CS4500 Sponsor Sales Pitch

SYSTRONIX
Embedded Java Spoken Here

Autonomous Robot Swarm with Legos, JCX, Sonar, Vision, and 802.15.4 Radios

Bruce Boyes

Technical Director, President & Janitor
Systronix, Inc
www.icx.systronix.com

Description of the problem:

2000 Robots?

"Right now we have two robots on Mars. What if we had 2,000? ...they would have to be autonomous... work on their own and cooperate and communicate. *But nobody has any idea how to program 2,000 robots right now*".

James McLurkin, 2003 Lemelson-MIT Student Prize Winner
http://www.pbs.org/wgbh/nova/sciencenow/3204/03-talk.html

Practical Robot Swarm Programming

THE GOAL:

Develop a working example of a simple robot swarm (n>8), building upon existing technologies, adding some new algorithms and hardware. Robots are heterogeneous yet share the same code.

Our Robot Swarm

Collaboration between many JCX robots

- "Robots" here = "autonomous"
  - Common CPU – native execution Java ($3x10^6$ bcps)
  - Common pool of sensors and effectors; Each robot entity includes [0..n] of each
- Sensors
  - Light, touch, rotation, vision, compass
  - Generic analog input (CdS cell, etc)
- Effectors & mechanical construction
  - Wheels, tracks, skid steer or other
- Legos Mindstorms for quick & easy chassis

Other Available Technologies

Almost baked – but not 100% complete

- 802.15.4 2.4 GHz wireless
- Octal Sonar, IR Rangefinder, etc
- Digital compass
- R/C Servos (use Yost prototype?)
- Integration with wireless sensors
  - Sunspots prototypes
  - Crossbow
- Audio output
- Robot face

Other Options and Possibilities

How to apply any extra resources

- Video analysis of behaviors
  - Does it match the intention? Can it be interpreted?
  - Interface with existing grad students
- Jay's lab – wireless routing
- Emergent behavior
  - i.e. That which was not specifically programmed
- Vex robotics chassis – JavaOne BOF
- Other sensors such as human body
- Configuration information in the sensor/effector
  - IEEE1451 wrappers

Practical problems, in general

Few swarms are implemented in hardware

- Affordable hardware is not capable of interesting behaviour
  - “insects” bumping into each other...
- Capable hardware is too expensive
- So, simulation is usually employed instead
- But now hardware is both capable, affordable, and (we hope) programmable
  - Leverage the benefits of Java
  - Build on existing libraries
  - Share code with the world

Particular issues to tackle here

If it were easy, it would already be done, right?
(Simulations seem to always include a Supreme Being...)

- Make better use of robot sensors to deal with:
  - Autonomy of each robot
  - Position relative to other swarm members
  - Stationary and mobile obstacles
- How: algorithms to interpret proximity sensor data as dynamic, multi-dimensional data
  - Use knowledge of your own motion, calibration
  - Start with two sonar modules per robot
  - Move up to ring of 4 to 8 sonar modules
- RF with other bots and stationary sensors

What's Expected of the Team(s)

public static void final SELF_DOCUMENTING_CODE=false;

- Well-documented code is usable code
  - I'd rather have well-commented non-working code than code which works and has no comments. Really.
  - Others must be able to use and understand your work
  - We're collaborating to create a common code base
  - There's no such thing as “self-documenting” code
  - Java has good doc help tools so there's no excuse
  - “Documentation” includes supporting files and notes
- Poorly documented code is practically worthless
  - It's unmaintainable, even by the author(s)
  - “Quick and dirty is forever”

Anticipated unknowns

Known unknowns?

- Can we make swarm behavior “interesting”?
  - Seeking and following a human
  - Self-organizing robot leadership
- Can we expect emergent behaviors?
- Interference of sonar between robots
- Some believe sonar just can't be used this way

What Systronix Will Provide

- Weekly meetings
- Loan of JCX hardware, including radios
- LCAD models of suggested chassis
- Prompt answers to questions
- Hardware prototype expertise if needed
- Possible co-authorship of JavaOne presentation

What we have to attack the problem

Self-configuring hardware vs hard-coding per robot

- Self describing I/O:
  - Each board stores basic information
  - Manufacturer, board type, revision, etc
  - Runtime class(es) used to support this board
  - (Protect this data from accidental erasure)
- Each I/O point has specific information
  - Type of sensor (light, touch) or effector (wheel, tread)
  - Performance and calibration (thermocouple, etc) data
  - Space for a useful name such as “steering servo”
- We use XML to store this data
  - Single code base self-configures on startup

Build code on existing work

A lot of the basic layers are done already

- Motor and sensor libraries available today:
  - DC motors (Lego & other similar 9 VDC 250 mA)
  - Light, touch, rotation sensors (Lego)
  - Sonar (detect broomstick @ 3 meters), 5 cm min
  - CMUcam color vision
- Prototypes available for CS4500
  - 802.15.4 Maxstream radio modules
  - Sunspots (Sun labs)
  - Wireless stationary sensors w/802.15.4
  - others

Why you should choose this project

(besides it being fun)

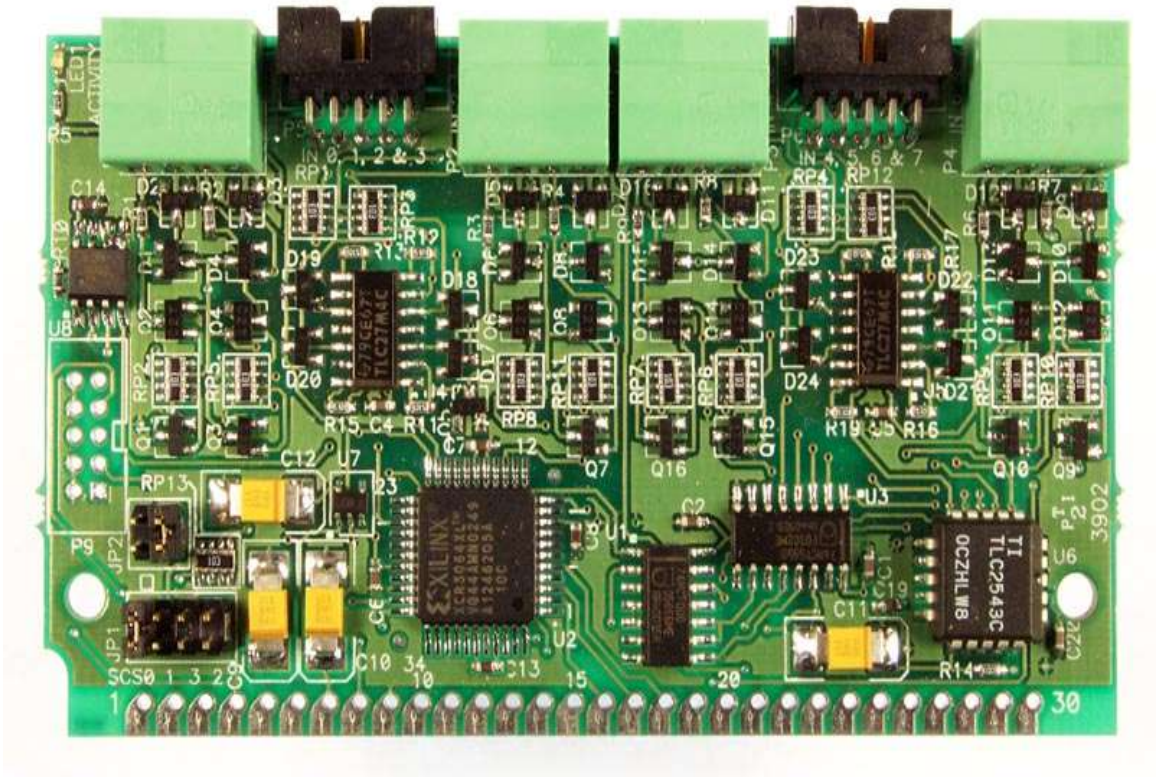
- Learn/Use useful technologies
  - Java, Eclipse, Ant
  - Embedded systems
  - Realtime control
  - Robotics
  - Wireless 802.15.4, maybe Zigbee
  - Working with electro-mechanical hardware
- Industry exposure
  - Project and code posted at java.net (150,000 devs)
  - Demo at JavaOne2006 (SFO, May 15-18) (Keynote?)
  - Project description on systronix website

JCX hardware architecture & tagging

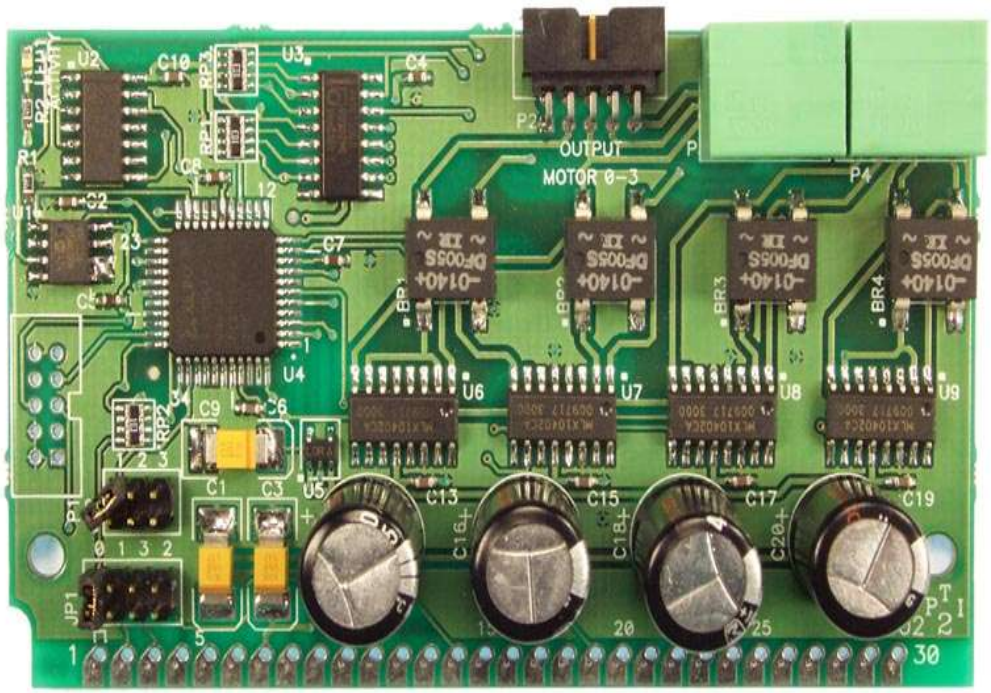
EEPROM tags are built into the I/O address space

- All I/O devices are SPI or I2C slaves
  - Each board or device uses 1 or more slave selects
  - 24-32 slave select addresses are available
  - Each slave select is typically 1-8 motors, sensors, etc
- Each SPI slave has “shadow” tagging memory
  - Eeprom exists in the lsb of each slave address
  - Enumeration thus must find tagging at each slave
  - Enumeration “tells all” about that device
  - CRC detects slave address conflict
- Same idea applies to I2C
- SPI & I2C references at jcx.systronix.com

JCX Sensor Board Photo

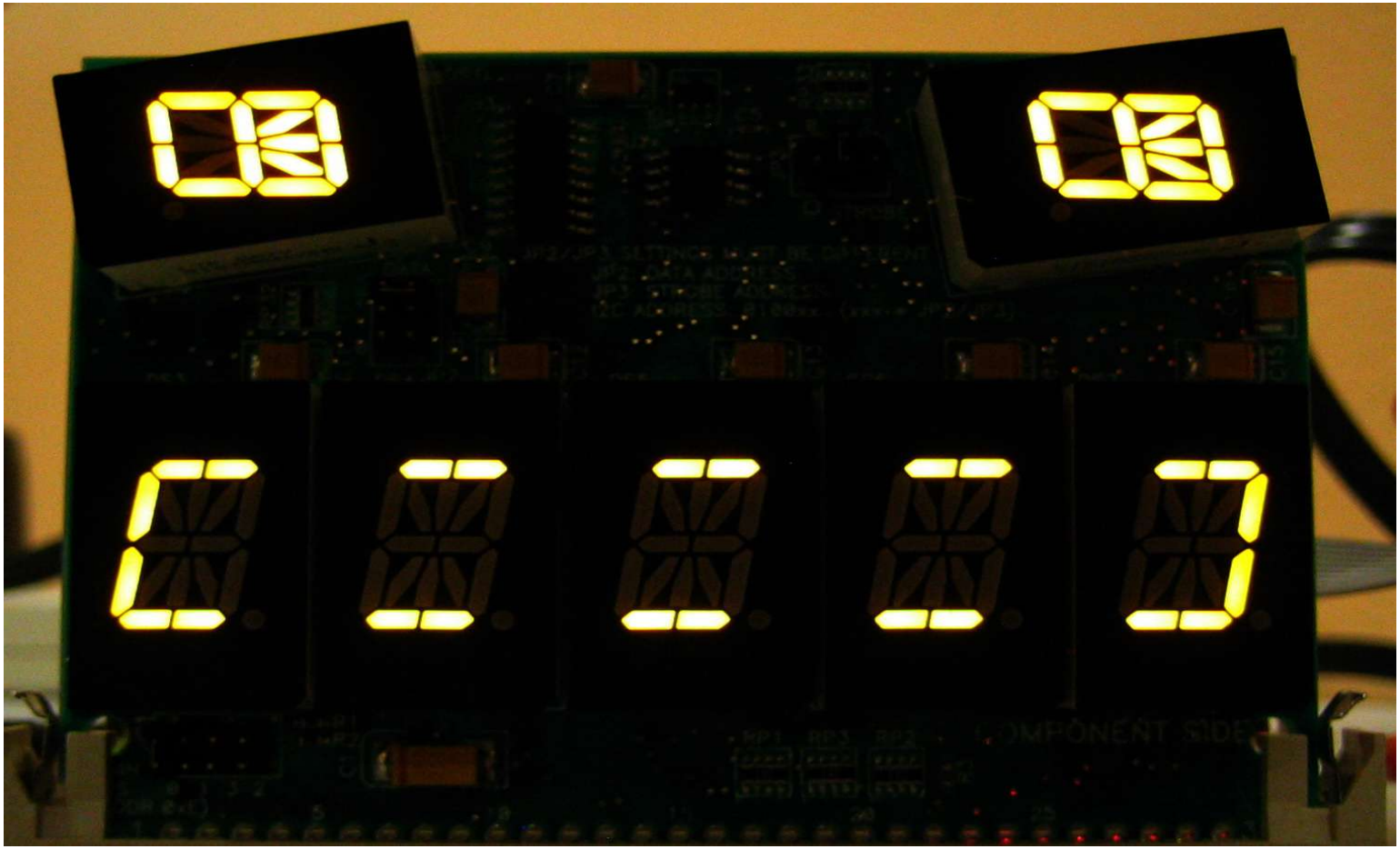


JCX Motor Board Photo



Robot Face

note 'eyes' looking left and 'mouth' which is open wide



More Information

Items referenced in this session:

- java.net robotics community (started this week)
- http://www.jcx.systronix.com
- http://www.ajile.com
- http://kxml.sourceforge.net/
- *Creating Socially Interactive Robots*, Cynthia Breazal

Standards and swarm research

- http://www.jdroid.com
- http://www.omg.org/robotics-corner/2.htm
- http://news.zdnet.com/2100-9584_22-993385.html
- http://www.inel.gov/adaptiverobotics/default.shtml

Q&A

Or email: bboyes@systronix.com
please include 'cs4500' in the subject