Introduction to Robotics

Manipulation and Programming

Unit 4: Motion Control

MOBILE ROBOT PART 3 – FINITE STATE MACHINE

DR. ERIC CHOU

IEEE SENIOR MEMBER

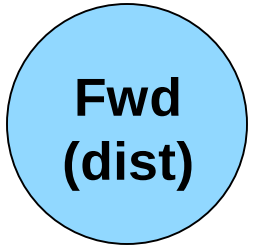
Objectives

- Control Design – Finite State Machine
- Case Study of a Finite State Machine Control

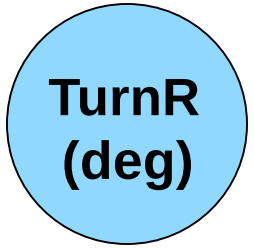
Finite State Machine

SECTION 1

Finite State Machines offer another alternative for combining behaviors

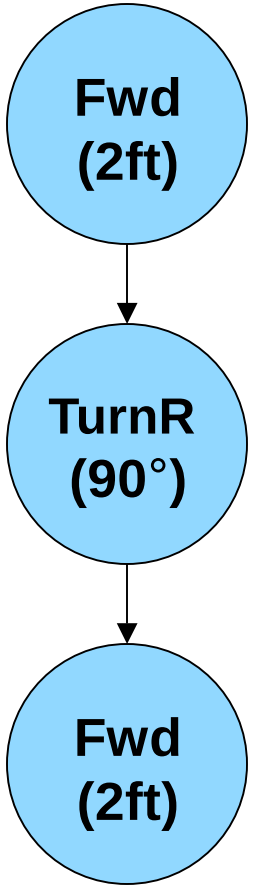


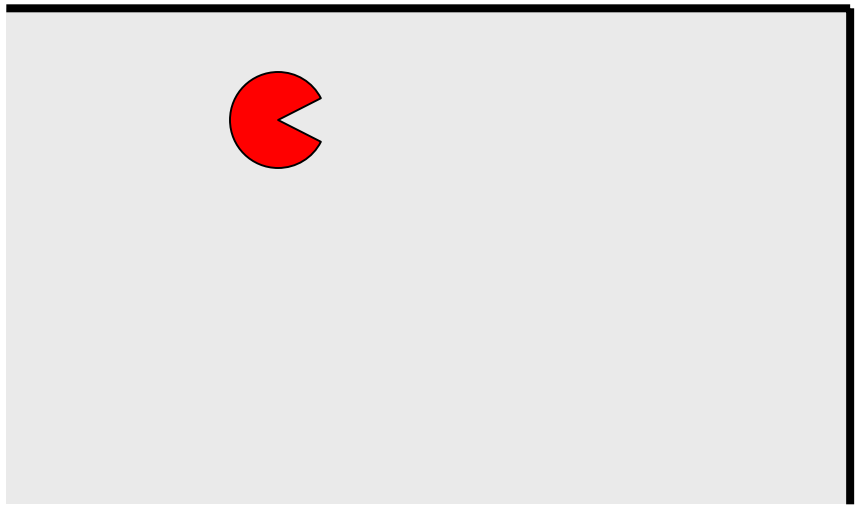
Fwd behavior moves robot straight forward a given distance



TurnR behavior turns robot to the right a given number of degrees

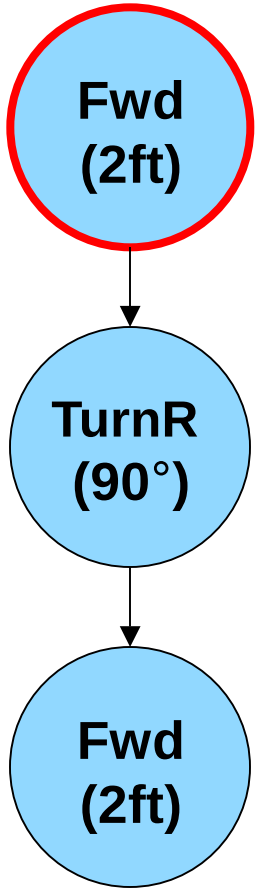
Finite State Machines offer another alternative for combining behaviors

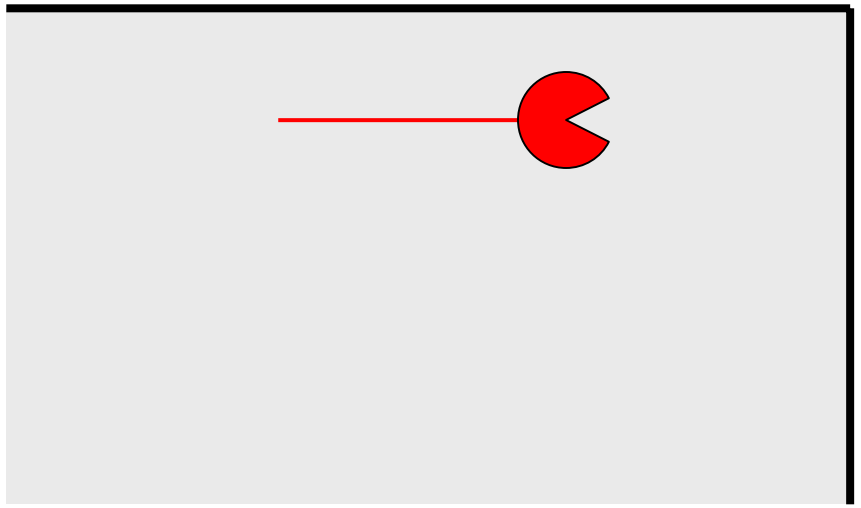




Each state is just a behavior and we can easily link them together to create an open loop control system

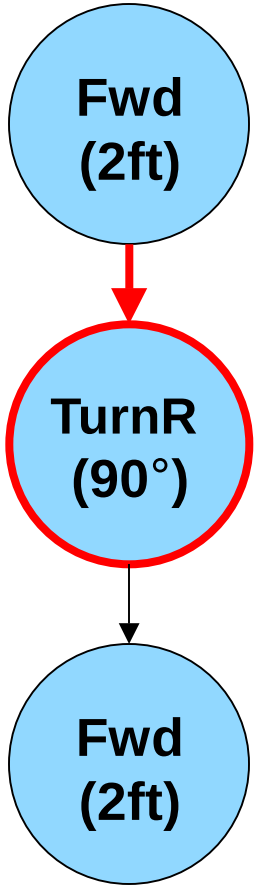
Finite State Machines offer another alternative for combining behaviors

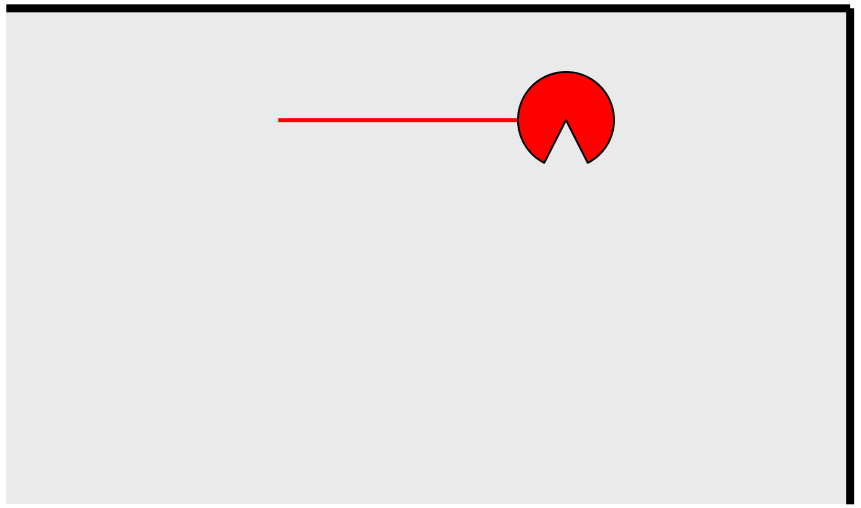




Each state is just a behavior and we can easily link them together to create an open loop control system

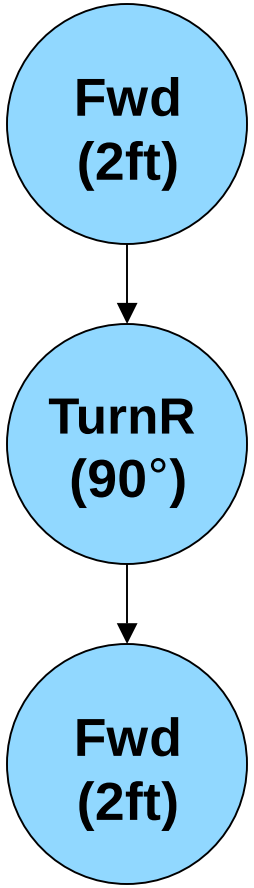
Finite State Machines offer another alternative for combining behaviors

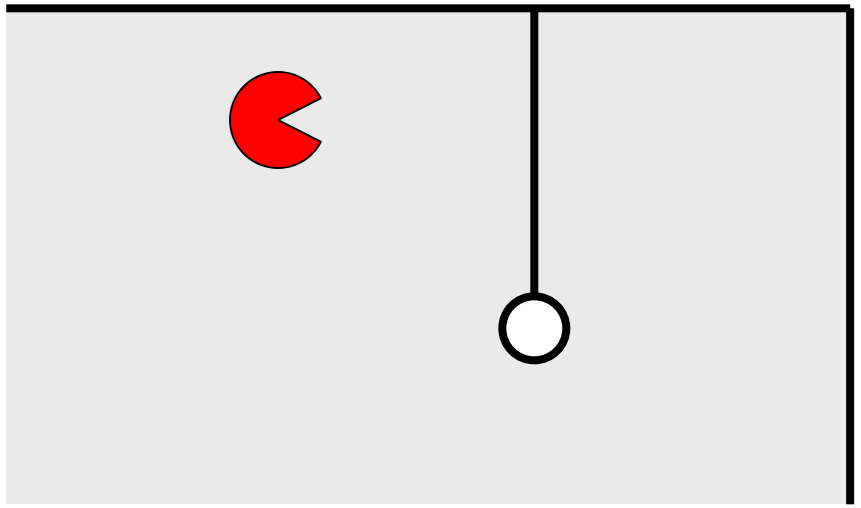




Each state is just a behavior and we can easily link them together to create an open loop control system

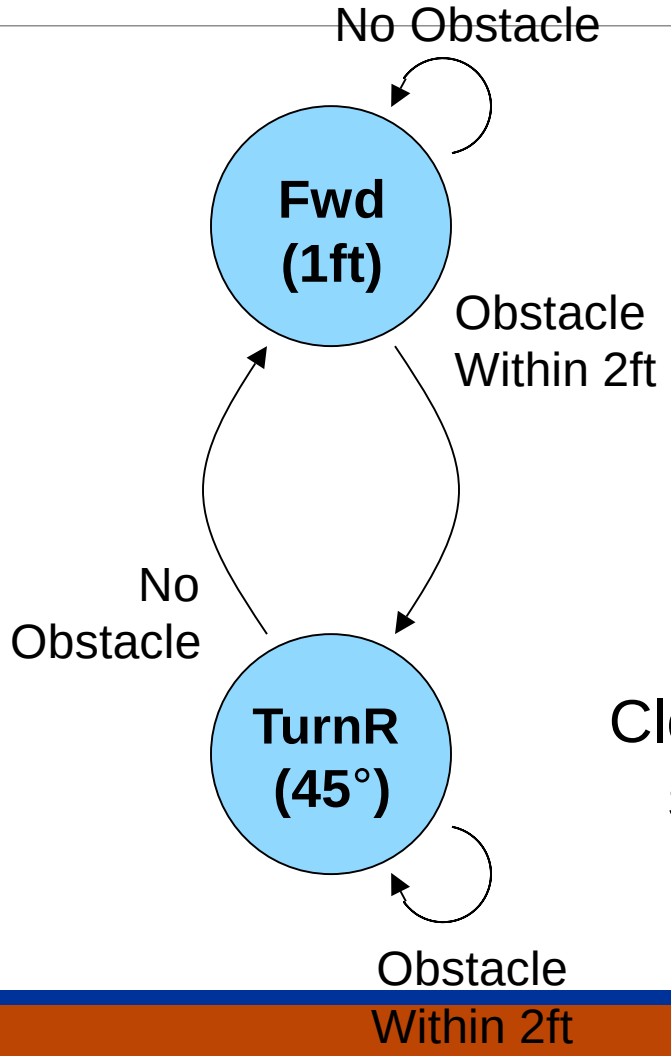
Finite State Machines offer another alternative for combining behaviors

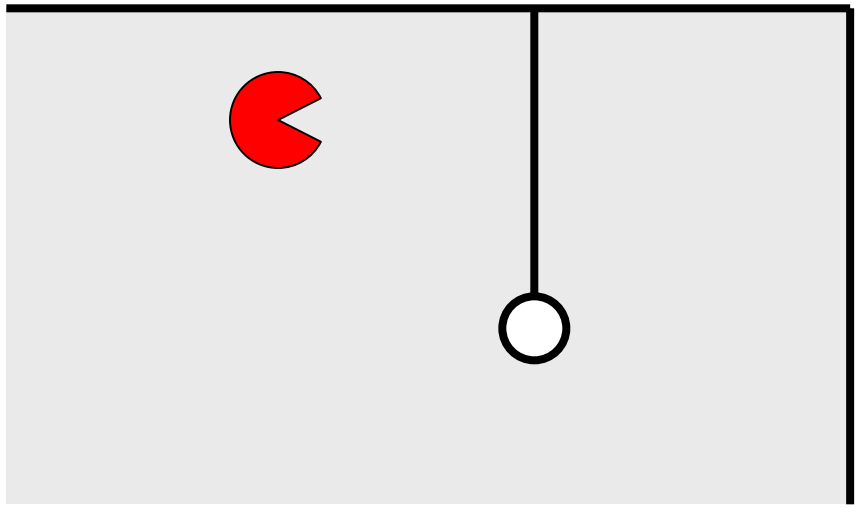




Since the Maslab playing field is unknown, open loop control systems have no hope of success!

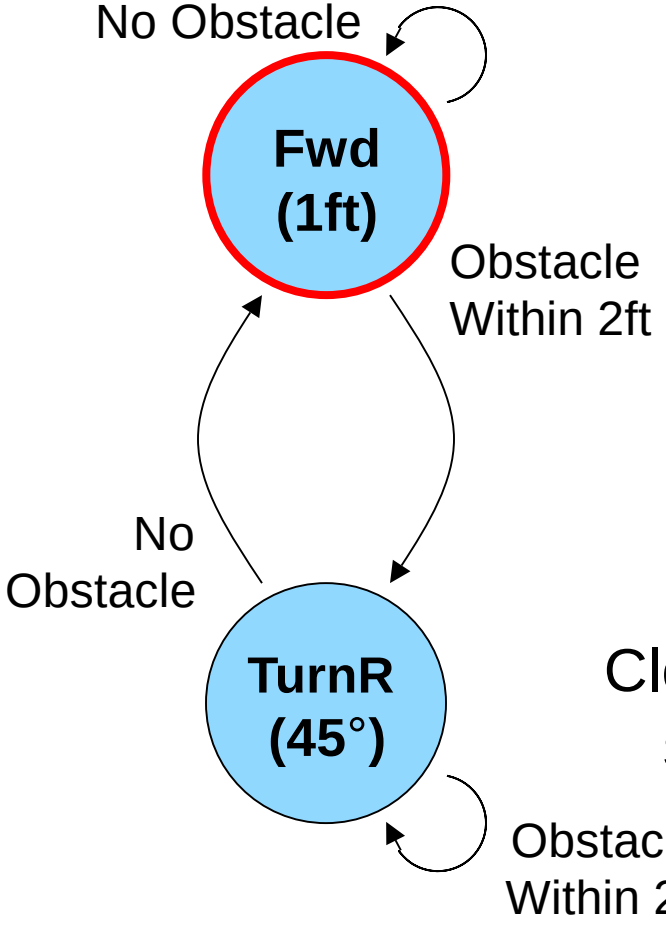
Finite State Machines offer another alternative for combining behaviors

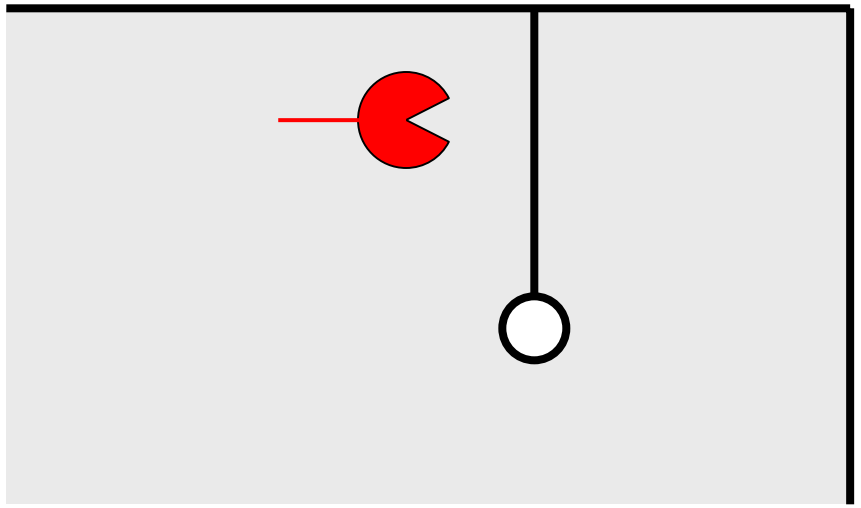




Closed loop finite state machines use sensor data as feedback to make state transitions

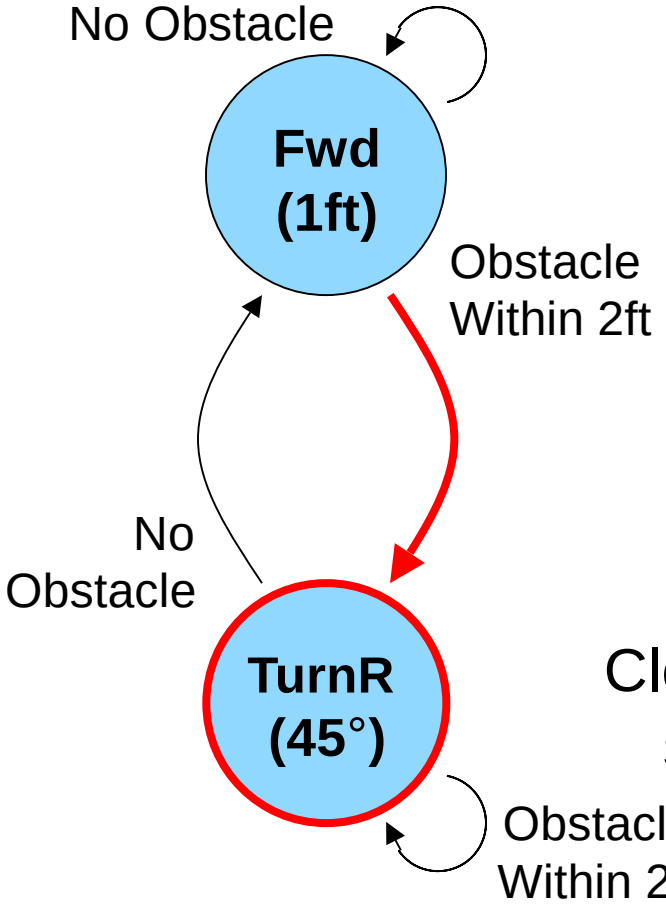
Finite State Machines offer another alternative for combining behaviors

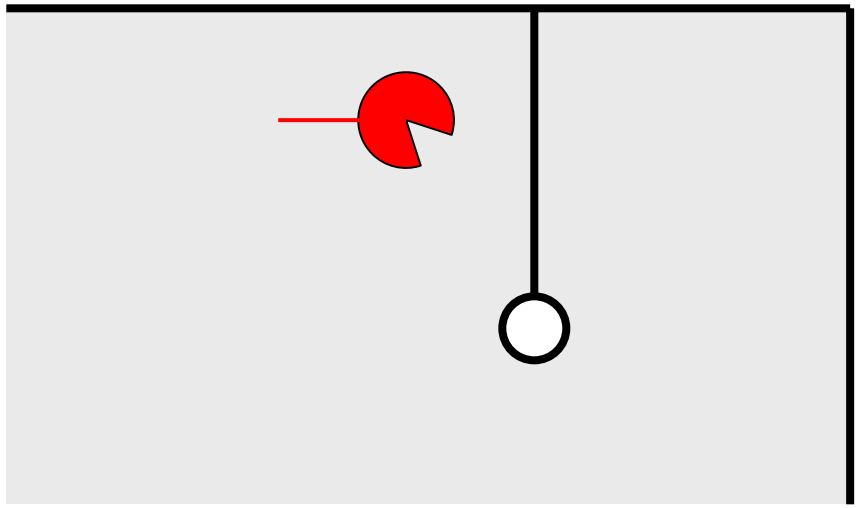




Closed loop finite state machines use sensor data as feedback to make state transitions

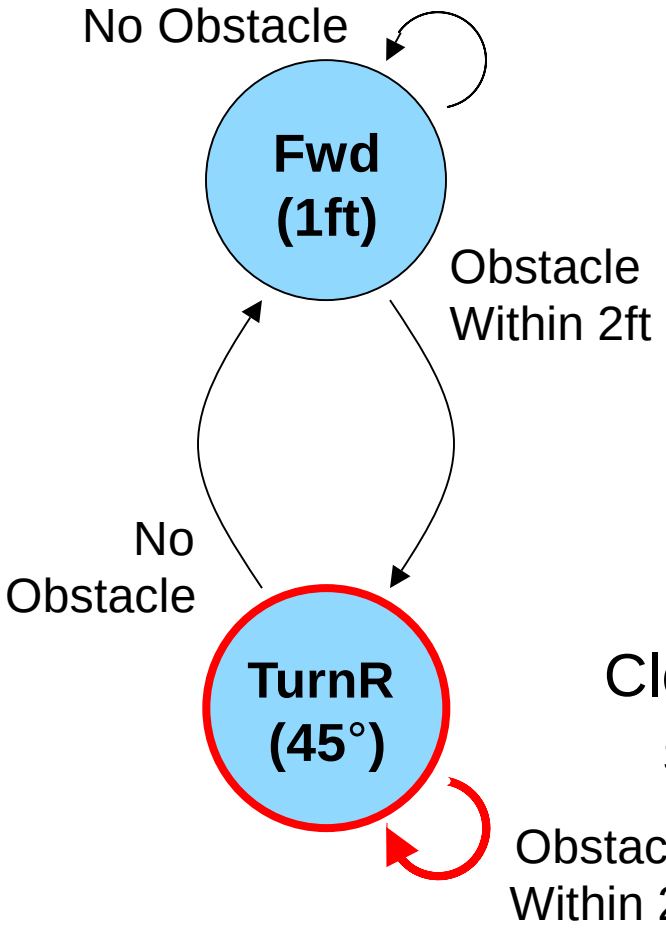
Finite State Machines offer another alternative for combining behaviors

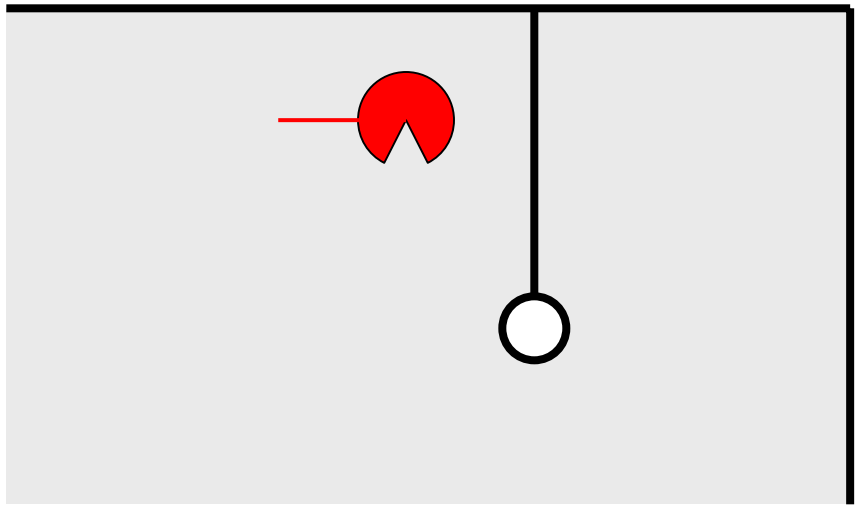




Closed loop finite state machines use sensor data as feedback to make state transitions

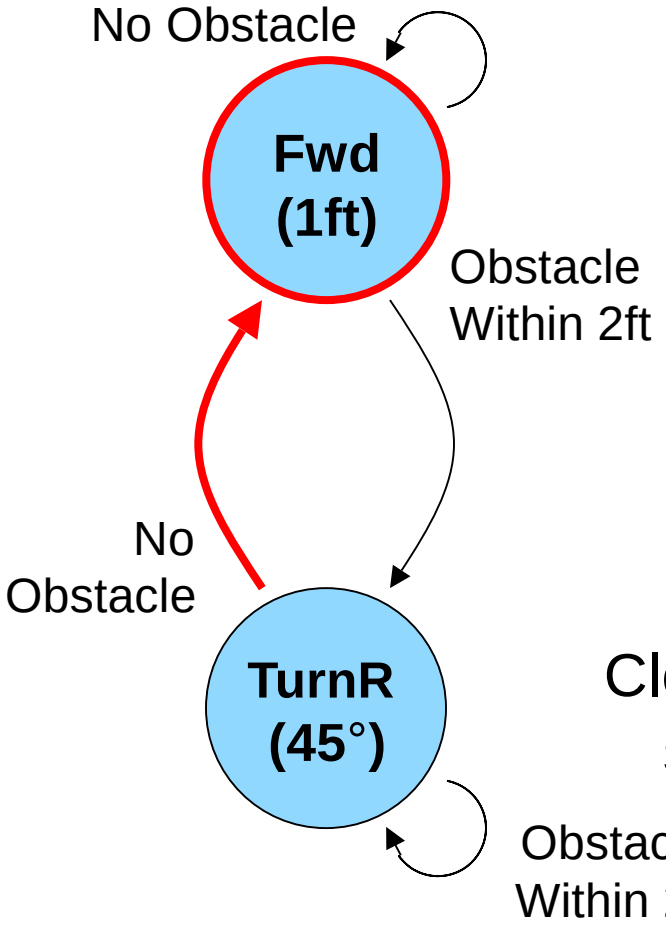
Finite State Machines offer another alternative for combining behaviors

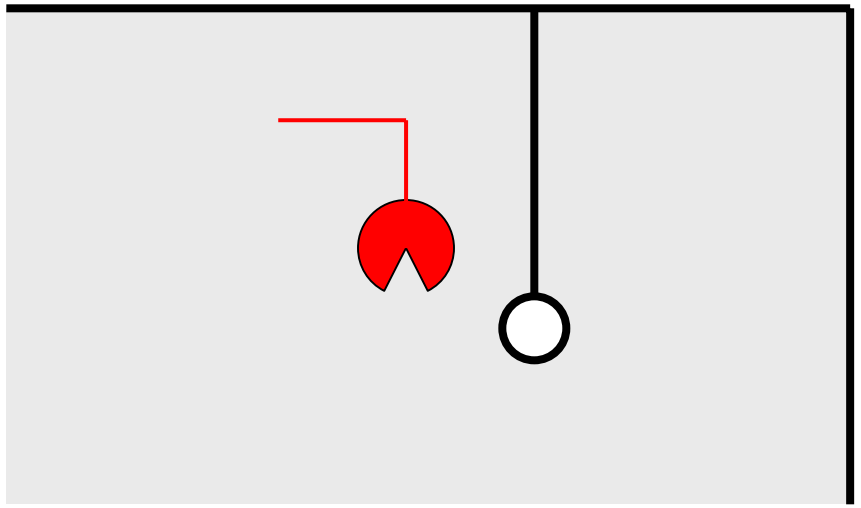




Closed loop finite state machines use sensor data as feedback to make state transitions

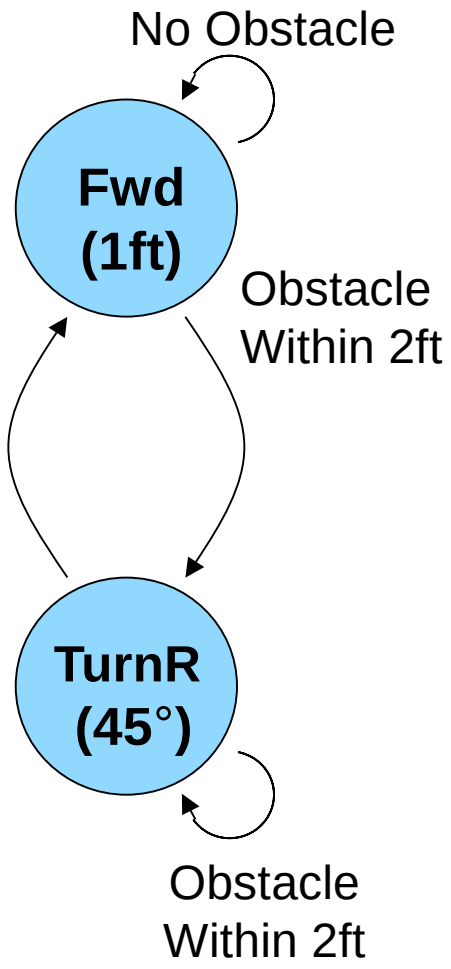
Finite State Machines offer another alternative for combining behaviors





Closed loop finite state machines use sensor data as feedback to make state transitions

Implementing a FSM in Java



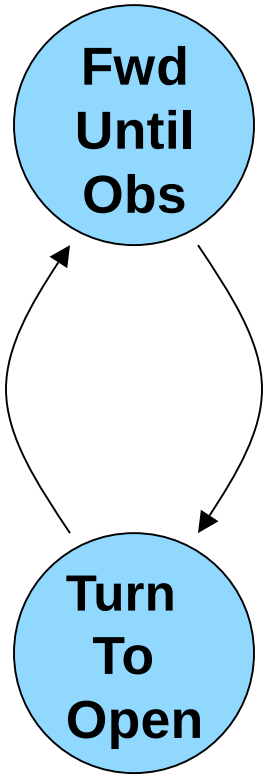

```
// State transitions
switch ( state ) {

  case States.Fwd_1 :
    if ( distanceToObstacle() < 2 )
      state = TurnR_45;
    break;

  case States.TurnR_45 :
    if ( distanceToObstacle() >= 2 )
      state = Fwd_1;
    break;

}

// State outputs
switch ( state ) {

  case States.Fwd_1 :
    moveFoward(1); break;

  case States.TurnR_45 :
    turnRight(45); break;

}
```

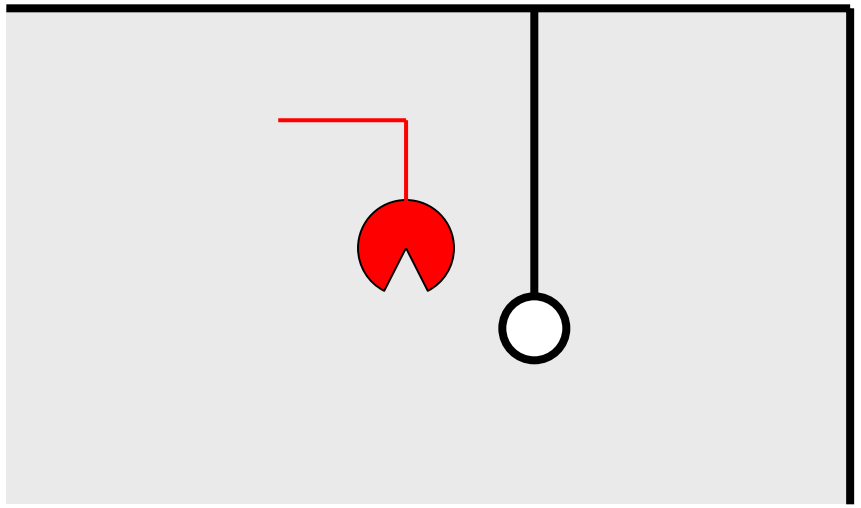
Implementing a FSM in Java

- Implement behaviors as parameterized functions
- First switch statement handles state transitions
- Second switch statement executes behaviors associated with each state
- Use enums for state variables

```
// State transitions
switch ( state ) {

  case States.Fwd_1 :
    if ( distanceToObstacle() < 2 )
      state = TurnR_45;
    break;

  case States.TurnR_45 :
    if ( distanceToObstacle() >= 2 )
      state = Fwd_1;
    break;

}

// State outputs
switch ( state ) {

  case States.Fwd_1 :
    moveFoward(1); break;

  case States.TurnR_45 :
    turnRight(45); break;

}
```

Finite State Machines offer another alternative for combining behaviors



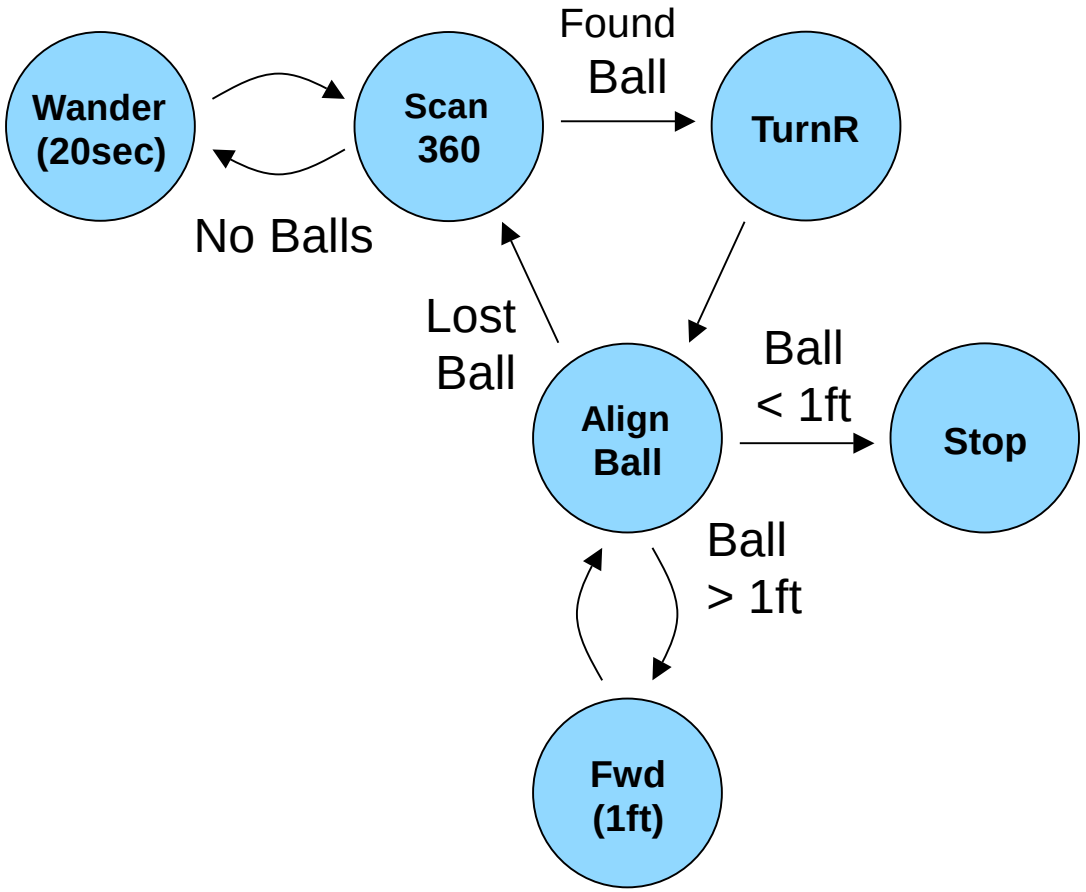


Can also fold closed loop feedback into the behaviors themselves

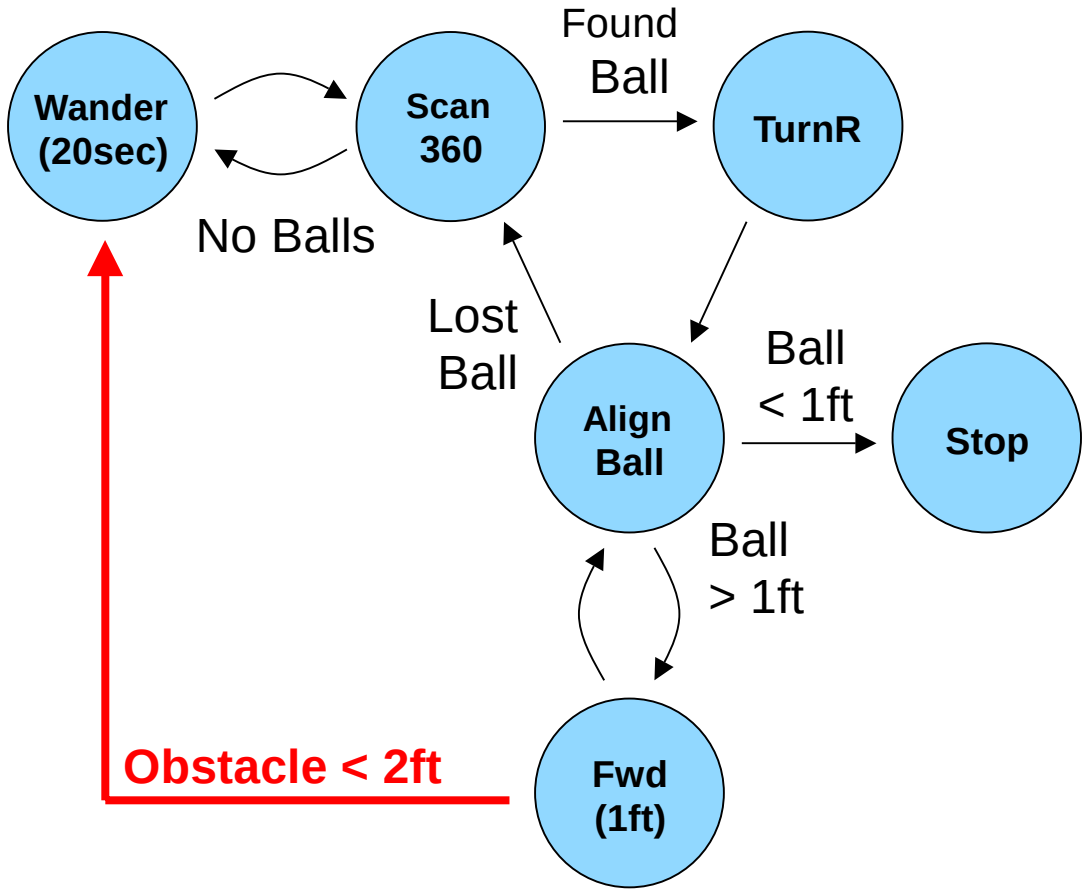
Case Study

SECTION 1

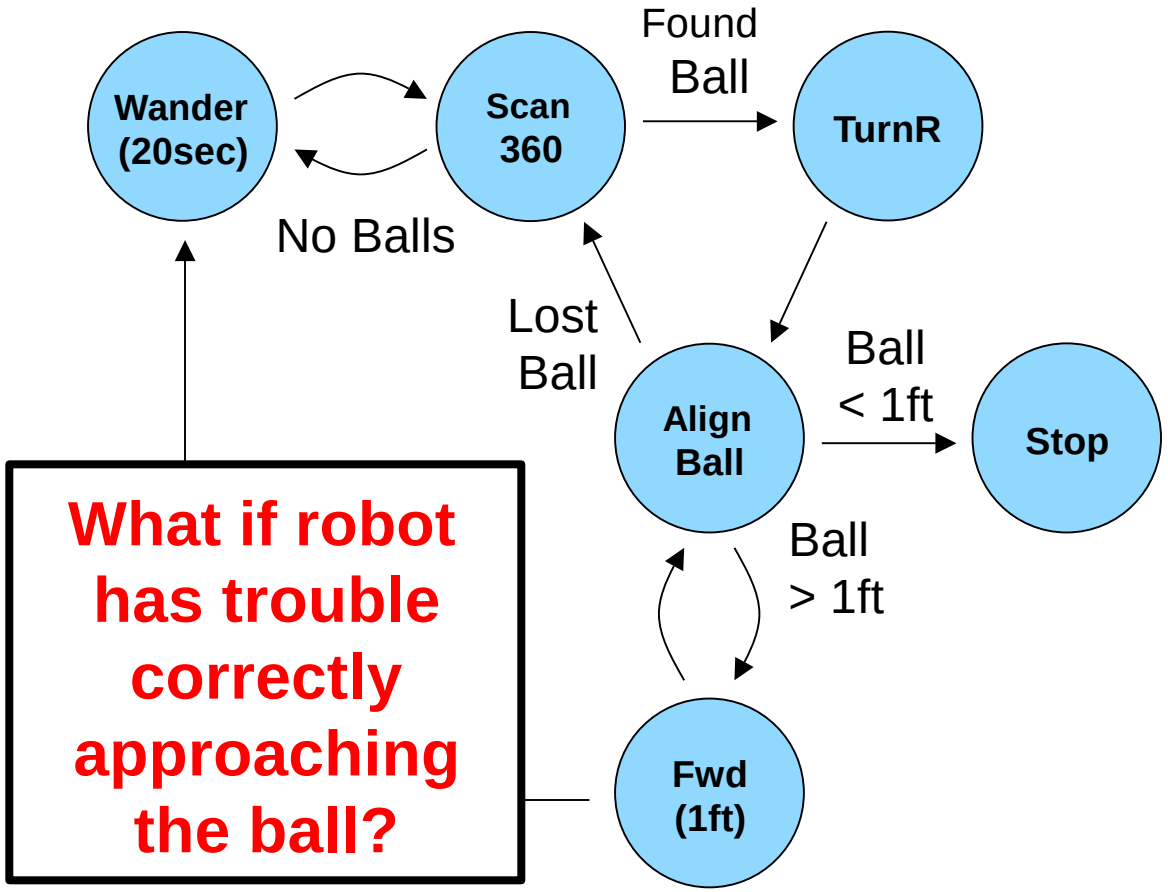
Simple finite state machine to locate red balls



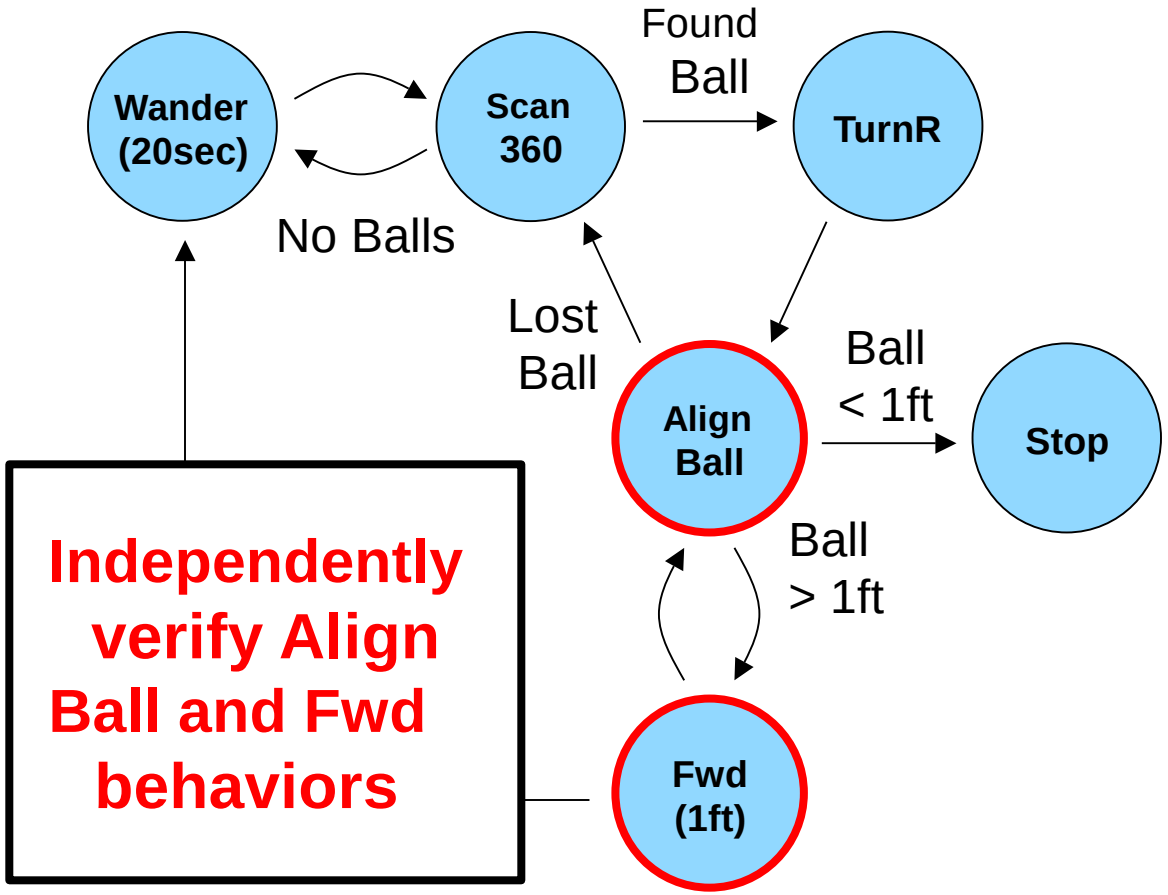
Simple finite state machine to locate red balls



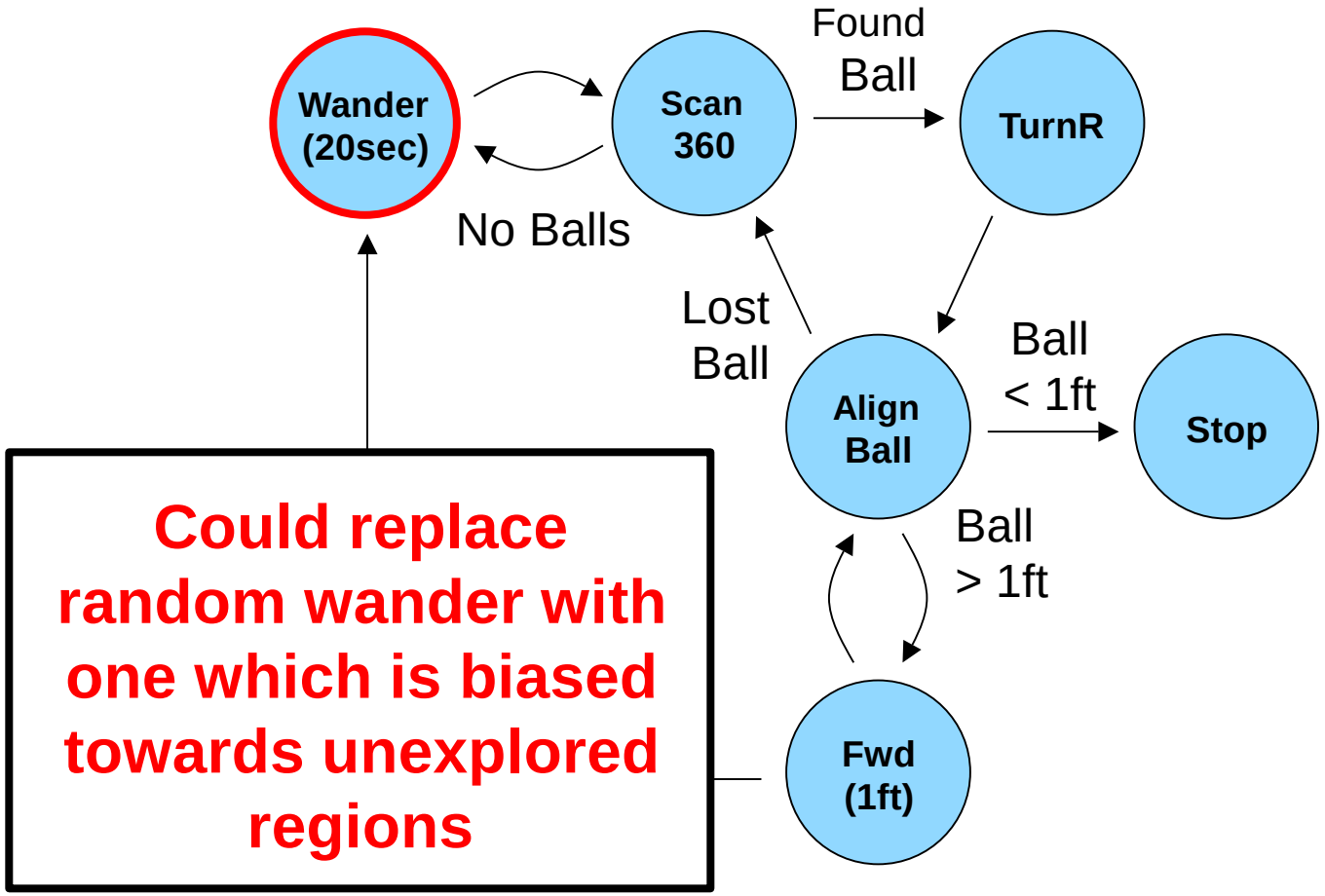
To debug a FSM control system verify behaviors and state transitions



To debug a FSM control system verify behaviors and state transitions



Improve FSM control system by replacing a state with a better implementation

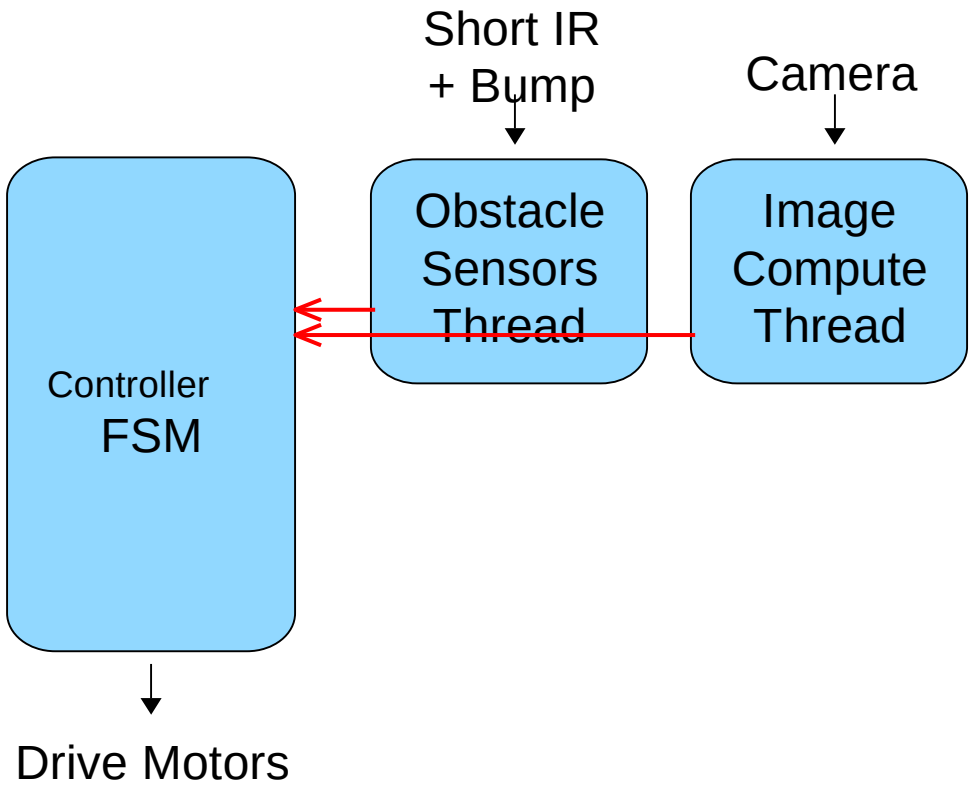


Improve FSM control system by replacing a state with a better implementation

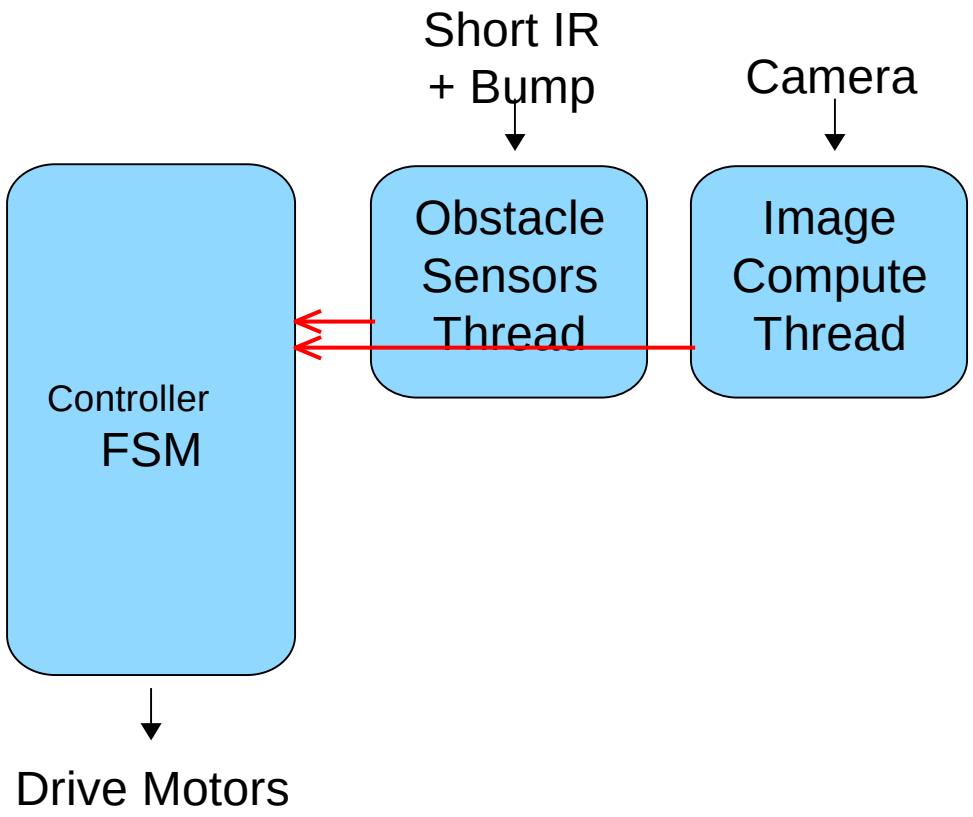
- What about integrating camera code into wander behavior so robot is always looking for red balls?
  - Image processing is time consuming so might not check for obstacles until too late
  - Not checking camera when rotating
  - Wander behavior begins to become monolithic

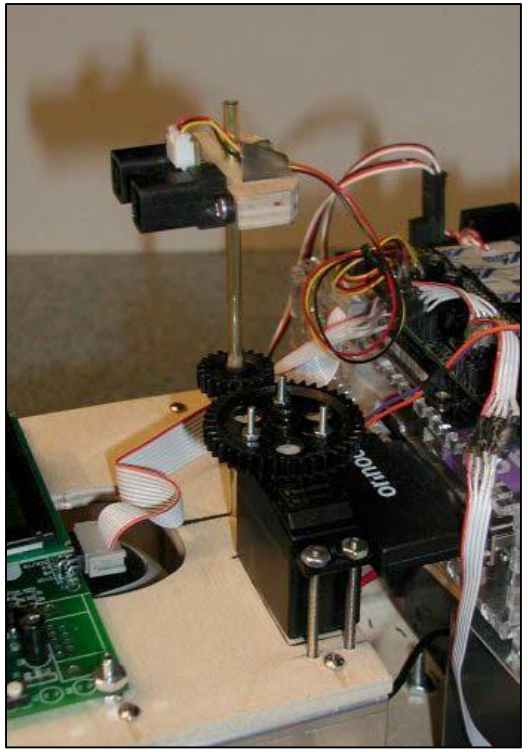
```
ball = false
turn both motors on
while ( !timeout and !ball )
  capture and process image
  if ( red ball ) ball = true

  read IR sensor
  if ( IR < thresh )
    stop motors
    rotate 90 degrees
    turn both motors on
  endif

endwhile
```

Multi-threaded finite state machine control systems

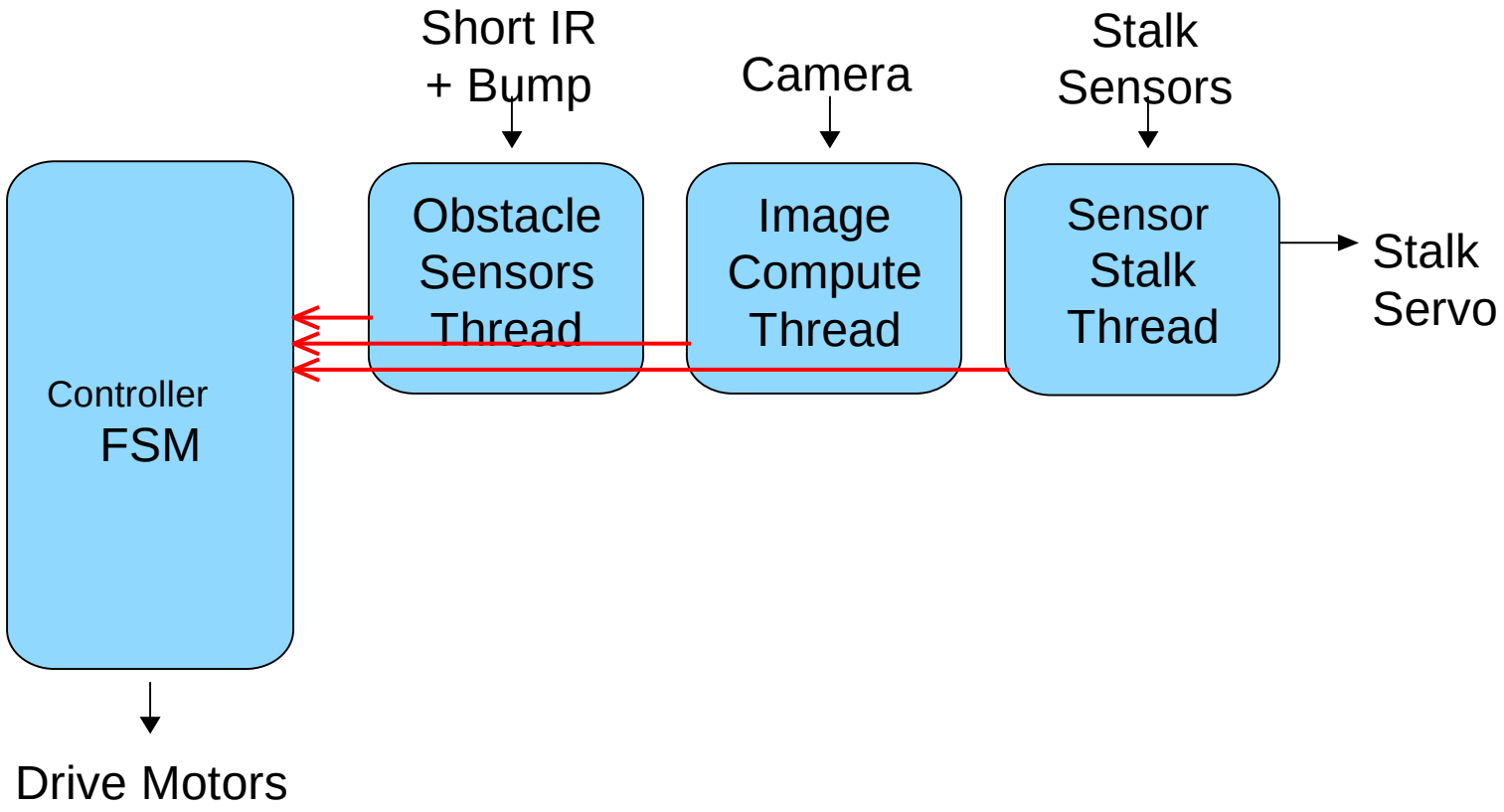


Multi-threaded finite state machine control systems

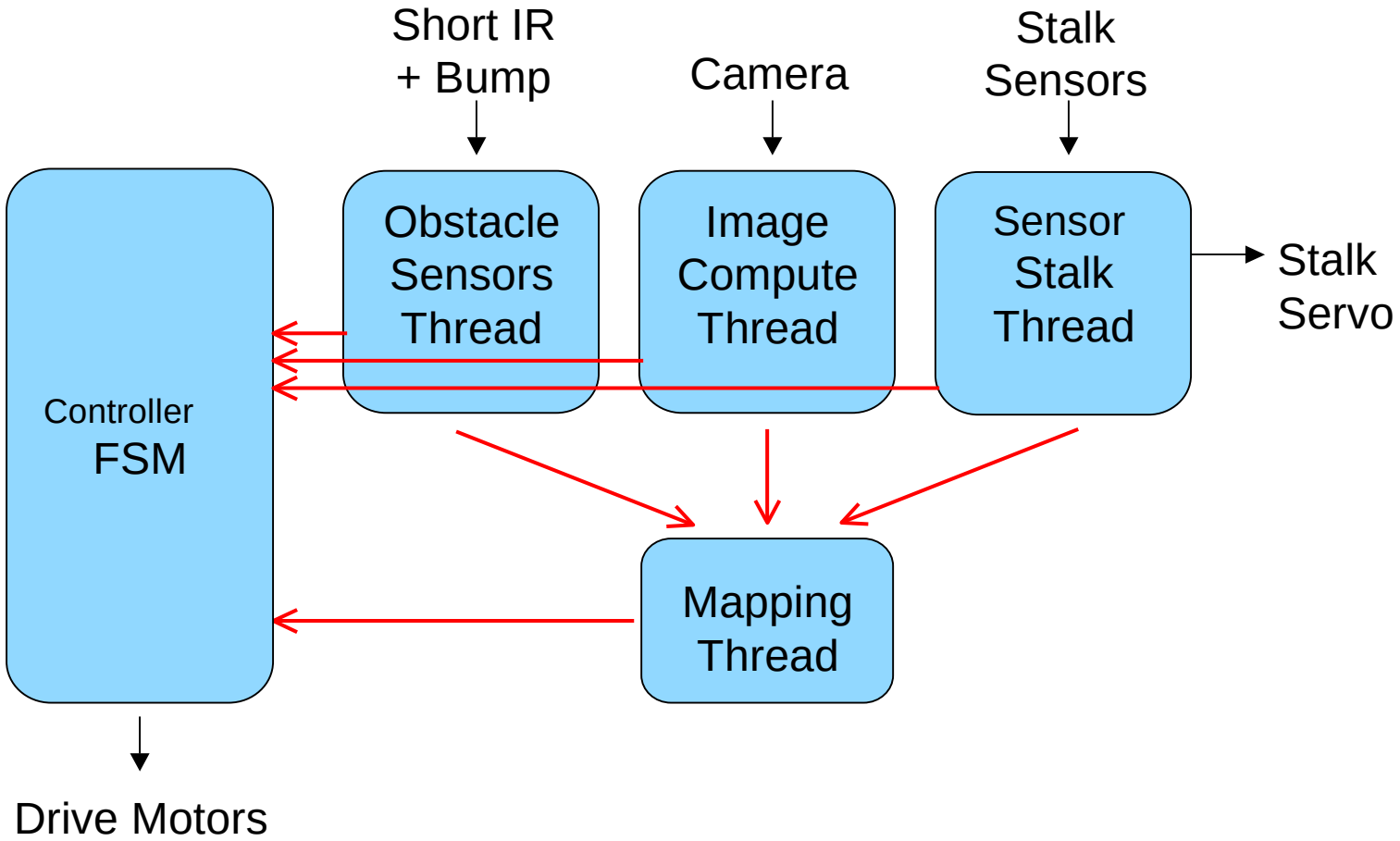




Multi-threaded finite state machine control systems



Multi-threaded finite state machine control systems



Python Coding

SECTION 1

FSM in Python

- Design of a Finite State Machine model in Python Language.
- Finite state machines can combine the model-plan-act and behavioral approaches and are a good starting point for your robotic control system modeling.