CISC 1003 - EXPLORING ROBOTICS

ROBOT CONTROL

CISC1003

Robot Control

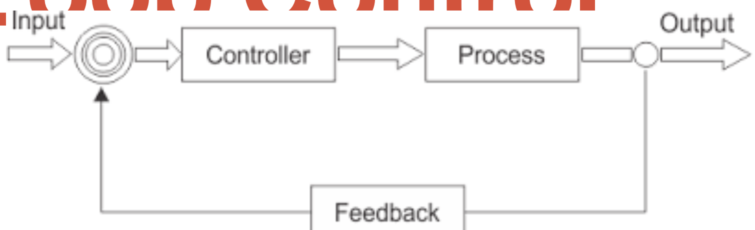
- So far we talked about the robot bodies
  - Sensors, effectors, etc.
- But what about their brains?
  - How do robots make decisions?

Open Loop Control

Open Loop Control System

- Sends commands to make a robot perform some movement
  - without attempting to check if it is doing things properly.
- For example: a rover on Mars being told by a human operator to go forward 1 meter.
  - If the wheels get dirt in them or hit a rock the robot won’t move straight.

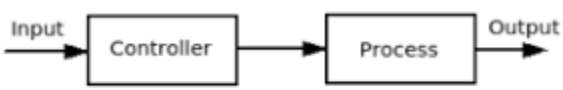
Feedback or Closed Loop Control



- ***Feedback control:***
  - A means of getting a system (a robot) to achieve and maintain a desired state
    - State is usually called the *set point*
  - Achieved by continuously comparing its current state with its desired state.
- ***Feedback*** refers to the information sent back
  - literally "fed back," into the system's controller.

Control Mechanisms

- Open Loop Control
  - Very few automatic controls or feedback system

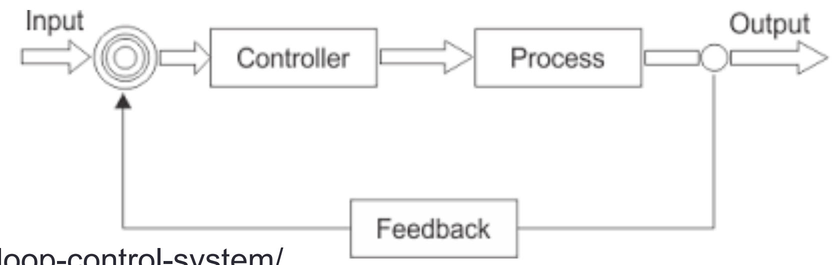


Open Loop Control System

https://www.marinerdesk.com/control-system/

Control Mechanisms

- Closed Loop Control
  - Has one or more feedback loops
  - Provides more accurate control of the process
    - Monitoring its output and “feeding” some of it back
    - compare the actual output with the desired output
      - Reduce the error
      - If required, bring the output of the system back to the original or desired response



https://www.electrical4u.com/control-system-closed-loop-open-loop-control-system/

Goals

- **Goal state**: the *desired state* of the system
  - where the system wants to be.
- In AI, there are two types of goals:
  - Maintenance and achievement goals

http://career-intelligence.com/get-what-you-want-with-smart-goals/

Goals

- ***Maintenance goals***:
  - Needed to reach a goal state
  - This will require ongoing active work on the part of the system.
  - Example Keeping a biped robot balanced
    - This is a maintenance goal.

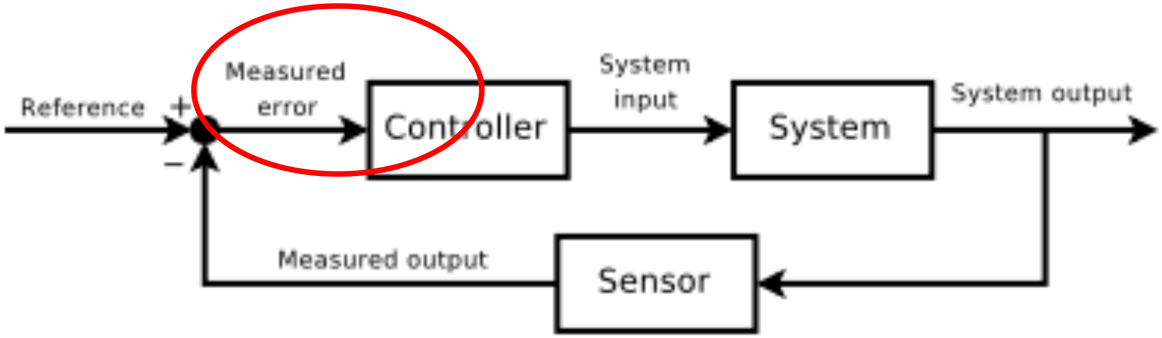
http://career-intelligence.com/get-what-you-want-with-smart-goals/

Goals

- ***Achievement goals***: states the system tries to reach
  - such as a particular location, perhaps the end of a maze
  - Once the system is there, it is done

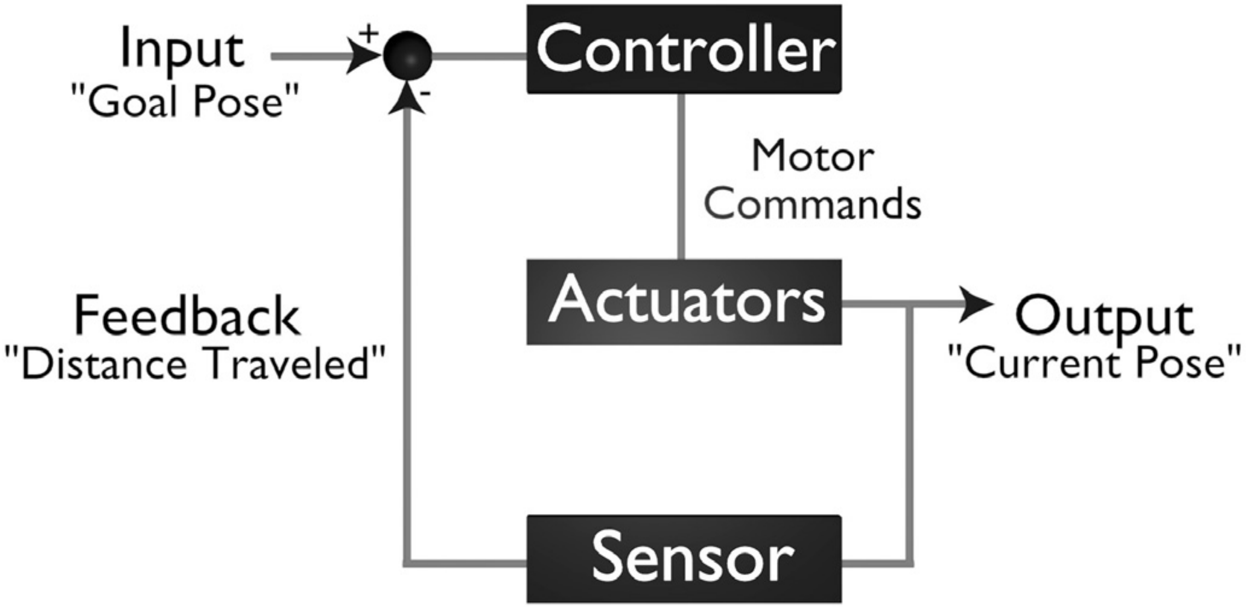
Errors

- *Error:* The difference between the current state and the goal state of a system.
  - The control system is designed to minimize error.
- Feedback control calculates the error in order to help the robot reach the goal.
  - When the error is zero (or small enough), the goal state is reached.



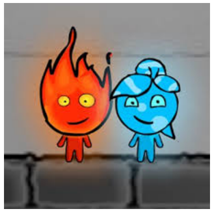
https://en.wikipedia.org/wiki/Control_theory

Feedback System



Feedback Example

- Real-world example of error and feedback:
  - The “hot and cold” game:
    - Player has to find or guess some hidden object (goal)
    - Participants help by saying things like “You’re getting warmer, hotter, colder, freezing” etc. (feedback)



http://www.safefood.eu/Start/Your-Tools/Bring-Back-Play/Hot-and-Cold.aspx
https://apkpure.com/hot-boy-and-cold-girl/com.frisbee.tpaf.stor

Feedback Example

- Imagine a overly simplified version of the same game
  - Users tell you only “You are there, you win!” or “Nope, you are not there.”
    - In that case, what they are telling you is only if the error is **zero** or **non-zero (**if you are at the goal state or not).
    - This is not very much information
      - It does not help you figure out which way to go in order to get closer to the goal, to minimize the error.

Feedback Example (cont.)

- In the normal version of the game, you are being given the ***direction of the error***
  - By being told “hot” or “cold”
  - Helps minimize the error and getting close to the goal.

Feedback Example (cont.)

- When the system knows how far off it is from the goal, it knows the ***magnitude of error***
  - The distance to the goal state.
  - In the “hot and cold” game:
    - Gradations of input used to indicate the distance from (or closeness to) the goal object.
      - freezing, chilled, cool, warm, and so on

Feedback Control Robot - Example

- *How would you write a controller for a wall-following robot using feedback control?*

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

Feedback Control Robot – Example (cont.)

- The first step is to consider the goal of the task.
- In wall-following, the **goal state** is a particular distance, or range of distances, from a wall.
  - This is a **maintenance goal**, since wall-following involves keeping that distance over time.

Feedback Control Robot – Example (cont.)

- What is the error in case of wall-following?
  - The difference between the desired distance from the wall and the actual distance at any time.
- Whenever the robot is at the desired distance (or range of distances), it is in the goal state.
  - Otherwise, it is not.

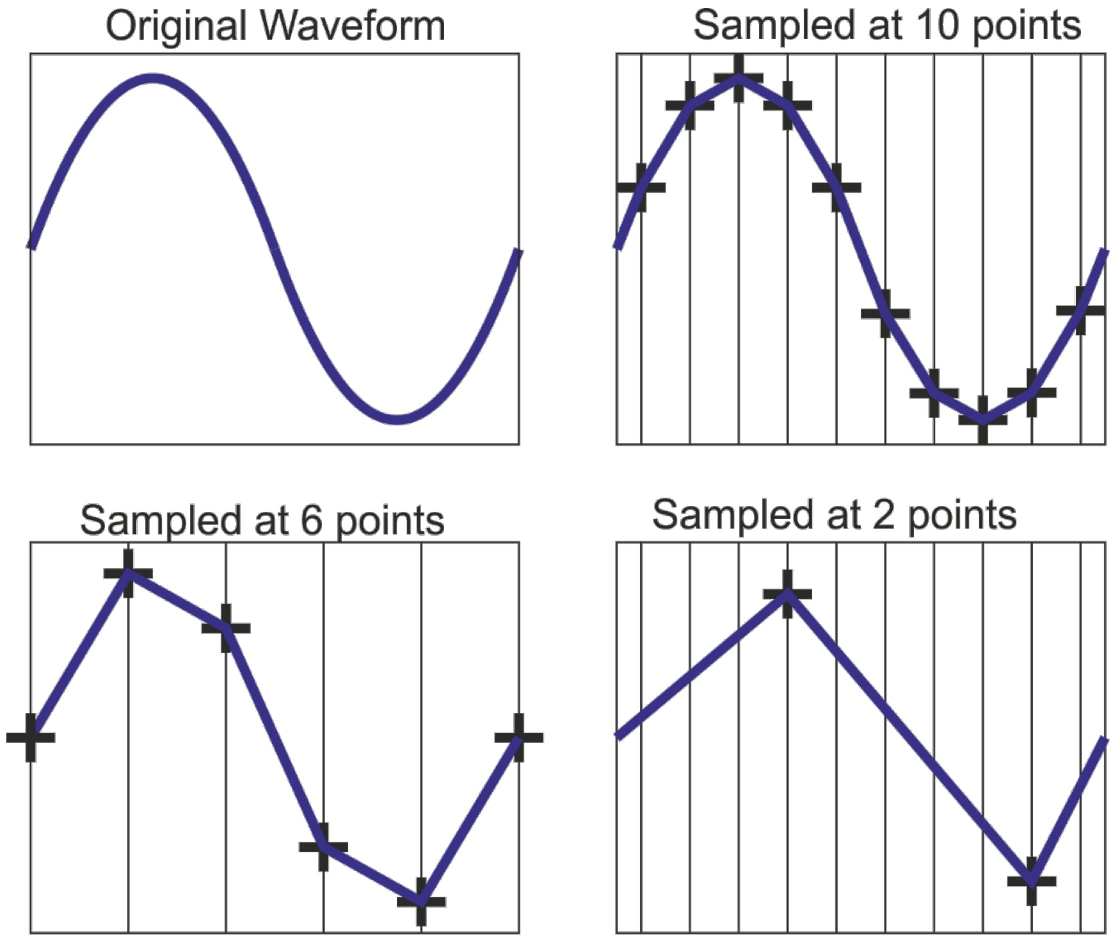
Feedback Control Robot – Example (cont.)

- *What sensor(s) would you use for a wall-following robot?*
  - *what information would they provide?*

Feedback Control Robot – Example (cont.)

- ***Sampling Rate:***
  - The rate with which new distance-to-wall is sensed and computed
  - Controls the interval at which sensor events are sent to your application
  - Sensors will have a maximum sampling rate
    - Part of their specifications

Sampling Rate



https://www.reddit.com/r/WeAreTheMusicMakers/comments/3x4hl2/what_is_sample_rate_and_how_to_use_it_correctly/

Feedback Control Robot – Example (cont.)

- Whatever sensor is used, assume that it provides the information to compute *distance-to-wall*.
- Consider the following controller:
  - If distance-to-wall is in the desired range,
    - keep moving forward.
  - If distance-to-wall is larger than desired,
    - turn toward the wall,
  - If distance-to-wall is smaller than desired,
    - turn away from the wall

Feedback Control Robot – Example (cont.)

- Given the previous controller algorithm, what will robot do?
  - The robot's behavior will keep it moving and wiggle back and forth as it moves along.
- How much switching back and forth will it do?
  - That depends on two parameters:
    - How often the error is computed.
    - How much of a correction (turn) is made each time.

Feedback Control Robot – Example (cont.)

Consider the following controller:

- If distance-to-wall is exactly as desired,
  - keep going.
- If distance-to-wall is larger than desired,
    - turn by 45 degrees toward the wall,
  - else
    - turn by 45 degrees away from the wall.

Feedback Control Robot – Example (cont.)

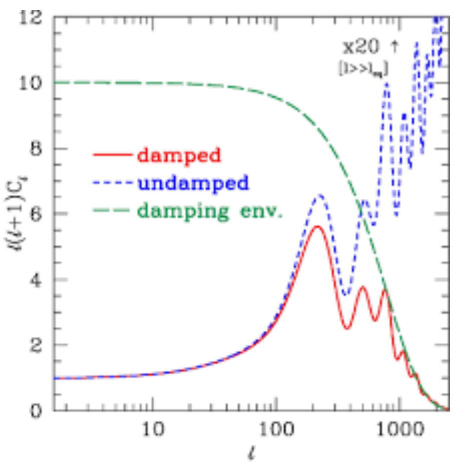
- Given this algorithm, what will robot do?
  - It oscillates a great deal
    - Rarely if ever reaches the desired distance before getting too close to or too far from the wall.
  - In general, the behavior of any simple feedback system oscillates around the desired state.
    - Therefore, the robot oscillates around the desired distance from the wall
    - most of the time it is either too close or too far away.

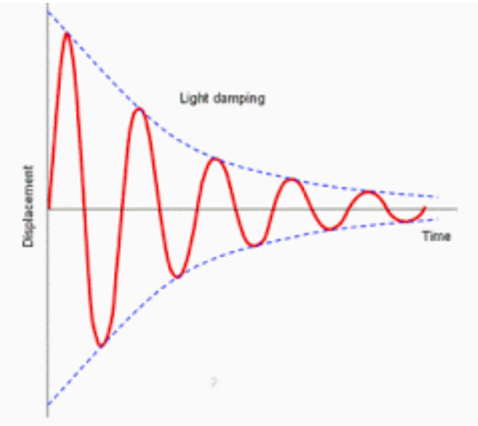
How can we decrease this oscillation?

- There are a few things we can do:
  - Compute the error often, so the robot can turn often rather than rarely.
  - Adjust the turning angle so the robot turns by small rather than large angles.
  - Find the optimal range of distances that defines the robot's goal.

Decreasing Oscillations (cont.)

- *Damping* refers to the process of systematically decreasing oscillations.
  - A system is properly *damped* if it does not oscillate out of control.





http://background.uchicago.edu/~whu/physics/damping.html
https://www.patana.ac.th/secondary/science/anrophysics/ntopic4/commentary.htm

Decreasing Oscillations (cont.)

- Motor response to speed commands plays a key part in control, wear and tear on the gears.
  - The faster the response, the better the control

https://www.indiamart.com/proddetail/10-hp-electric-motor-16438837788.html

Decreasing Oscillations (cont.)

- *Actuator uncertainty* makes it impossible for a robot to know the exact outcome of an action ahead of time
  - similar to human actions and responses
  - even for simple actions
    - such as "Go forward three feet."

https://www.robotshop.com/blog/en/how-do-i-choose-appropriate-actuators-6903

Feedback Control

The three most used types feedback control are:

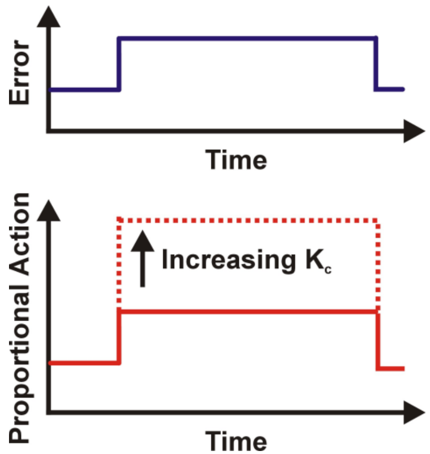
- Proportional control (P)
- Proportional Derivative control (PD)
- Proportional Integral Derivative control (PID)

Proportional Control

- System responds in proportion to the error
  - using both the direction and the magnitude of the error.
- For wall-following robot:
  - Use distance-to-wall to determine the angle and distance and/or speed with which the robot would turn.

.

http://blog.opticontrols.com/archives/344



Proportional Control

- *Gains*: The parameters that determine the magnitude of the system's response
  - In control theory
- *Damping:*
  - The process of systematically decreasing oscillations.
  - A properly *damped system* => does not oscillate out of control.

Derivative Control

- The controller corrects for the momentum of the system as it approaches the desired state.
  - Relative to the rate the error changes
- When the system is close to the desired state, it needs to be controlled differently than when it is far from it
  - Otherwise, the controller's correction will carry the system beyond the desired state
    - cause oscillations

Derivative Control

- Best estimate the future trend of the error, based on the current rate of change

Control - Example

- Robot moving towards a target
- Proportional control:
  - Speed changes based on distance from target
- Derivative control:
  - Speed changes based on current speed of robot
    - The faster it is, the more we will slow it down

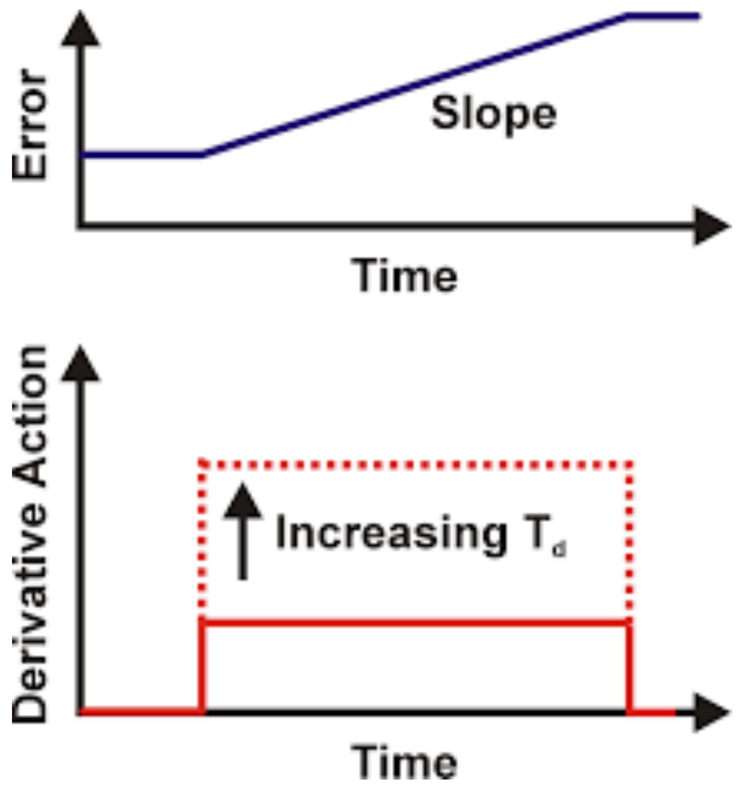
Derivative Control

- momentum = mass $*$ velocity
  - the faster you move and/or the bigger you are, the more momentum you have
  - => Control momentum by controlling velocity

Derivative Control

- For our wall-following robot:
  - A derivative controller would slow the robot down
  - Decrease the angle of its turning
    - As distance from the wall gets closer to the desired state
      - Desired state = optimal distance to the wall.

Derivative Control



http://blog.opticontrols.com/archives/344

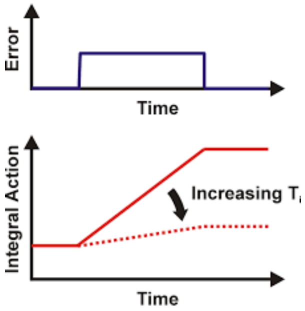
Integral Control

- The system integrates (sums up) these incremental errors over time
  - When reaching a predetermined threshold the system does something to compensate/correct.
    - Once the cumulative error gets large enough
- Example: A robot lawn mower with an error when turning to cut sections of grass.
  - With a system to detect the error, it can compensate for it over time.

Integral Control

- Account for past behavior of system
- Integrates the error over time

Integral Control



http://blog.opticontrols.com/archives/344

Control - Example

- Robot moving towards a target
- Proportional control:
  - Speed changes based on distance from target
- Derivative control:
  - Speed changes based on current speed of robot
    - The faster it is, the more we will slow it down

Control - Example

- Integral control:
    - Speed change based on overall distance from wall
        - If distance was small for a long time => small change required
            - May be good for a wall-following program
        - If distance changes drastically, then overall integral will minimize changes
            - The sum of changes changes less in time
            - Reduce the need to make constant changes to the robot speed

Derivative Control Systems

- Proportional Derivative control (PD)
  - Applied in most industrial plants
  - Extremely useful for process control.
- Proportional Integral Derivative control (PID)
  - A combination of proportional *P*, integral *I*, and derivative *D* control terms.

What Can the Robot Represent?

- There are numerous aspects that a robot can represent and model
  - and numerous ways in which it can do it.

What Can the Robot Represent?

- The robot can represent information about:
  - Self: Battery life, physical limits
  - Environment: navigable spaces, structures
  - Objects, people, other robots:
    - detectable things in the world
  - Actions: outcomes of specific actions in environment
  - Task: what needs to be done

Control Architectures

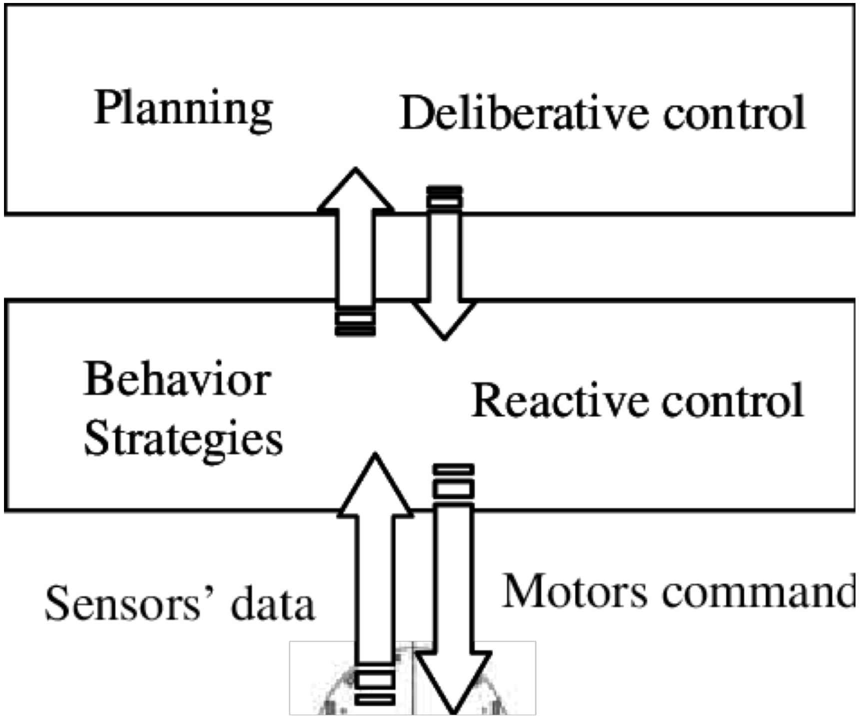
- The controller functions as the robot’s brains
  - So robot can be autonomous, achieve its goals.
- The robot can sense multiple things at once.
  - The controller decides what the robot needs to observe.

Control Architectures

- Not all control architectures are the same
- Different capabilities allow for different robot functionality
  - Regardless of which language is used to program a robot, architecture will determine functionality
    - Programming language is just a tool to program robot

Control Architectures

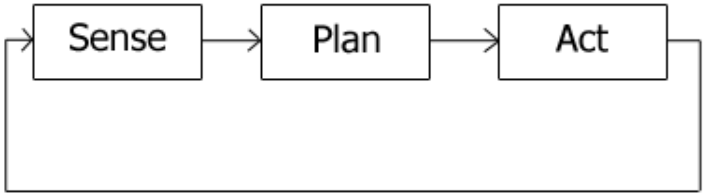
- May include two components:
  - Deliberative Control
  - Reactive Control



https://www.researchgate.net/figure/Hybrid-robot-architecture_fig1_304540754

Deliberative Control

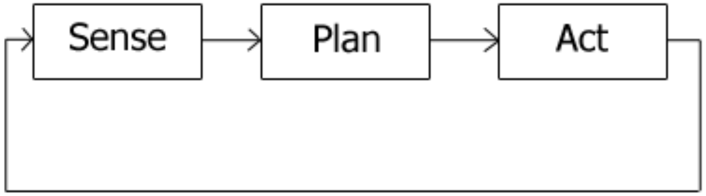
- *Planning:*
  - The process of looking ahead at the outcomes of the possible actions
  - searching for the sequence of actions that will reach the desired goal.
- *Search:*
  - An inherent part of planning.
  - involves looking through the available representation "in search of" the goal state.



https://en.wikipedia.org/wiki/Robotic_paradigm

Deliberative Control (cont.)

- Deliberative, planner-based architectures involve three steps that need to be performed in sequence:
  1. Sensing (S)
  2. Planning (P)
  3. Acting (A), executing the plan.



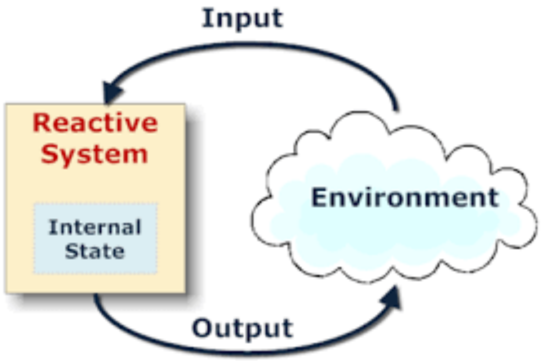
  - These steps need to be performed in sequence

Reactive Control

- Reactive rules are similar to *reflexes*
  - innate responses that do not involve any thinking.
- How to design a good reactive system?
  - Keep it simple and straightforward
  - Define robot states:
    - Situations that can be detected by the robot's sensors
  - Each unique state should trigger only one unique robot action

Reactive Control

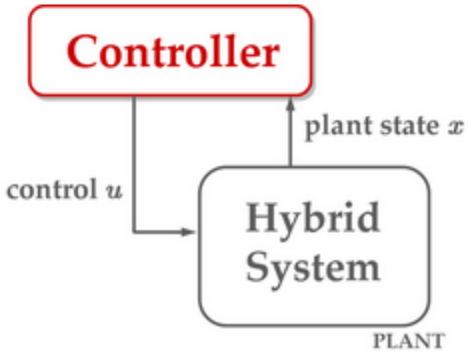
- Reactive rules must support parallelism
  - To handle checking multiple sensors.



http://www.reactive-systems.com/c-automatic-test-generation.html

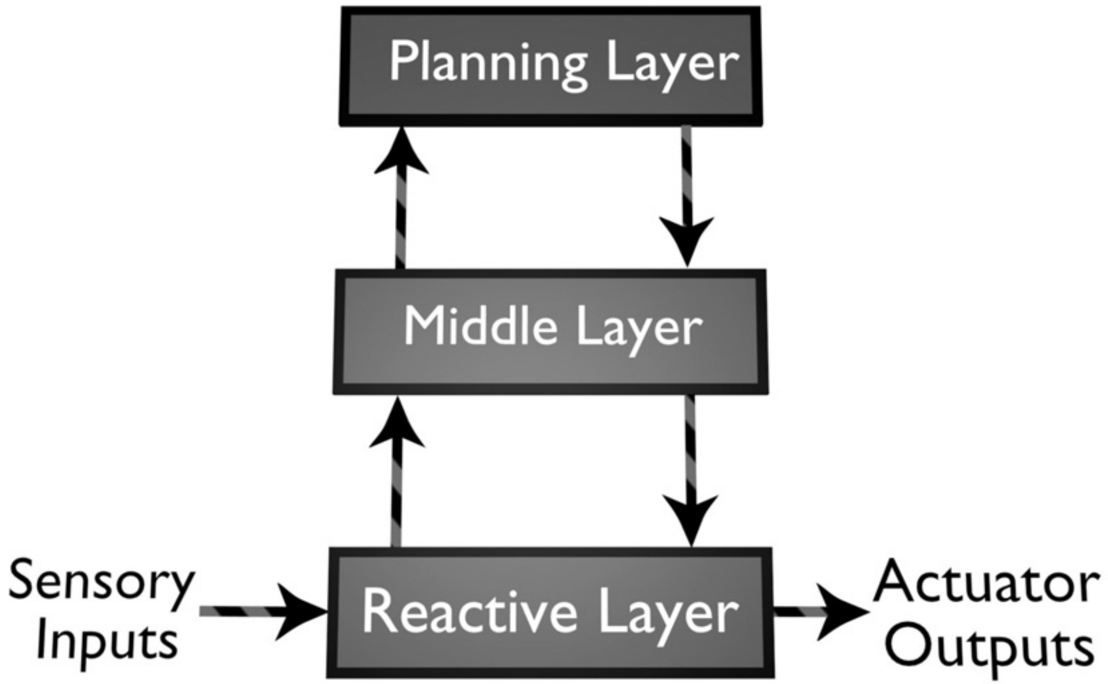
Hybrid Control

- Involves the combination of reactive and deliberative control
  - within a single robot control system
- Components:
  - Planner
    - Middle layer that links the layers togethe
      - (by issuing commands).
    - Reactive layer



http://people.ee.ethz.ch/~cohysys/control.php

Hybrid Control



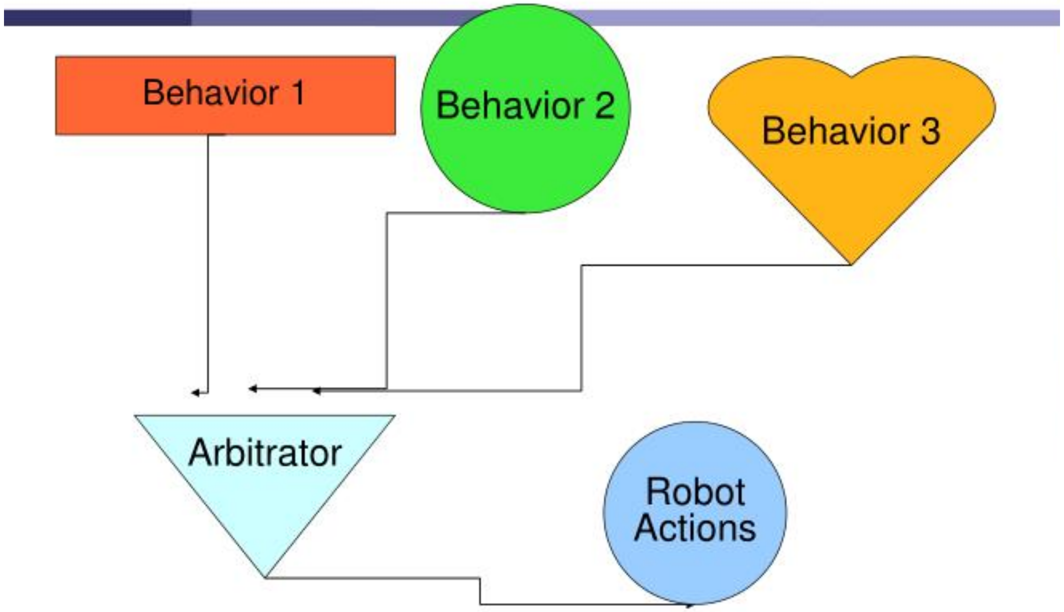
Behavior-Based Control

- *Behavior-based control* (BBC) involves the use of "behaviors" as modules for control.
- Behaviors achieve and/or maintain complex goals.
  - A *homing* behavior:
    - Achieves the goal of getting the robot to the home location.
  - A *wall-following* behavior:
    - maintains the goal of following a wall.

Behavior-Based Control

- Behaviors take time to execute and are not instantaneous.
- Requires constantly monitoring the sensors and other behavior status variables.

Behavior-Based Control



https://www.slideserve.com/mostyn/behavior-based-control


QUESTIONS

INTRODUCTION TO PROGRAMMING

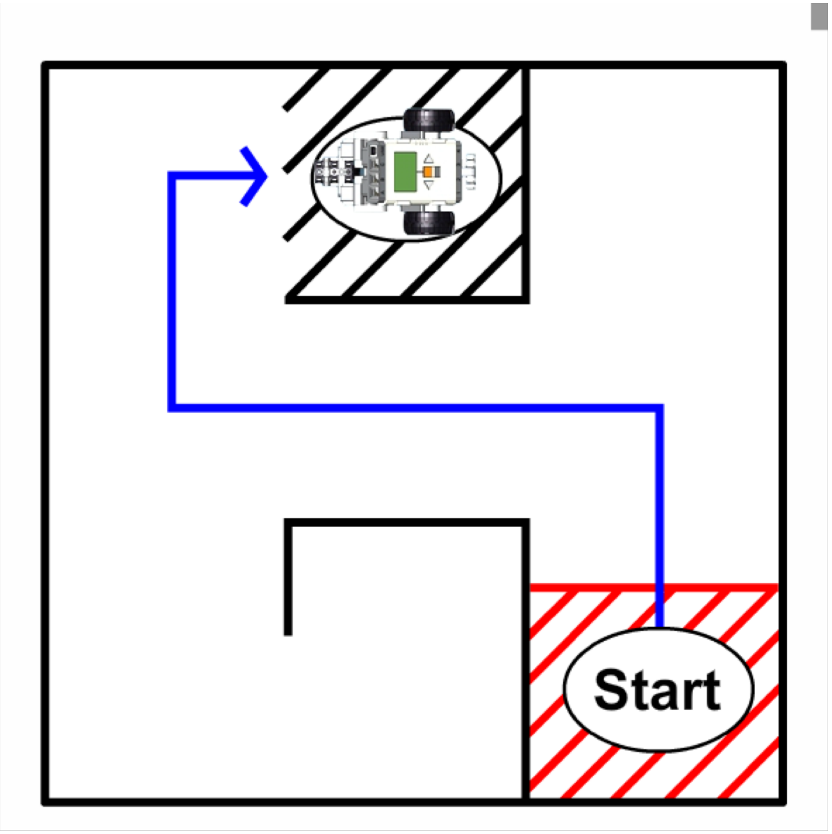
Task Planning, Pseudocode

Task Planning

- What is the problem?
  - Identify the tasks you need

Task Planning

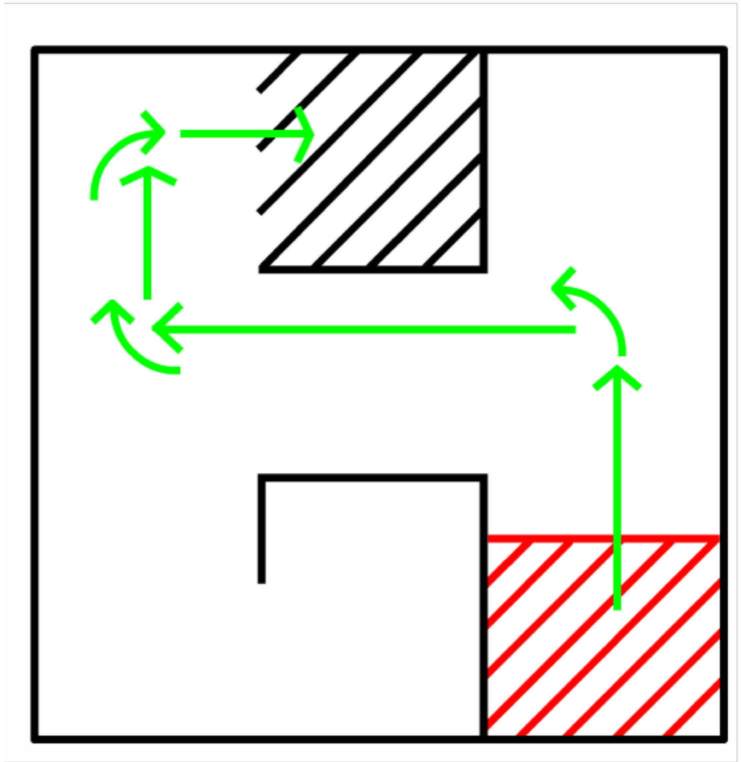
- Example: follow the path



http://cmra.rec.ri.cmu.edu/previews/nxt_products/nxt_video_trainer2/

Task Planning

- Break the main path into smaller paths:



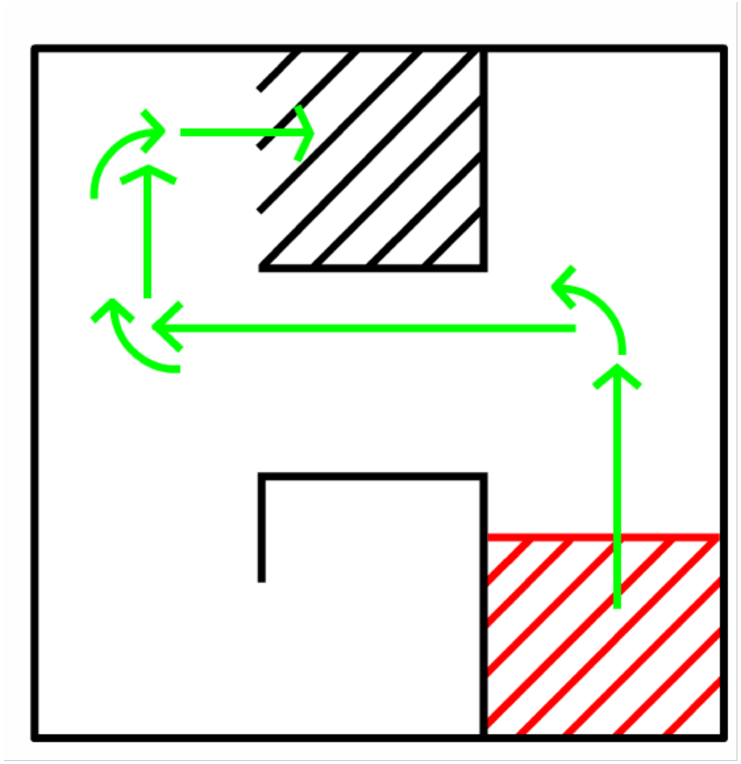
http://cmra.rec.ri.cmu.edu/previews/nxt_products/nxt_video_trainer2/

Task Planning

- Each of the smaller paths is a sub-task
- Write down the sequence of sub-tasks that is needed

Task Planning

- Follow the path:
  - Move forward
  - Turn left
  - Move forward
  - Turn right
  - Move forward
  - Turn right
  - Move forward



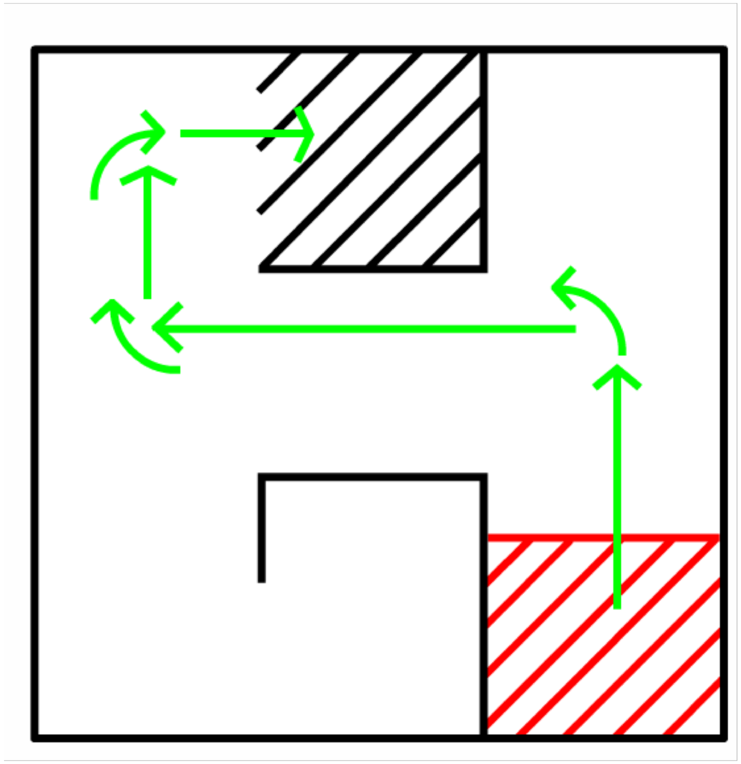
http://cmra.rec.ri.cmu.edu/previews/nxt_products/nxt_video_trainer2/

Task Planning

- Can we break these into smaller tasks?

Task Planning

- Follow the path:
  - Move forward
    - Left motor forward
    - Right motor forward
    - Wait 2 seconds
  - Turn left
    - Left motor reverse
    - Right motor forward
    - Wait 1 second
- Etc…



http://cmra.rec.ri.cmu.edu/previews/nxt_products/nxt_video_trainer2/

Pseudocode

- As we increase the level of details, we will reach commands we can express directly in programming language
- This is the plan the robot needs to follow
- The steps are written in English
  - So can be understood by the human programmer
- This is called *Pseudocode*

Pseudocode Example

```
task main()
{
 while ( touch sensor is not pressed )
 {
   Robot runs forward

   if (sonar detects object < 20 cm away)
   {
     Robot stops

     Robot turns right
   }
 }
}
```

http://cdn.robotc.net/pdfs/nxt/reference/hp_pseudo_flow.pdf


QUESTIONS

Lab time!

- Let’s work with our robots!