



CSCI 1106 Lecture 12

Introduction to Robotics



Announcements

- Today's Topics
 - Overview of Robotics Module
 - What is Robotics?
 - Anatomy of a Robot
 - The Sense-Decide-Act Cycle
 - Introduction to the Aseba Studio



The Robotics Module

Topics

- Overview of Robotics
- Hardware
 - Sensors
 - Actuators
- Software
 - State Transition Diagrams
 - Event Based Architecture
 - Dealing with Failure
 - Planning
 - Debugging
 - Programming Techniques

To Do List

- Six tutorials:
 - Introducing the Thymio II
 - Modeling sensors
 - Modeling actuators
 - Modeling the real world
 - Recovering from faults
 - Programming Techniques
- Robot Olympics Project
 - Design three programs to compete in the Robot Olympics
 - Implement the programs
 - Compete in the Robot Olympics
 - Write a technical report



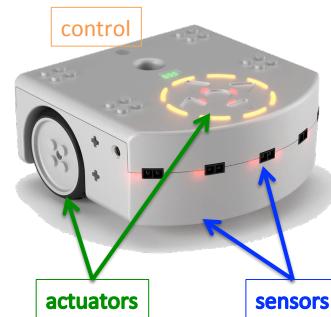
What is Robotics?

- From the Thrun, Burgard, and Fox
“Robotics is the science of perceiving and manipulating the physical world through computer-controlled devices.”

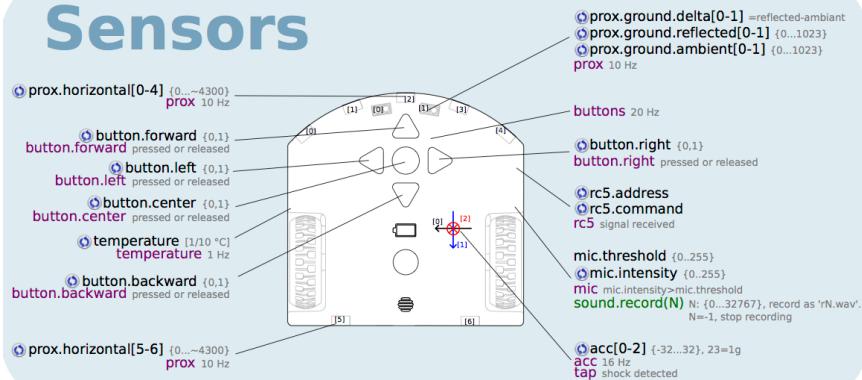
Anatomy of a Robot

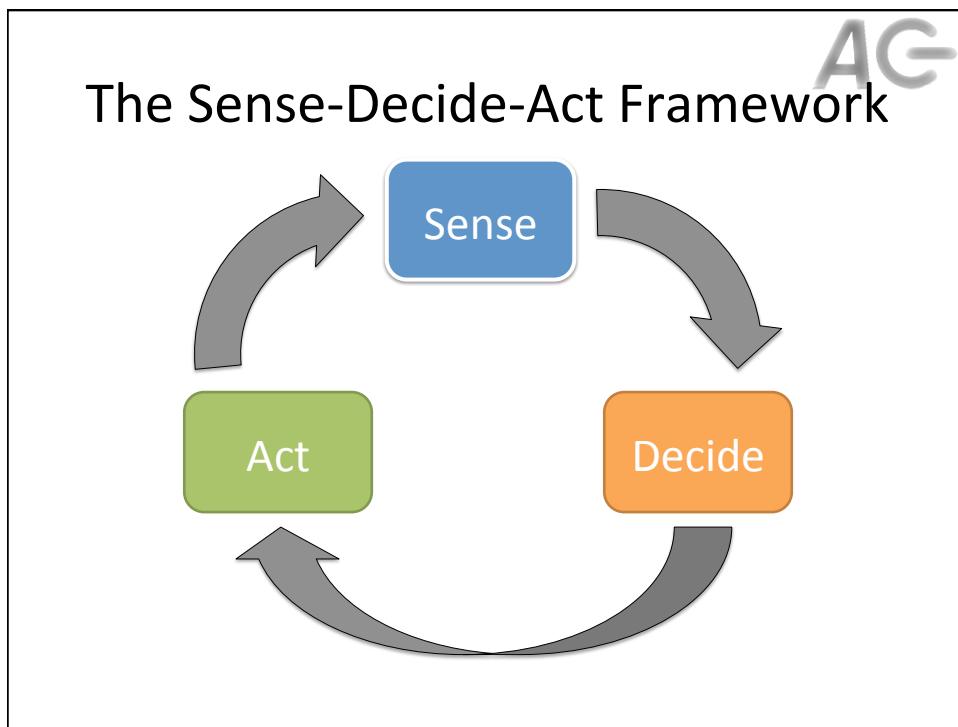
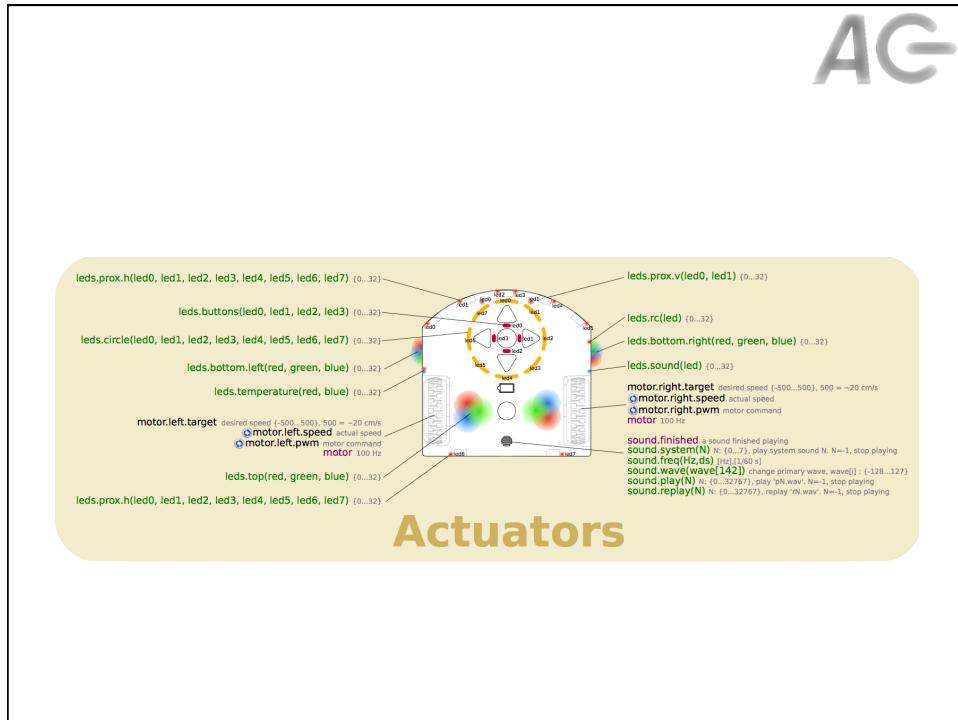
- Components:

- Sensors
- Controller
- Actuators



Sensors







Programming in Aseba

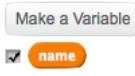
- Programs are text-based
- Key Ideas:
 - Everything is done by event handlers
 - A robot is a sprite
 - The world is the stage
- Observation this is similar to game design!



Scratch vs Aseba

Scratch

- Variables



- Event Handler



- Conditional



Aseba

- Variables

```
var name
var list[]
```

- Event Handler

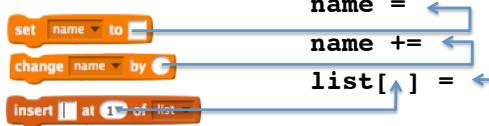
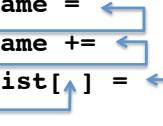
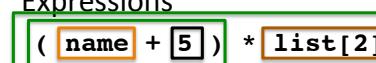
```
onevent prox
```

- Conditional

```
if then
end
```

AG

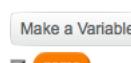
Scratch vs Aseba

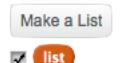
<p>Scratch</p> <ul style="list-style-type: none"> • Variable/List Assignment  <ul style="list-style-type: none"> • Expressions  <ul style="list-style-type: none"> • Motion 	<p>Aseba</p> <ul style="list-style-type: none"> • Variable/List Assignment  <ul style="list-style-type: none"> • Expressions  <ul style="list-style-type: none"> • Motion <pre>motor.left.target = motor.right.target =</pre>
--	---

AG

The Four Parts of an Aseba Program

- Variable declarations
 - Begin with the **var** keyword
- Initialization code
 - Anything except declarations
- Subroutines
 - Begin with the **sub** keyword
- Event handlers
 - Begin with the **onevent** keyword









A Sample Program

Make a Variable

```

var speed = 100
 speed

motor.left.target = 0
motor.right.target = 0

onevent button.forward
  motor.left.target = speed
  motor.right.target = speed
when up arrow key pressed
  move speed

onevent button.backward
  motor.left.target = 0
  motor.right.target = 0
when down arrow key pressed
  move 0

onevent button.left
  motor.left.target = -speed
  motor.right.target = speed
when left arrow key pressed
  turn ↘

onevent button.right
  motor.left.target = speed
  motor.right.target = -speed
when right arrow key pressed
  turn ↙

```

Key Idea: Actuators are controlled by setting variables that represent them

Sensors and Actuators in Aseba

- Key Idea: All sensors and actuators are accessed via predefined variables, e.g.,
 - to control motors, assign values to motor variables


```
motor.left.target = 100
motor.right.target = 100
```
 - to check if an object is close, read proximity variable


```
if prox.horizontal[2] > 1000 then
  ...
end
```


- Where are all the predefined variables listed?
- When do we check variables?

A screenshot of the Aseba Studio software interface. The main window title is "Untitled - Aseba Studio". The "Code Area" contains the number "1" and the text "Compilation success. ✓". The "Variables" panel shows a list of variables with their names and values, such as "id" (1), "event.source" (1), "event.args" (32), etc. The "Constants" panel is empty. The "Global Events" panel lists several events with timestamps: 11:53:48.919, 11:53:48.931, 11:53:48.934, 11:53:48.934, 11:53:48.934, 11:53:48.956, 11:53:48.956, 11:53:48.959, 11:53:48.959, and 11:53:48.959. The "Native Functions", "Local Events", and "Local Tools" panels are also visible.

<https://aseba.wikidot.com/en:thymioapi>

When do We Check the Sensors?

- Key Idea: Sensors generate events. Event handlers check sensors. E.g.,
 - Proximity (prox) sensors generate 10 events per second

```
onevent prox<
    if prox.horizontal[2] > 1000 then<
        motor.left.target = 0
        motor.right.target = 0
    else
        motor.left.target = 100
        motor.right.target = 100
    end
```

• Scratch and Aseba are very similar!

A Scratch script showing a "when I receive prox" hat block. It has an "if distance to [] < [] then" control block. Inside the control block, there are two "move [] steps" blocks: one set to 0 and another set to 100. Arrows from the Aseba code above point to the "prox" sensor in the Scratch script and the "if" condition in the Scratch script.

