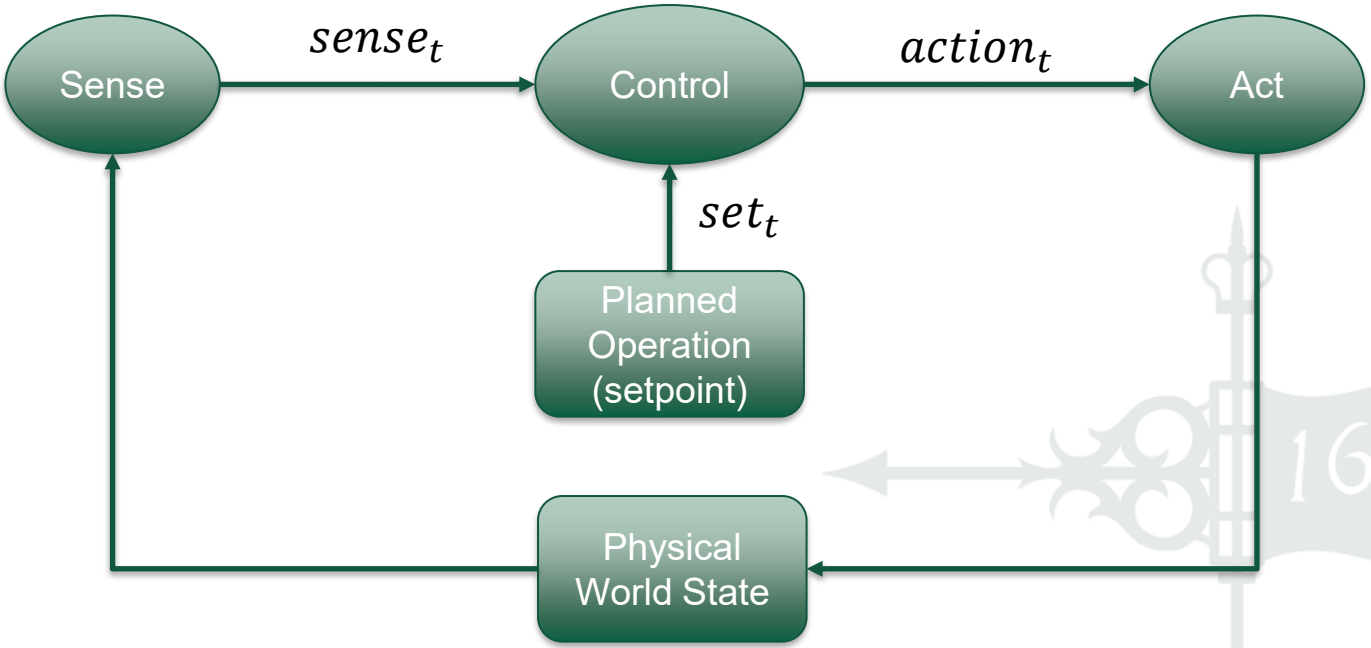
Open Loop Control

- Cannot account for error:
  - Actuation Error
    - The sprinkler didn't go for the time we commanded
  - Model Error
    - The flow rate of the sprinkler was wrong

Closed Loop Control: Feedback

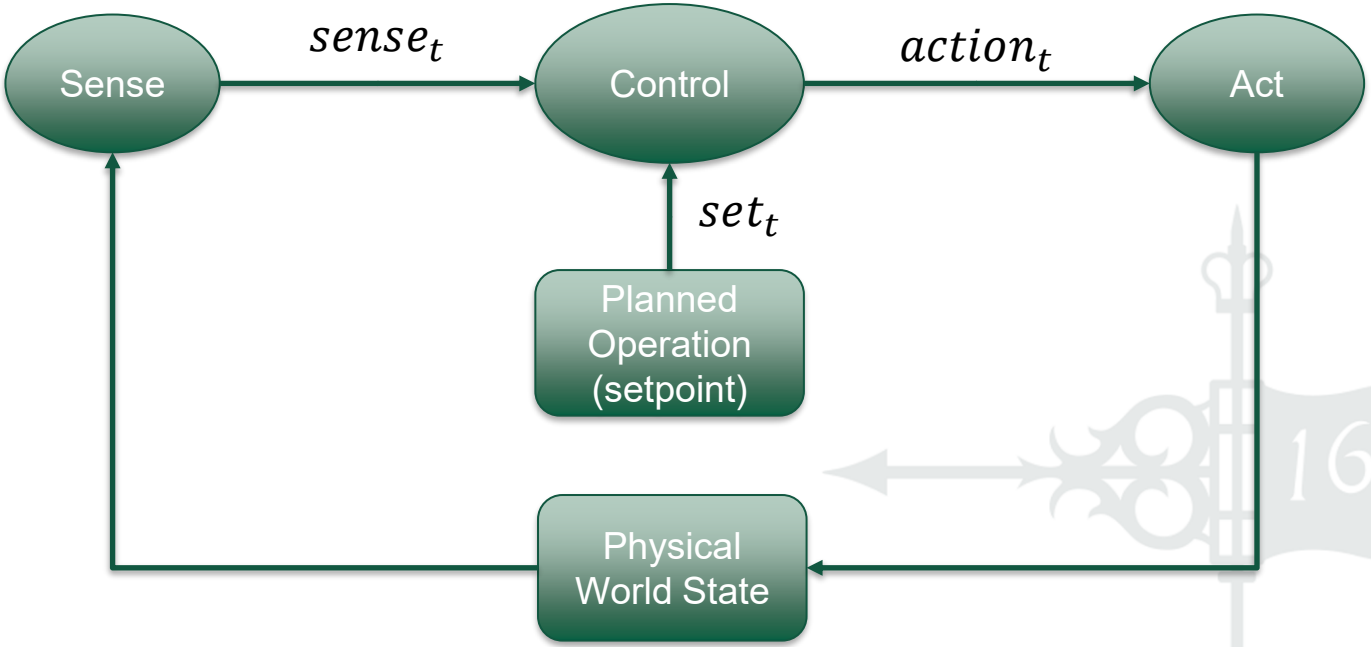


Closed Loop Control: Feedback



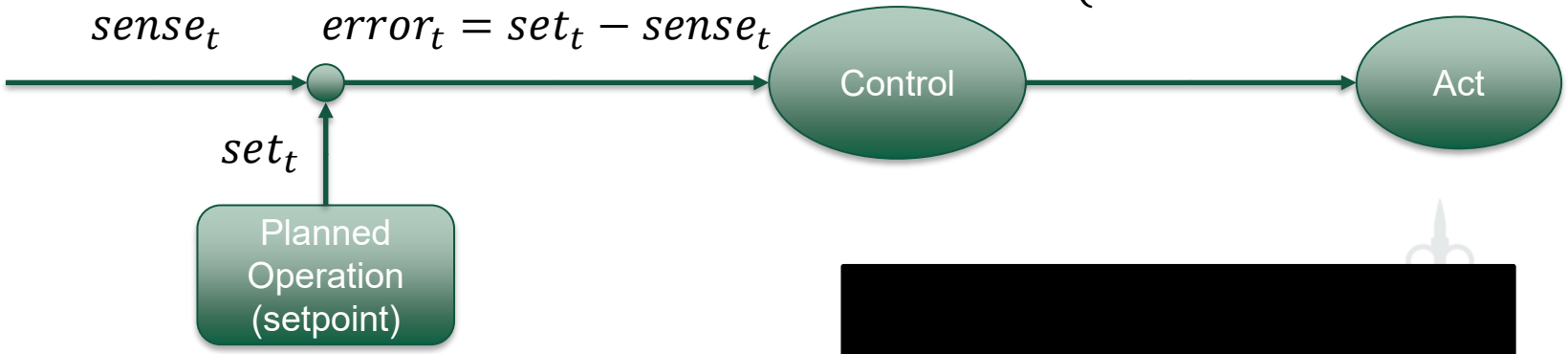
Closed Loop Control: Feedback

$$error_t = set_t - sense_t$$
$$action_t = F(error_t)$$



Bang-Bang Controller

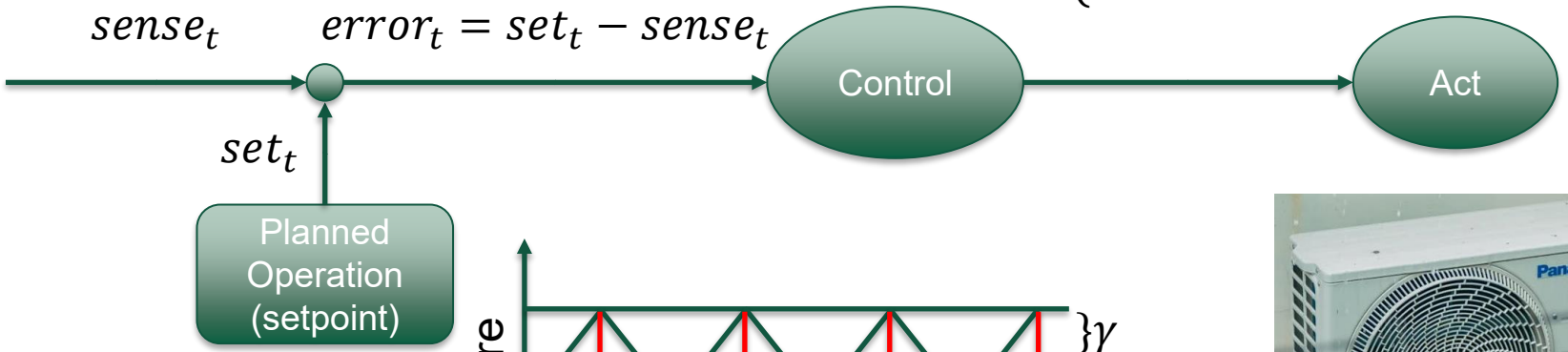
$$action_t = \begin{cases} ON & if\ error_t > \delta \\ OFF & if\ error_t < \gamma \end{cases}$$

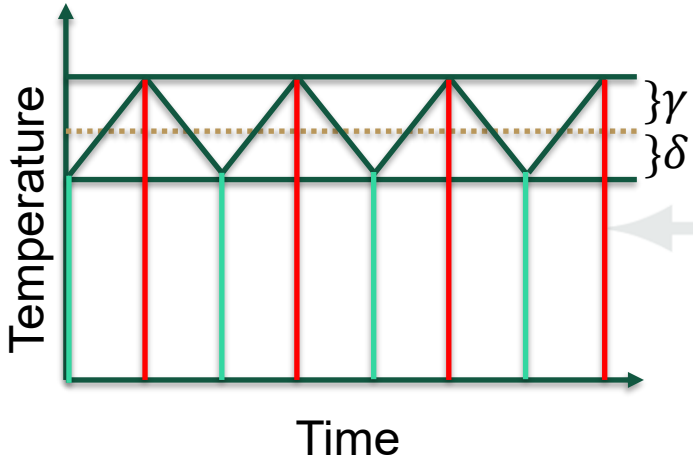


Simple controller for plants that can either be on/off

Bang-Bang Controller

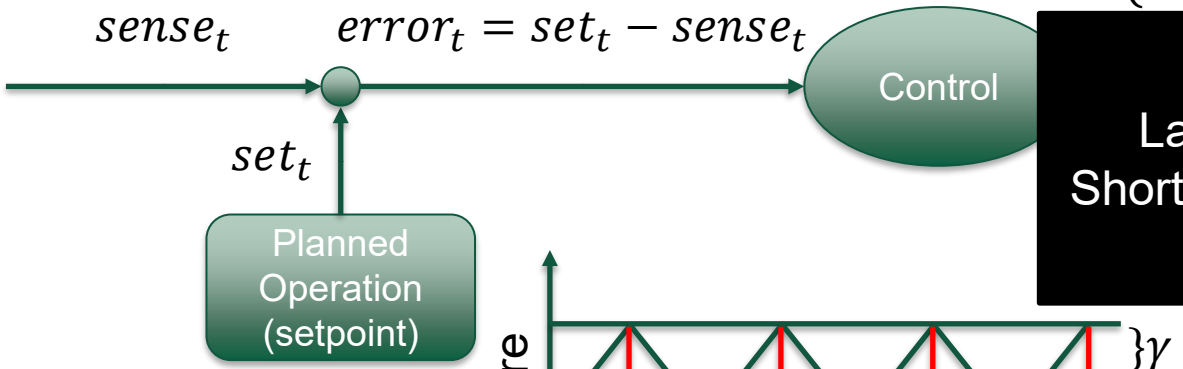
$$action_t = \begin{cases} ON & if\ error_t > \delta \\ OFF & if\ error_t < \gamma \end{cases}$$



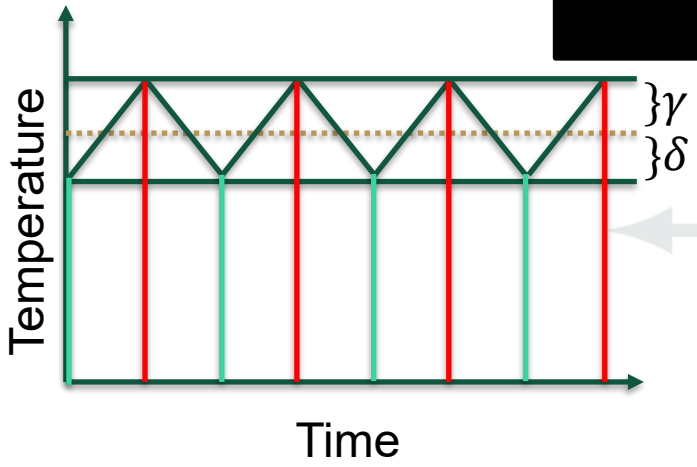


Bang-Bang Controller

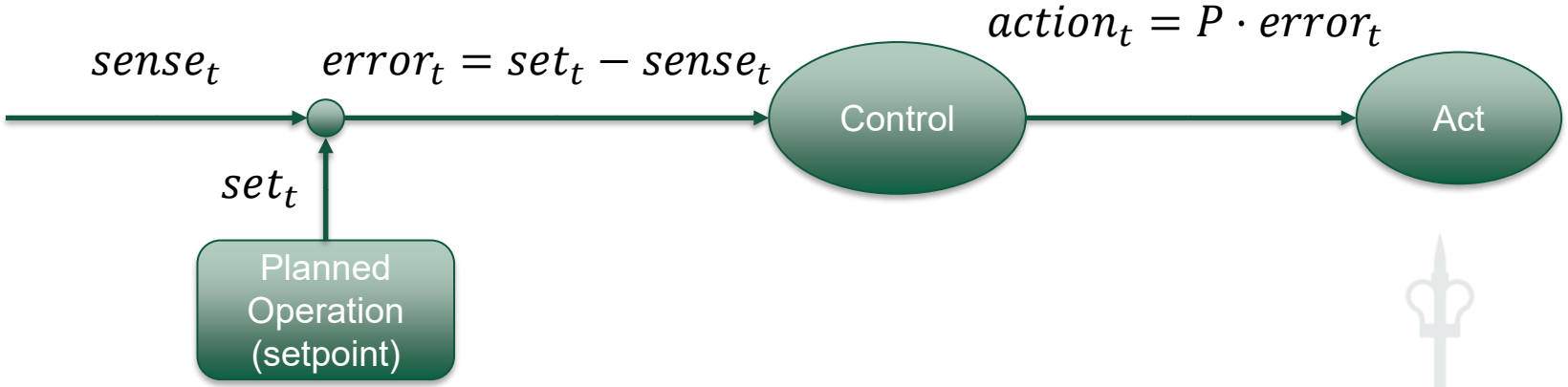
$$action_t = \begin{cases} ON & if\ error_t > \delta \\ OFF & if\ error_t < \gamma \end{cases}$$



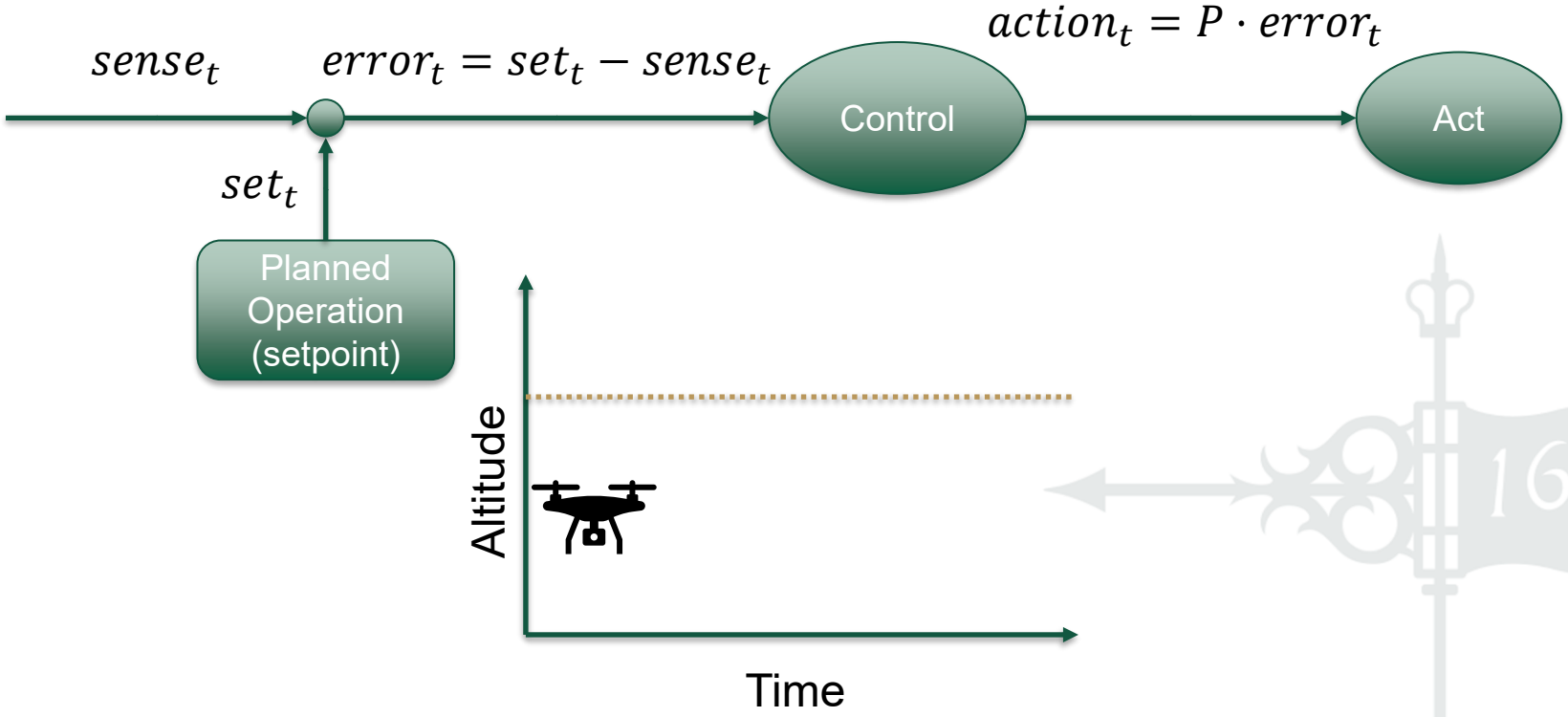
Large range – uncomfortable
Short range – lots of on/off, burnout



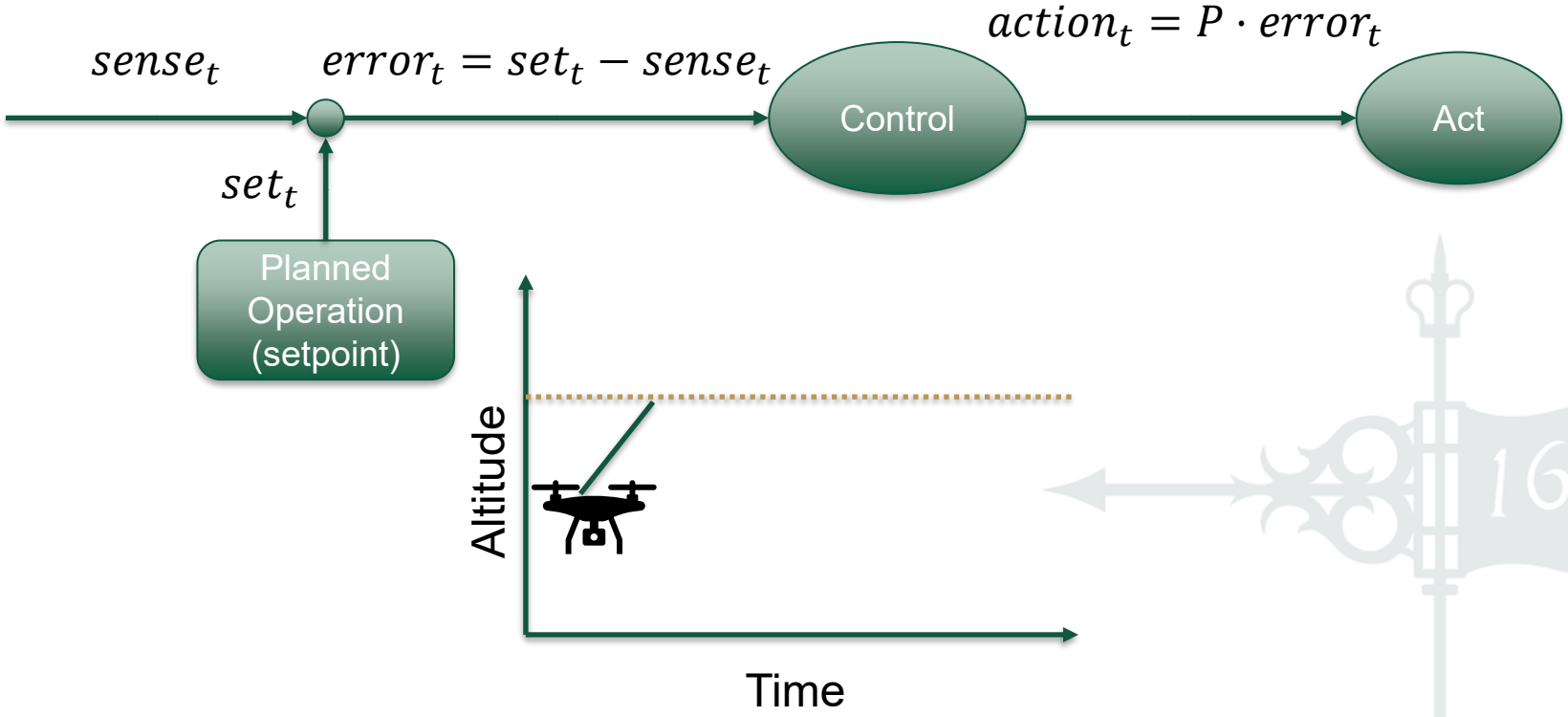
Closed-Loop: Proportional



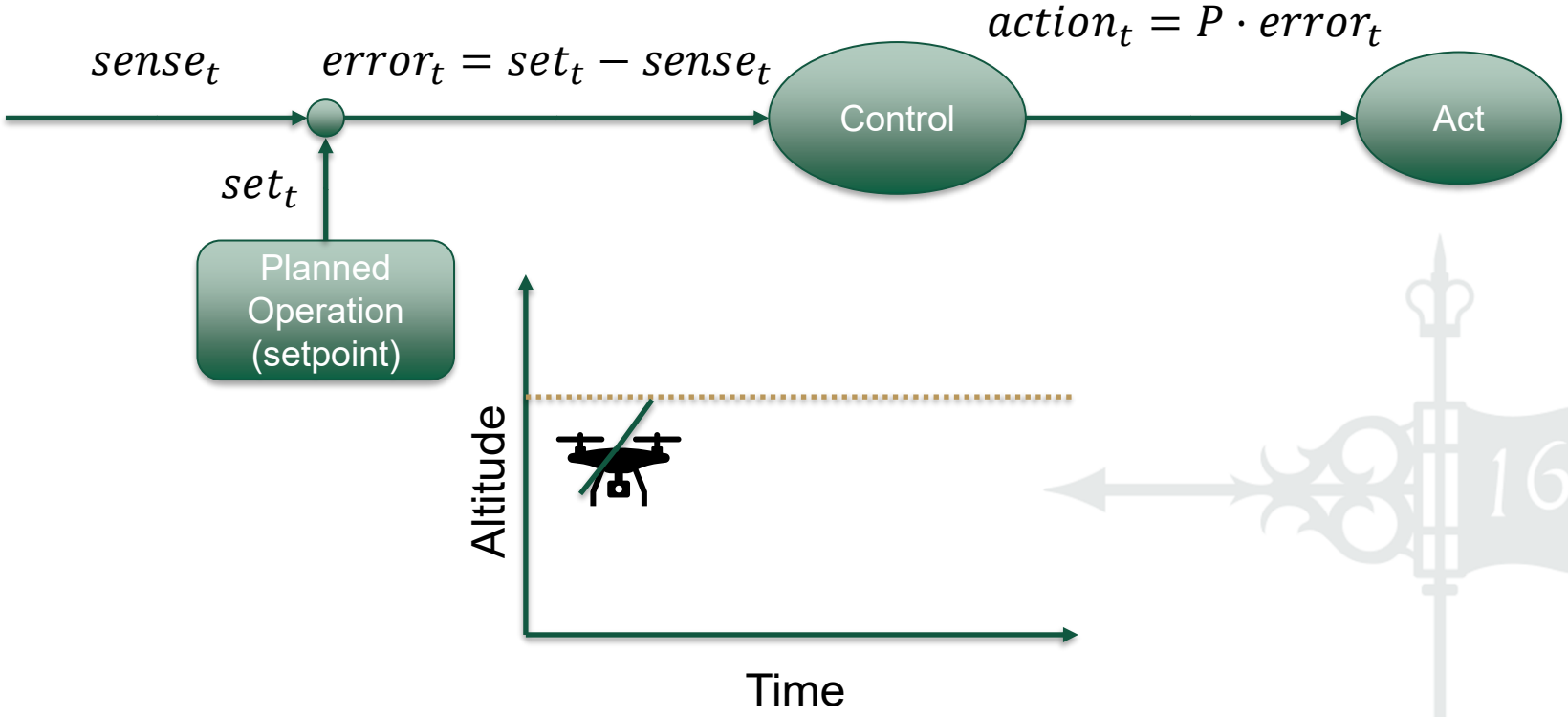
Closed-Loop: Proportional



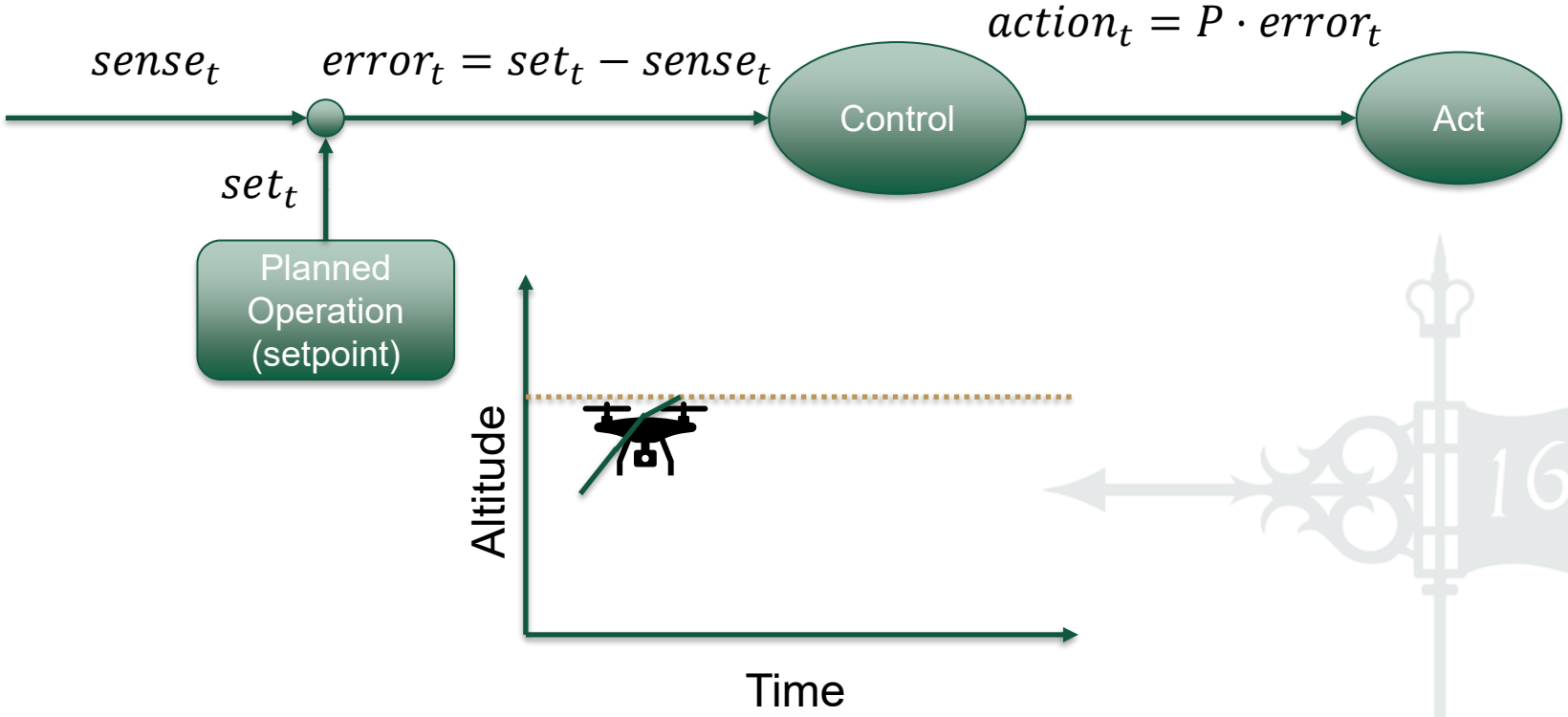
Closed-Loop: Proportional



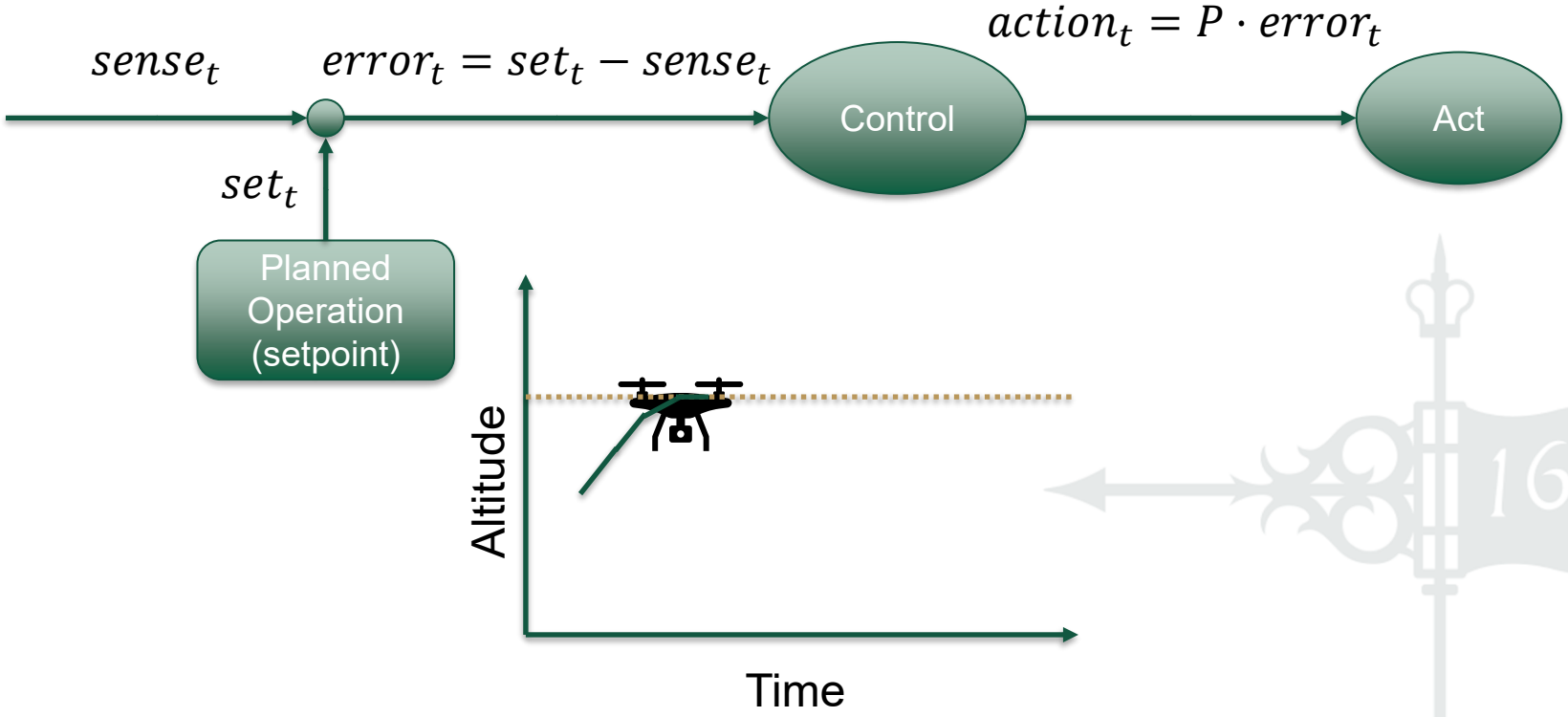
Closed-Loop: Proportional



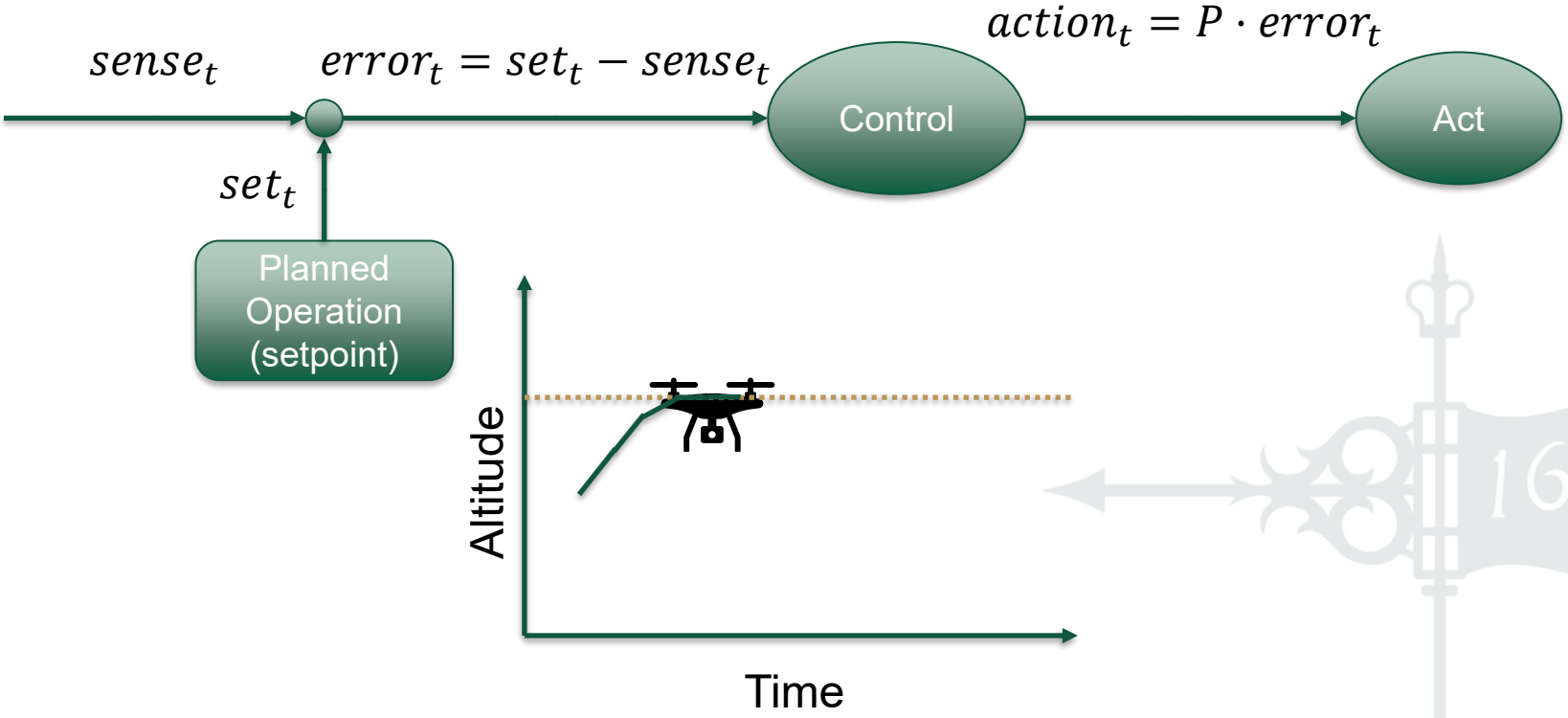
Closed-Loop: Proportional



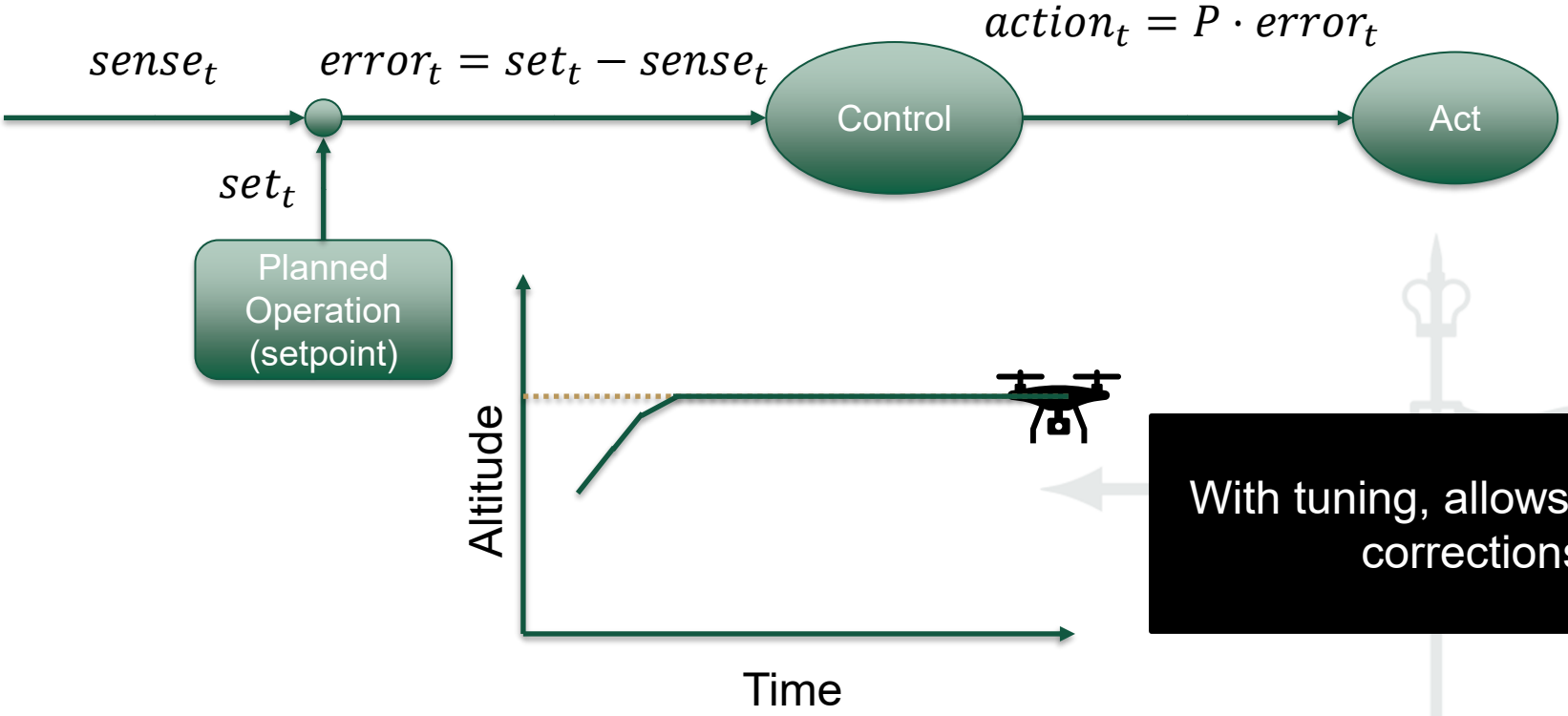
Closed-Loop: Proportional



Closed-Loop: Proportional

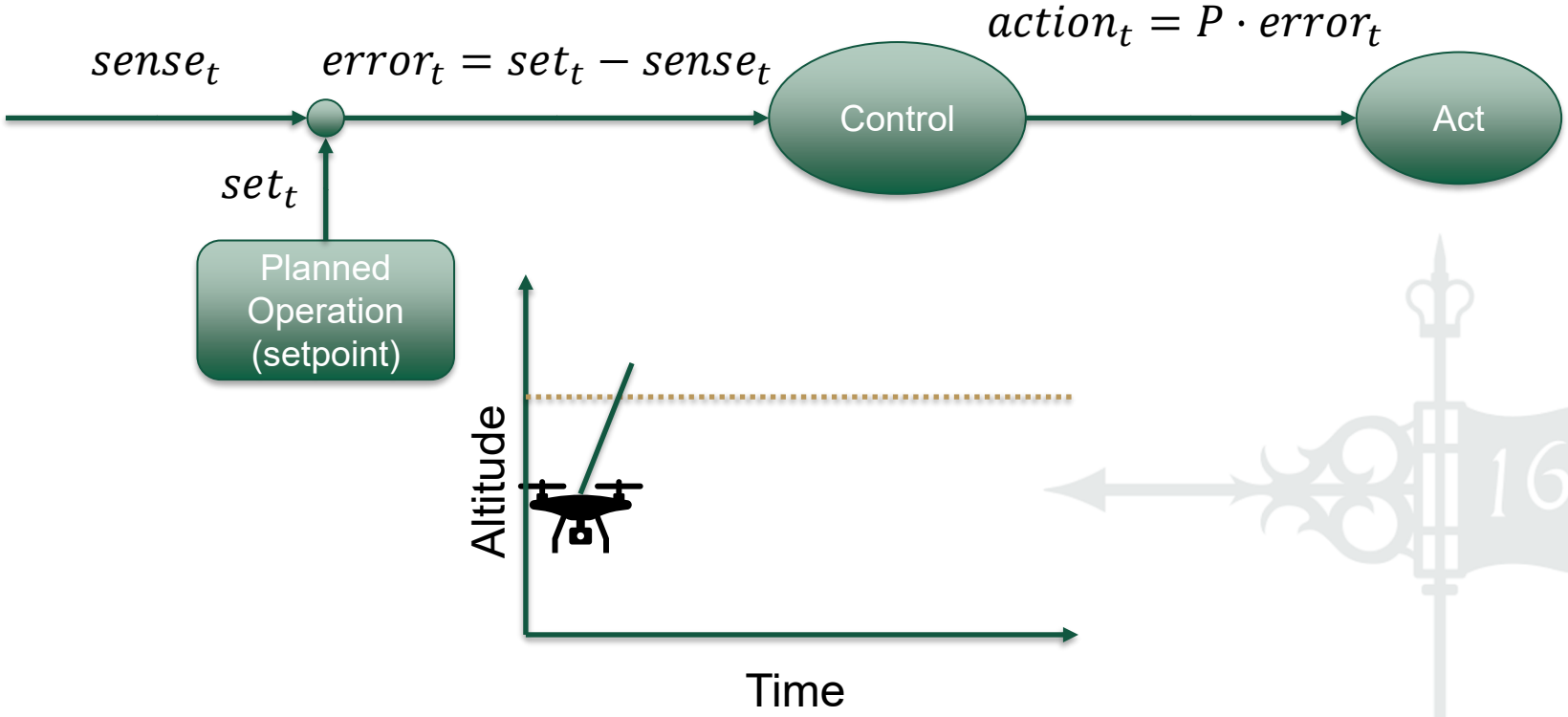


Closed-Loop: Proportional

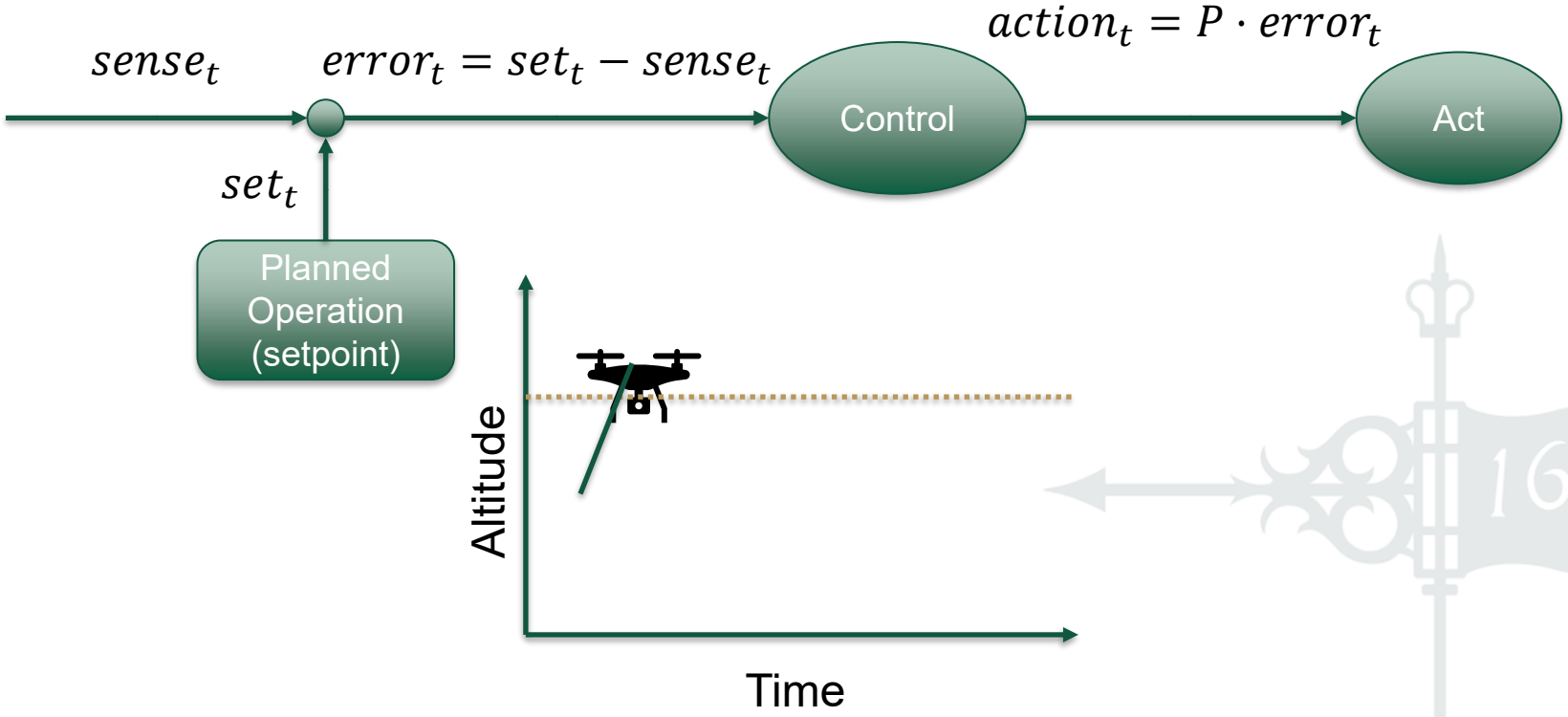


With tuning, allows for quick corrections

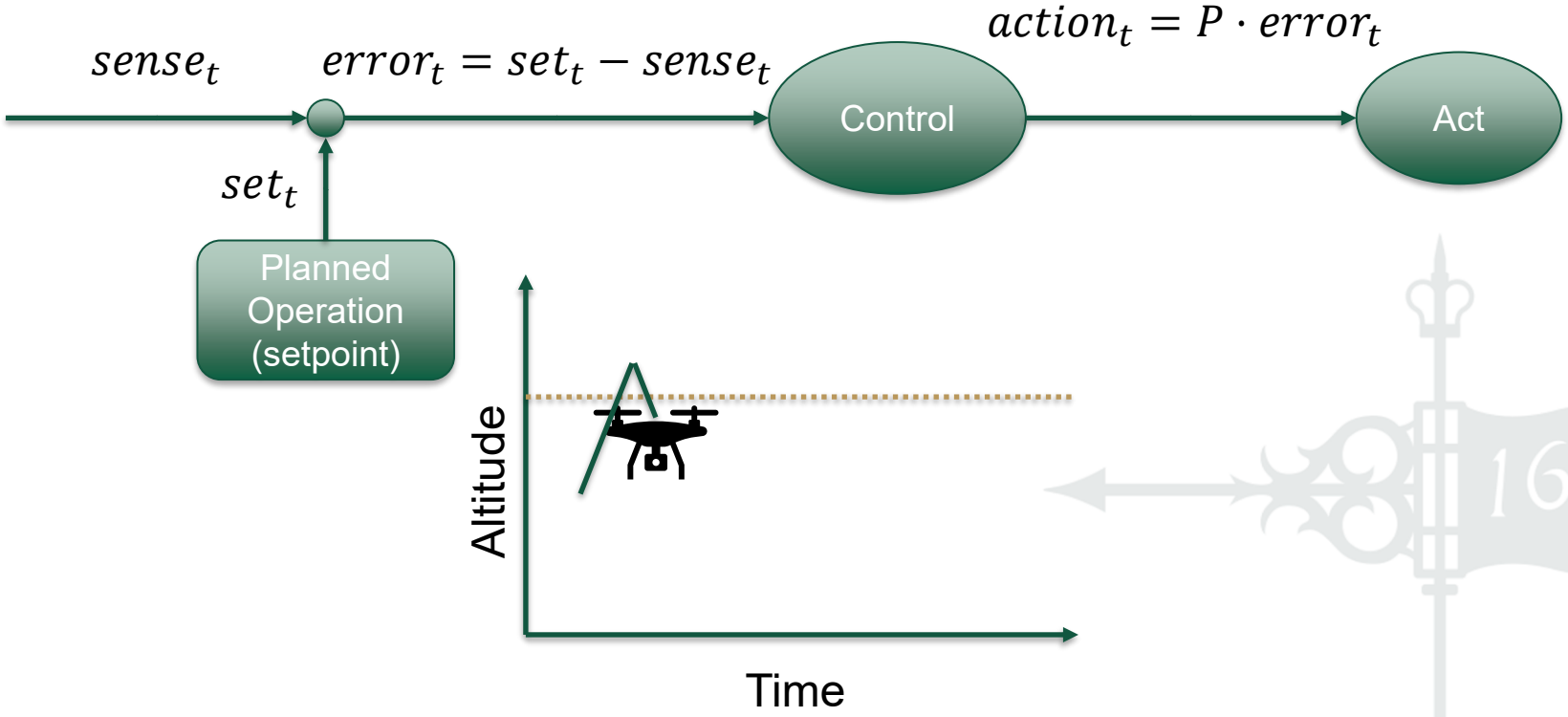
Proportional (bad tuning)



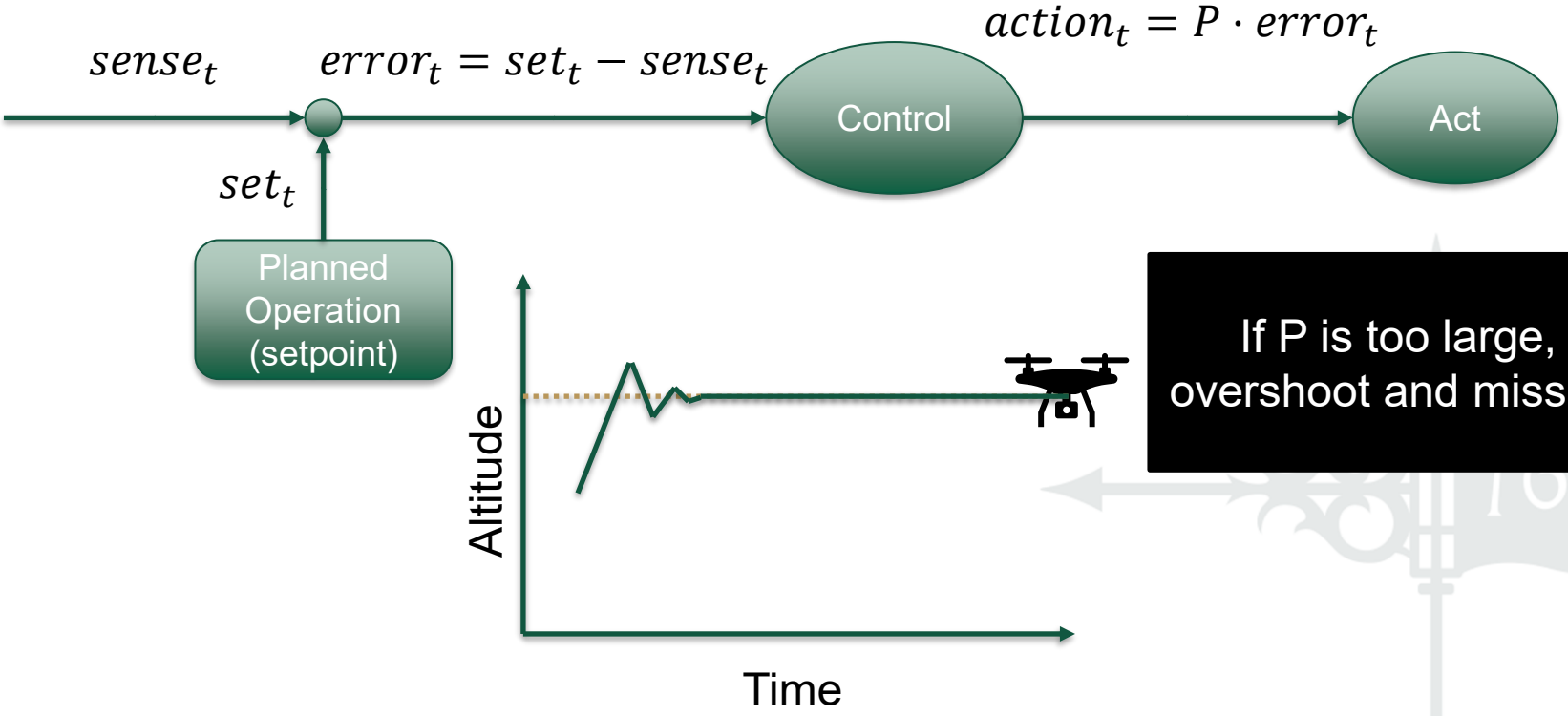
Proportional (bad tuning)



Proportional (bad tuning)

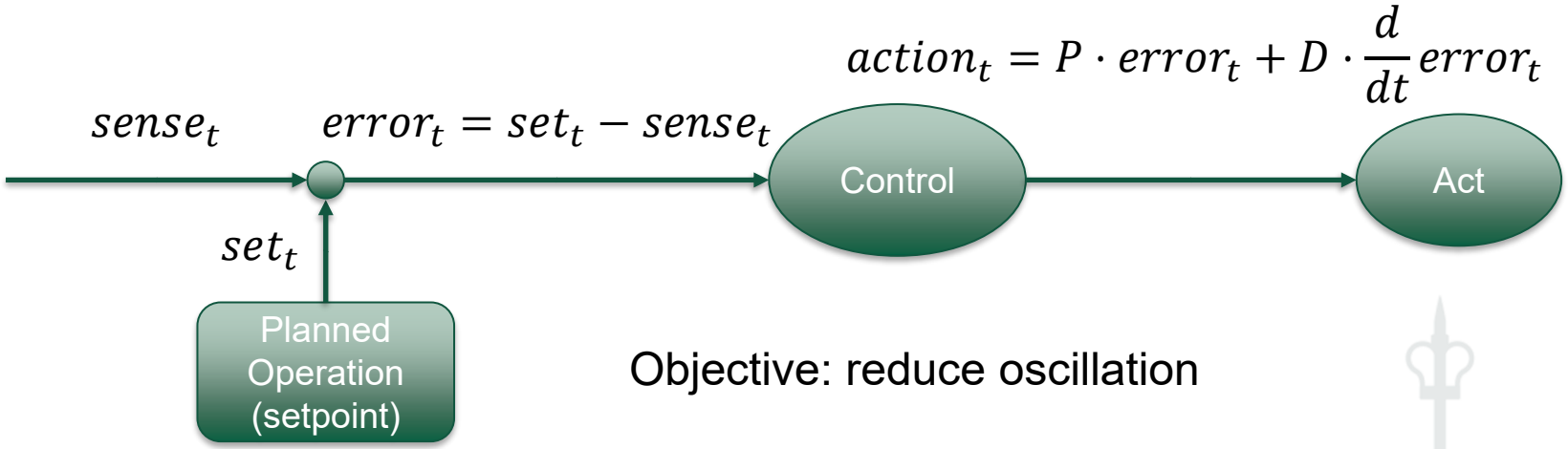


Proportional (bad tuning)



If P is too large, we will overshoot and miss the target

Closed-Loop: Derivative

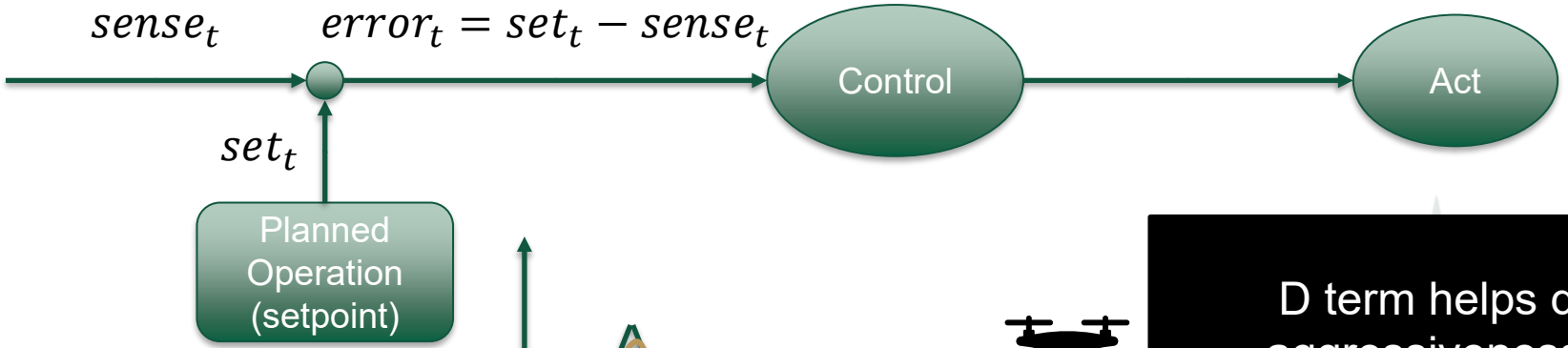


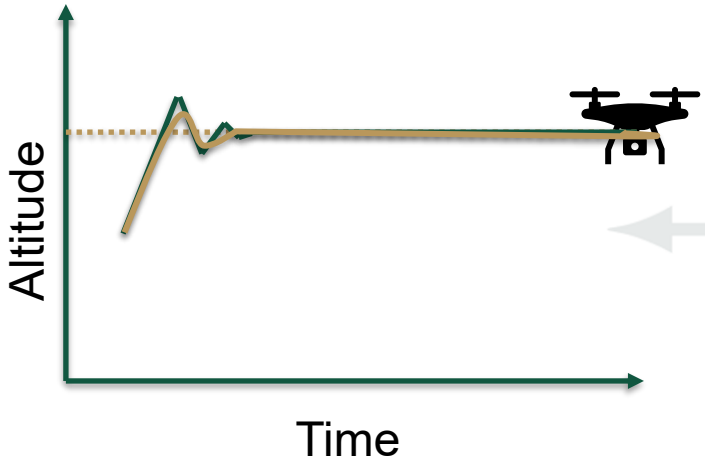
Objective: reduce oscillation

- Adjust output based on rate of output change
  - If too slow, increase the output
  - If too fast, decrease the output

Closed-Loop: Derivative

$$action_t = P \cdot error_t + D \cdot (error_t - error_{t-1})$$

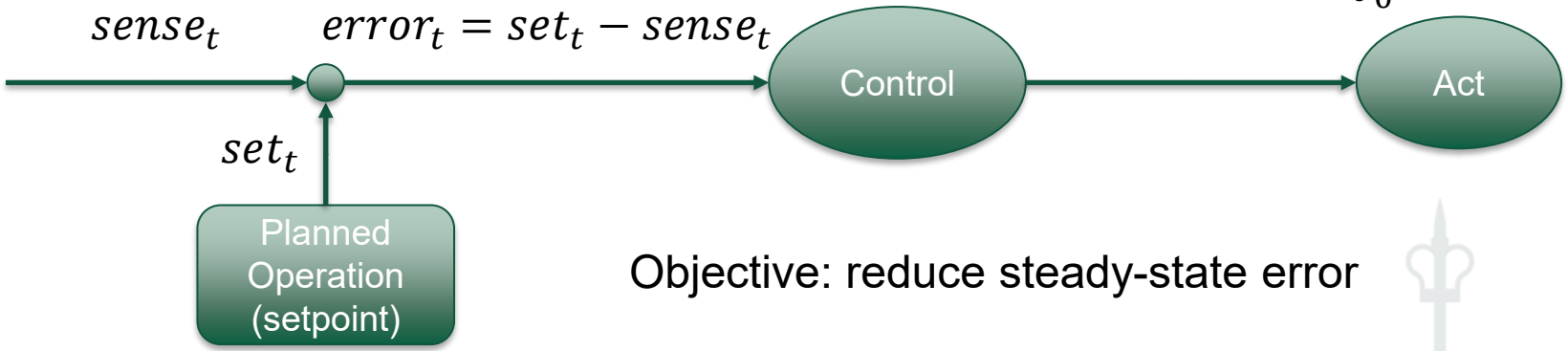




D term helps damp aggressiveness of P

Closed-Loop: Integral

$$action_t = P \cdot error_t + I \cdot \int_0^t error$$

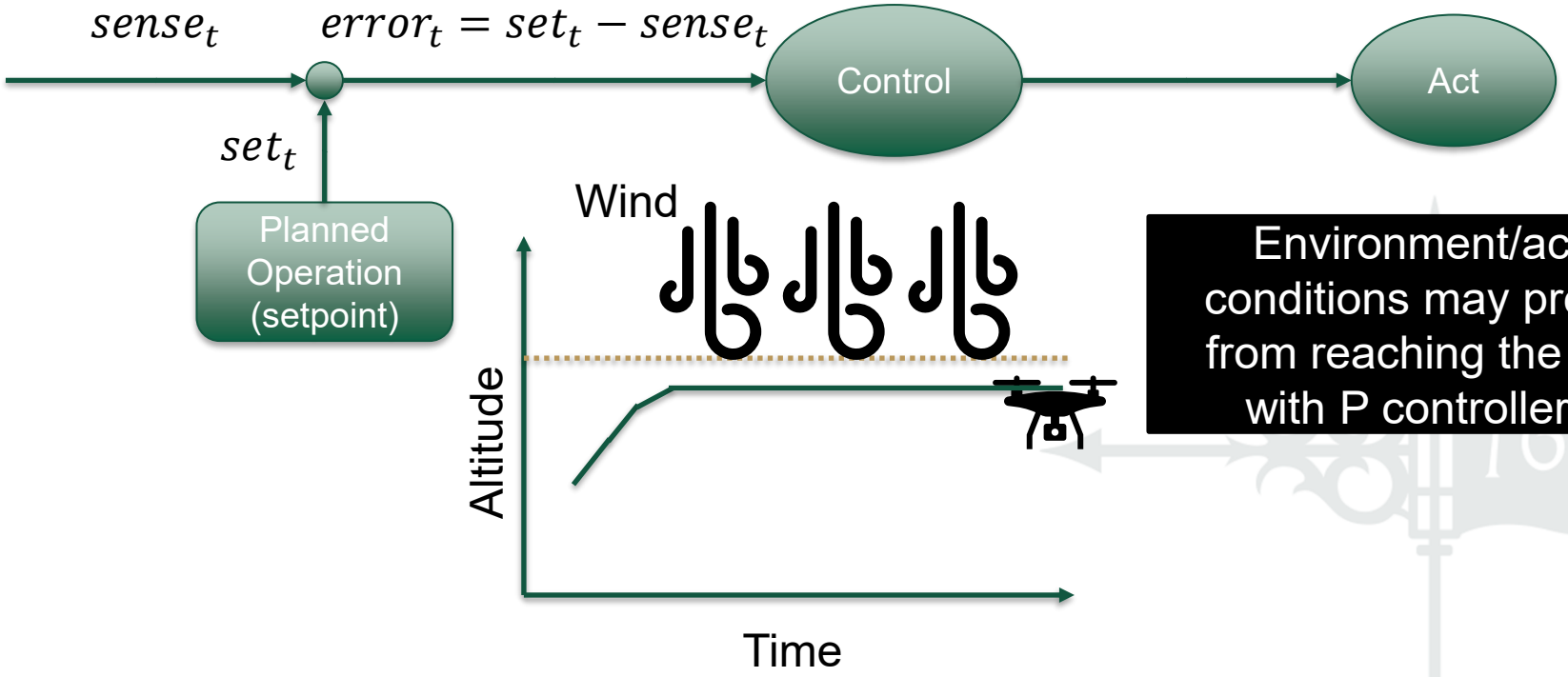


Objective: reduce steady-state error

- Add extra correction if error is not overcome

Closed-Loop: Integral

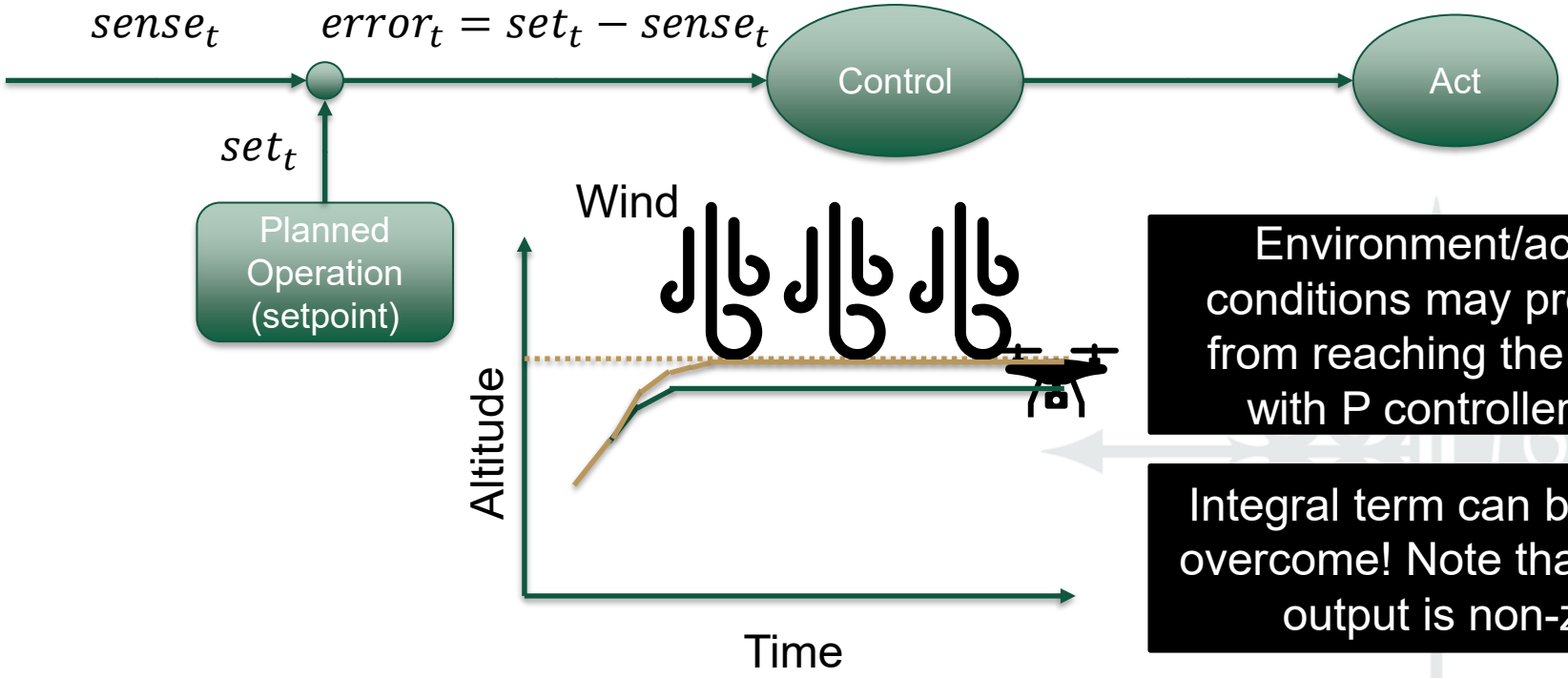
$$action_t = P \cdot error_t + I \cdot (error_0 + \cdots + error_t)$$



Environment/actuator conditions may prevent us from reaching the setpoint with P controller alone

Closed-Loop: Integral

$$action_t = P \cdot error_t + I \cdot (error_0 + \cdots + error_t)$$

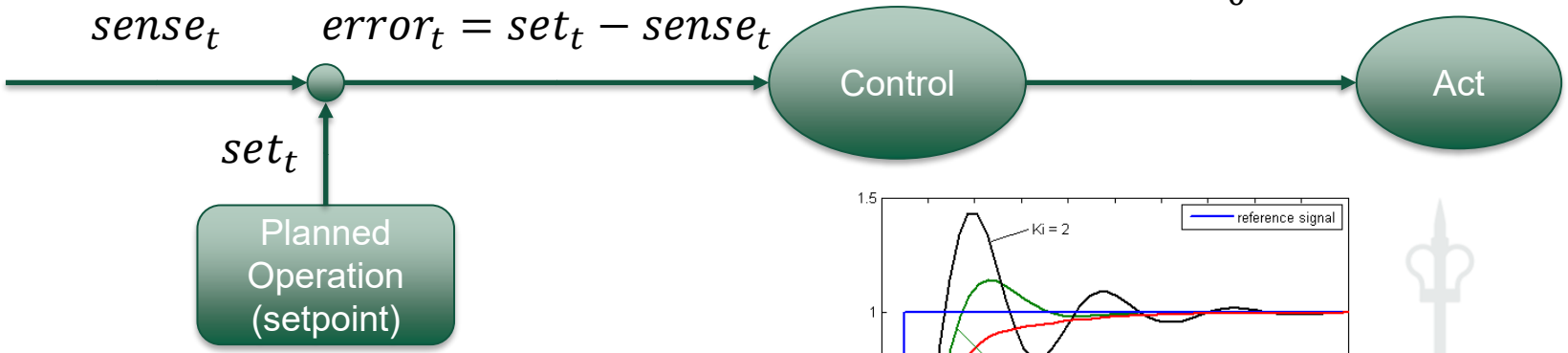


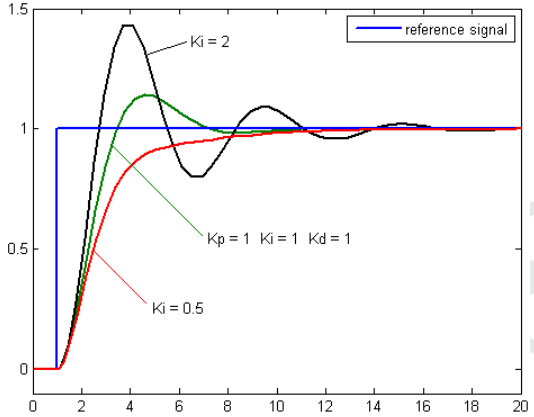
Environment/actuator conditions may prevent us from reaching the setpoint with P controller alone

Integral term can build up to overcome! Note that the final output is non-zero!

Putting it all together: PID

$$action_t = P \cdot error_t + I \cdot \int_0^t error + D \cdot \frac{d}{dt} error_t$$





PID Controller

```
set_point = input()
last_err, integral, last_time = 0, 0, 0;
def PID(measurement) {
  err = setpoint - measurement
  dt = time.time() - last_time
  last_time = time.time()
  deriv = (err - last_err) / dt
  integral += err*dt
  last_err = err
  return Kp*err + Kd*deriv + Ki*integral
}
```

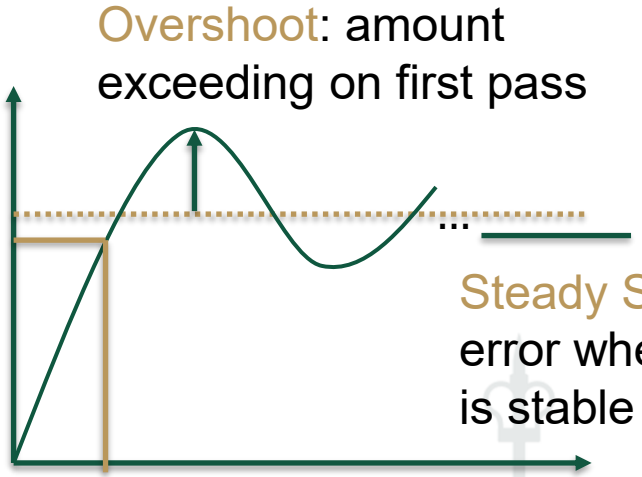
Missing:

- Definition of K constants
- Bounds on output
- Bounds/reset of integral

Tuning PID

- Depends on timing of loop
- Lots of methods

Rise Time: time to reach X% (e.g. 95%)



Increase	Rise Time	Overshoot	SS Error
Proportional	Decrease	Increase	Decrease
Integral	Decrease	Increase	Eliminate
Derivative		Decrease	

Controller Goals

- Stable
  - Error should converge
  - No oscillation (reach steady state)
- Performs to spec wrt rise/overshoot/ss error
- Robust
  - Stable & performant if plant/environment changes

Control

- Controllers:
  - Respond to physical phenomenon
  - Abstract away physics from response
- Closed-Loop = feedback
- PID
  - Extremely common, solve many problems
  - Picking K constants and timing is hard!