6.270 Lecture 3

Board programming & Control Theory

Edwin Foo '98 <efoo@mit.edu>
Compaq Cambridge Research Laboratory

Overview

- Practical Stuff
  - When you'll be getting boards
  - Board details, care/maintenance
  - Workstations/serial connections
  - Software overview
- Boring but important stuff
  - open/closed loop systems
  - feedback
  - multitasking
  - programming style

When You'll be Getting Boards

- Hopefully Wednesday, 1/12/99
- What you'll be getting:
  - controller board/daughtercard assembly
  - serial cable
  - nice mylar plastic baggie
- What else you'll need (getting next Monday/Tues):
  - parts for battery packs
  - LCD panels

Board Care

- Obvious Stuff
  - um, don't drop it
  - avoid static - shoes, clothes, other equipment, etc.
- Not so obvious stuff
  - make sure to wire up batteries correctly! (diagram will be posted in lab)
  - wire sensors/motors properly
    - check for shorts, not just internal to sensor, but between sensors
  - don't run stalled motors for too long (poof!)

Personal Server MLB specs

- 200 MHz StrongARM CPU
- 16 megabytes SDRAM
- 4 megabytes Flash
- USB controller
- runs NetBSD, Linux
- We're using NetBSD for 6.270
- Kaffe Java virtual machine

RoboDC Specifications

- USB peripheral + 8051 microcontroller (24 MHz, 8KB RAM)
- 32 analog inputs
- 18 Half H-Bridge motor outputs
- 4 servo ports
- LCD connector
- 8 shaft encoder/counter ports

Workstations/Serial Connections

- Lab workstations
  - HP-UX workstations in 6th floor lab (roughly 18)
  - PCs in back of 6th floor lab (18)
  - 6.001 lab PCs
  - don't steal the serial cables! Grr!
- Personal computers
  - Windows 95/98/NT supported
  - x86 Linux/NetBSD supported
  - limited Macintosh support
  - other stuff - come see me

Software

- Much more detail TBA @ http://web.mit.edu/6.270/tech (not up yet)
- Mailing list - 6.270-tech@mit.edu, discuss archives
- Serial terminal programs
  - 38400 baud, no parity, 8 data bits,1 stop bit (38400 N81)
- Java JDK version 1.1.7
- NetBSD kernel/ramdisk versions available via web
- demonstrations on how to compile/load/download code available on Wednesday

Quick 10-minute Break

Q&A?

Intro to Control Theory

- Open loop control
  - no feedback
  - assumes successful completion of each step
  - It's bad
- Closed loop control
  - has feedback
  - It's good
  - but time-consuming to implement

Feedback and Branching

- How to get feedback
  - sensor thresholds
  - timeouts
  - heuristics
  - preserve state between contest runs
- How to handle feedback
  - nested ifs/switch statements
  - threads/multitasking
  - event-driven models
  - behaviors

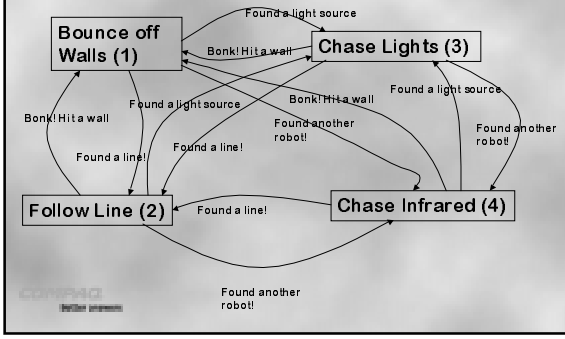
Potential Problems

- Debug
  - always reduce to simplest possible case
  - test components separately
  - don't just randomly change stuff until it works! =)
- Control
  - try to anticipate all possible actions
  - flowchart your code flow
  - always have a way out!

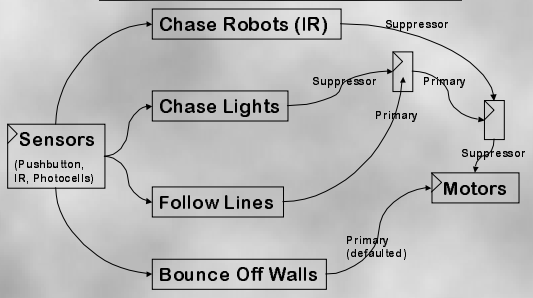
Examples

- Opening an arm
- Shaft encoding
- Line follower
- Making a turn
- timing out on a block capture

Embedded State Machine Example



AFSM Solution - Behavior Modeling*

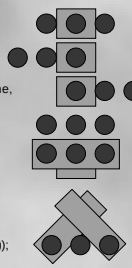


*based on work by Rodney Brooks (MIT AI Lab) - subsumptio

Robot->FSM->Java code example

Line Follower

```
class LineFollower extends AFSM {
  Register leftSensor, rightSensor
        middleSensor;
  Register steeringAngle, special,
        speed;
  State leftOfLine, onLine, rightOfLine,
      offOfLine, onJunction,
      weirdoLine;
  LineFollower(Register ls,
        Register rs,
        Register ms) {
    leftSensor = ls; ...
    leftOfLine = new
        State(enterState(),
         action(),leaveState());
    Etc...
};
```



States

OnLine: steeringAngle.set(90);
LeftOfLine: steeringAngle.set(?);
RightOfLine: steeringAngle.set(?);
OffOfLine: special.set(OFFLINE);
Junction: special.set(JUNCTION);
Weird: special.set(WIERDO);

And now..

The Matt and Kenneth show