

Factory Floor Testbed

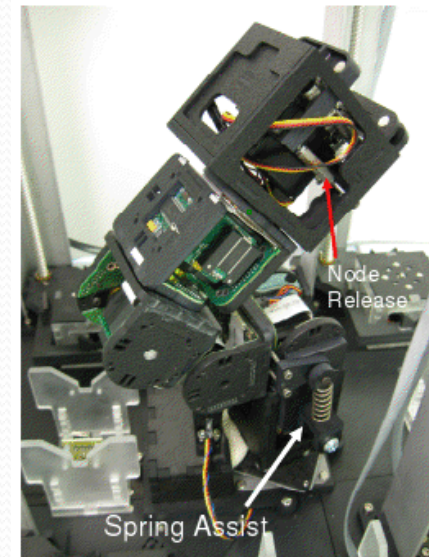
Stefan Kristjansson

Andrew Lawrence

Richard Wood

Table of Contents

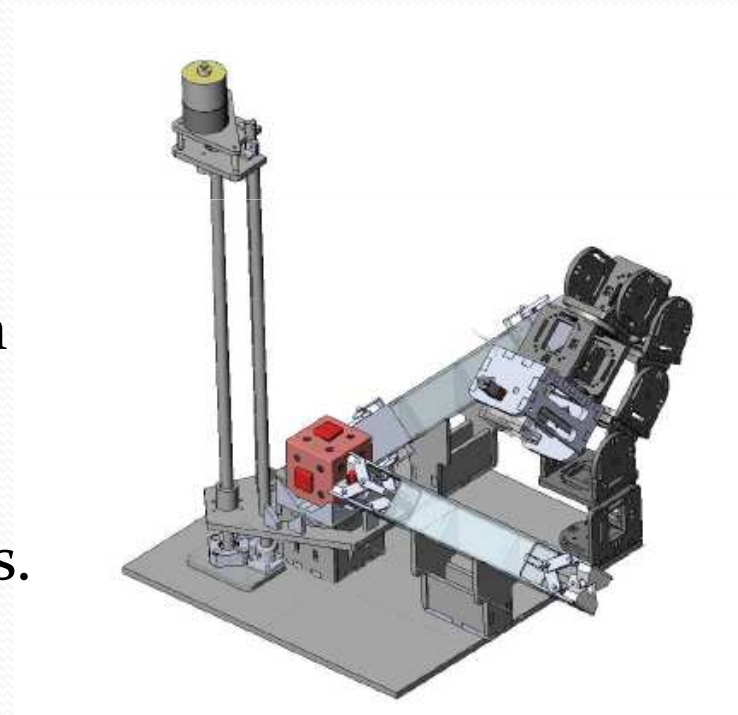
- Project Overview
- Who is the customer?
- Development Goals
- Description / identification of plant, actuators, sensors, control resources
- Performance Criteria
- System inputs and interfaces.
- Cost and Schedule constraints
- Project Plan and personnel use
- Related work and bibliography



Project Overview

The factory floor testbed is an experimental project for robotically constructing reconfigurable truss structures

Distributed assembly algorithm allows an array of construction robots to work in unison for higher functional developments.





Project Overview cont.

- The mechanical system was designed and built by the Modular Robotics Laboratory at the University of Pennsylvania.
- With a single experimental test unit, our goal is to develop the software and algorithms to control the robot in the construction of a structure.

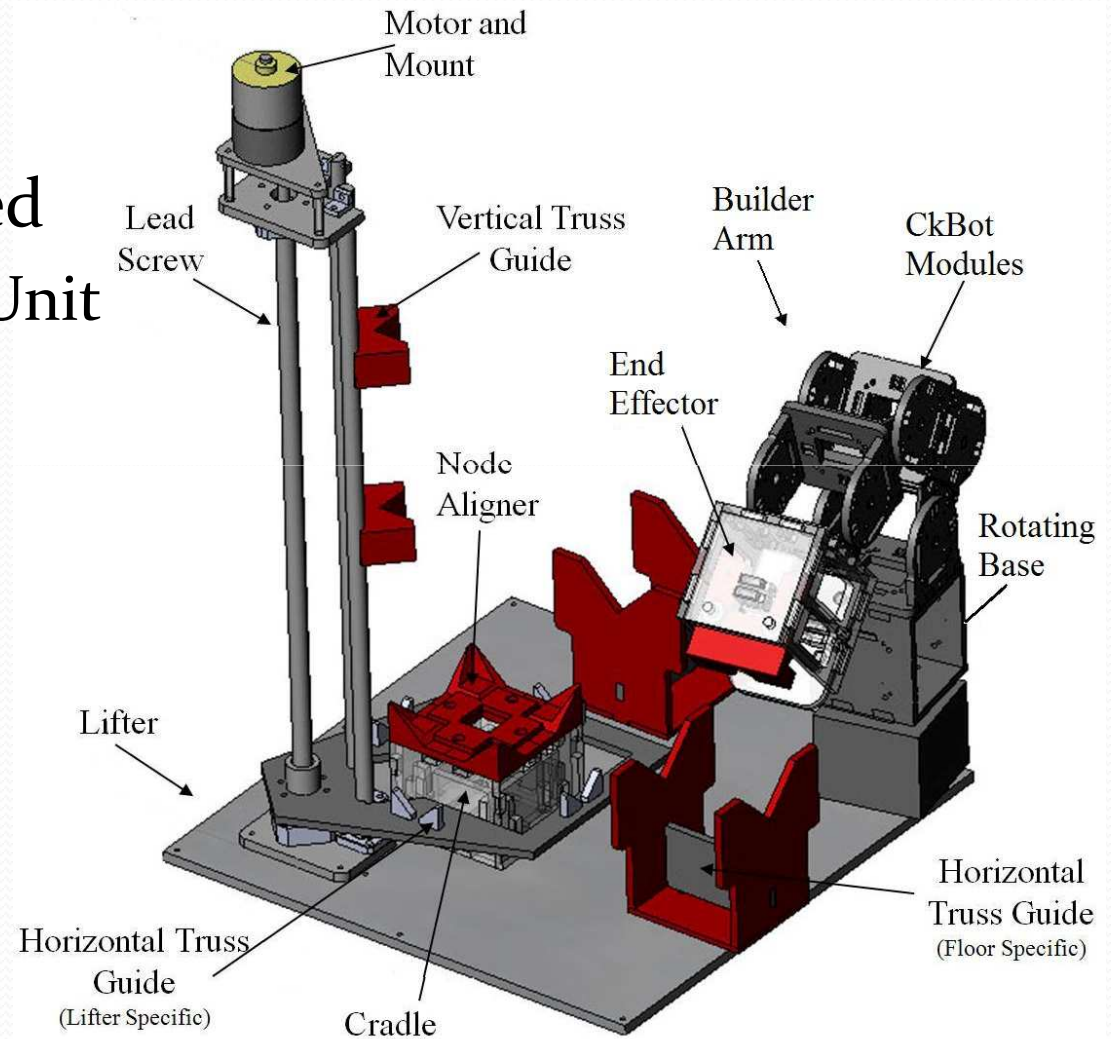
The Customer

- Customer: Eric Klavins
- Associate Professor, Controls and Robotics, UW
- Self-Organizing Systems (SOS) lab



Plant

- Factory Floor Testbed
 - 1 Platform Square Unit
 - 1 Arm
 - 4 Node Cradles
 - 4 Truss Cradles
 - 4 Elevator Posts

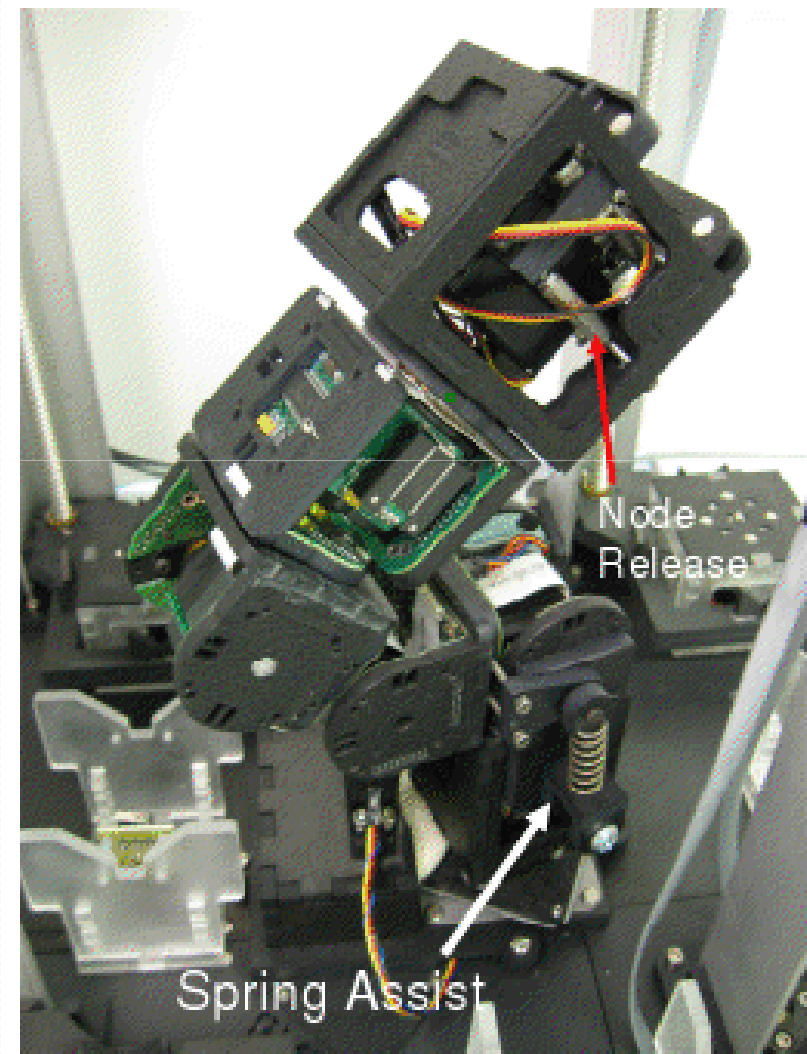


Actuators

- Arm
 - Moves Nodes & Trusses
 - Components:
 - CKBot
 - U-BAR, $\pm 90^\circ$ rotation
 - L-7, $\pm 90^\circ$ rotation
 - Base
 - End-Effector

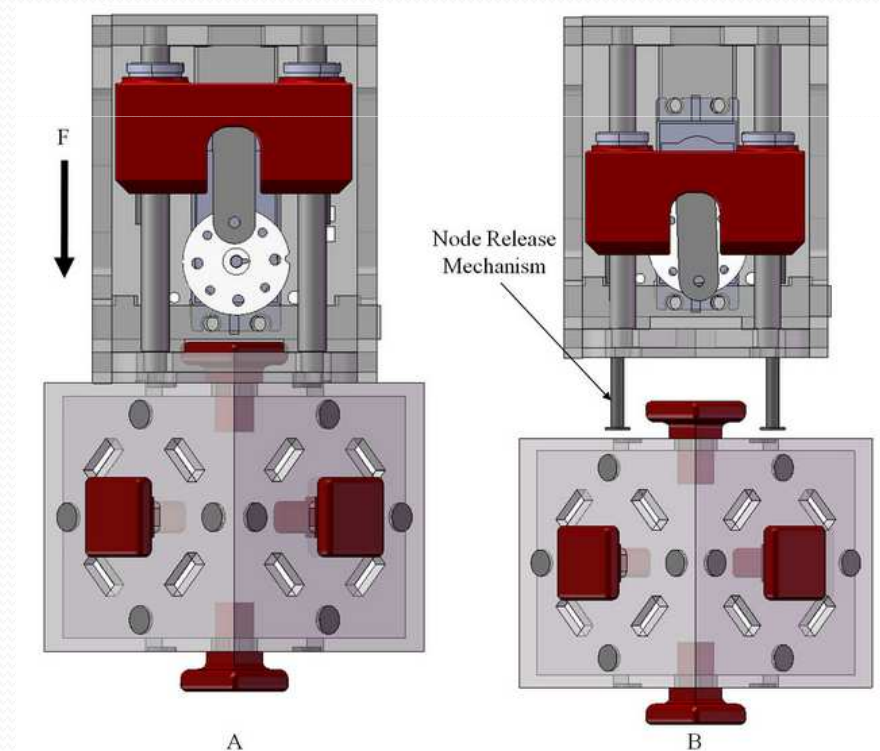


U-BAR (L) and L-7 CKBot Modules



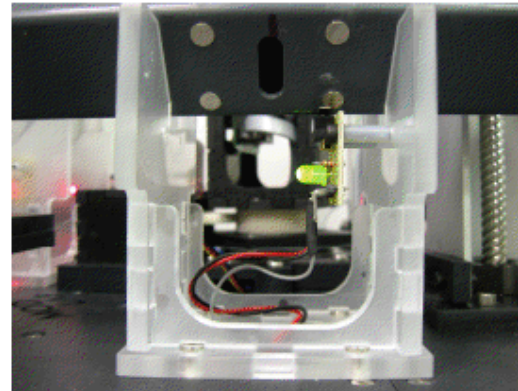
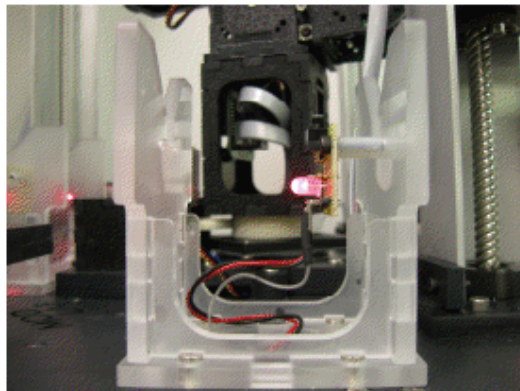
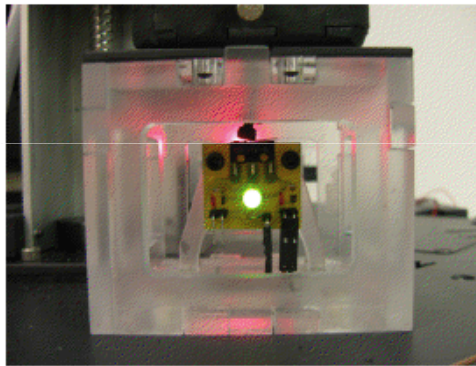
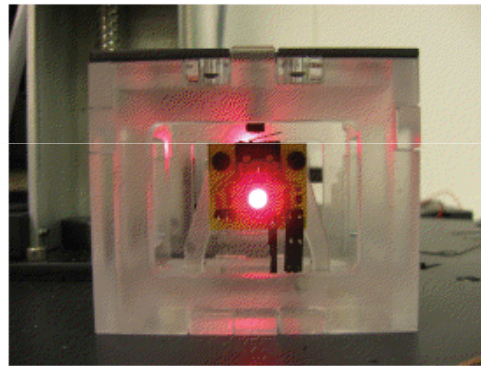
Actuators

- Base
 - Geared Servo Rotates 270°
- End-Effector
 - Dual Purpose Servo
 - Releases Nodes
 - Grabs Trusses



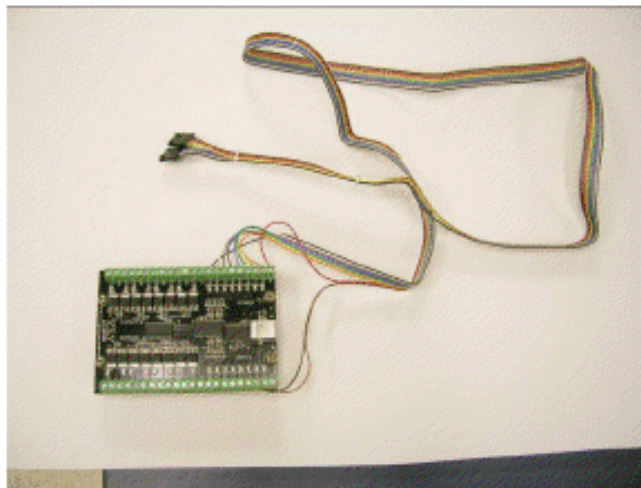
Sensors

- Contact Switches
 - Node and Truss Cradles



Control Resources

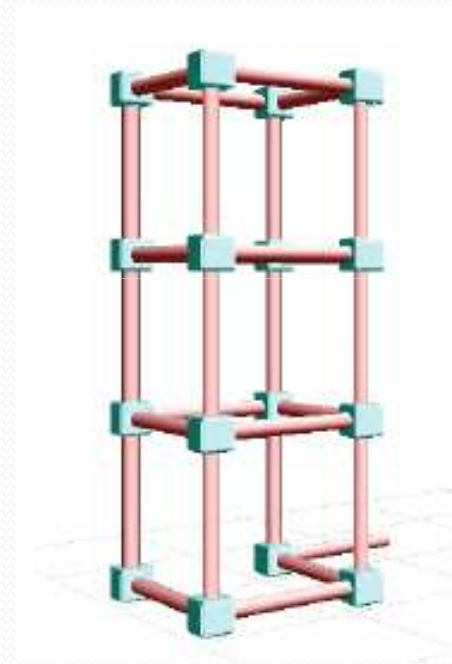
- Control Resources
 - Python/ CKBot GUI
 - PCAN
 - Computational Control Language (CCL)
 - Phidget Board (I/O Board for Sensors)



Development Goals

The functional performance of the system is determined by the ability to:

- Properly place trusses and nodes to construct rectangular sections.
- The ability to construct basic structures such as a tower.
- The ability to react to an error or failure
- The ability to deconstruct a previously built tower
- The ability to communicate and interact with additional robotic arms





Performance Criteria

- Placement of Node ~ 15 seconds
- Placement of Truss ~ 20 seconds
- Construction of Rectangle ~ 2.5 minutes
- Construction of Cube ~ 7 minutes
- Removal of Node ~ 30 seconds
- Removal of Truss ~ 40 seconds
- Deconstruction of Rectangle ~ 5 minutes
- Deconstruction of Cube ~ 15 minutes
- Minimal Failure Rates



System inputs and interfaces

- User Inputs:
 - CCL program (i.e. BuildRectangle.ccl)
- Interfaces:
 - Software: CCL/Python
 - USB to CANbus (250 kbits/sec)
 - USB from workstation to Phiget Board (sensor output)



Cost and Schedule Constraints

- Cost Constraints:
 - Cost associated with this development stage (<\$500)
 - Cost of replacement parts
 - CAN bus, modifications etc.
 - Project is software/algorithm based
 - All hardware is provided by UPenn
- Schedule Constraints:
 - ICRA Competition (5.3.10-5.8.10)
 - Quarter time frame



Project Plan, Milestones

- Build the robot and modules; wire the system to the distribution board (4/2)
- Compile the existing CCL code and libraries (4/4)
- Establish communication between the PC and hardware using CKBot GUI in Python (4/9) *MS1
- Develop path planning, joint angles, ect. (4/13)
- Get CCL to communicate with the hardware (4/16)



Project Plan, Milestones Cont.

- Execute pre-written programs on the hardware (4/25) *MS₂
- Characterize the reliability of the operations and failure rates for specific placements, speed of movement (4/30)
- Develop high level module to build rectangles (5/12) *MS₃
- Nominal demonstration: build a cube (5/14)
- Execute large simulation: where a section of the simulated module is actually built by the single robot acting as if operating with multiple robots. (5/30) *MS₄
- Detect removal of node, and repair structure as needed (6/4) *MS₅

ID	Task Name	Start	Finish	Duration	Apr 2010					May 2010				Jun 2010	
						4/4	4/11	4/18	4/25	5/2	5/9	5/16	5/23	5/30	6/6
1	Build robot and modules; wire system	4/1/2010	4/2/2010	2d											
2	Compile existing CCL code	4/2/2010	4/4/2010	3d											
3	Establish Communication, PC > Hardware	4/5/2010	4/9/2010	5d											
4	Get CCL to communicate with hardware	4/8/2010	4/13/2010	6d											
5	Develop path planning	4/9/2010	4/16/2010	8d											
6	Execute pre-written programs	4/20/2010	4/25/2010	6d											
7	Characterize the System	4/26/2010	4/30/2010	5d											
8	ALASKA! Competition	5/3/2010	5/7/2010	5d											
9	Develop high level module to build rectangles	5/8/2010	5/12/2010	5d											
10	Build a cube	5/12/2010	5/14/2010	3d											
11	Large simulation execution	5/15/2010	5/30/2010	16d											
12	Error detection and correction	5/25/2010	6/4/2010	11d											



Personnel Use

- Each team member is involved in all project aspects, however, “experts” for each component are designated:
 - Stefan: Robot characterization, joint angles, path movement
 - Richard: CCL and PC-Hardware communication
 - Andrew: Algorithm implementation, module construction
- External Resources
 - Nils Napp
 - Fay Shaw
 - Albert Chiu

Related work and bibliography

- MODLAB <http://modlabupenn.org/>



- SOS <http://soslalab.ee.washington.edu/>



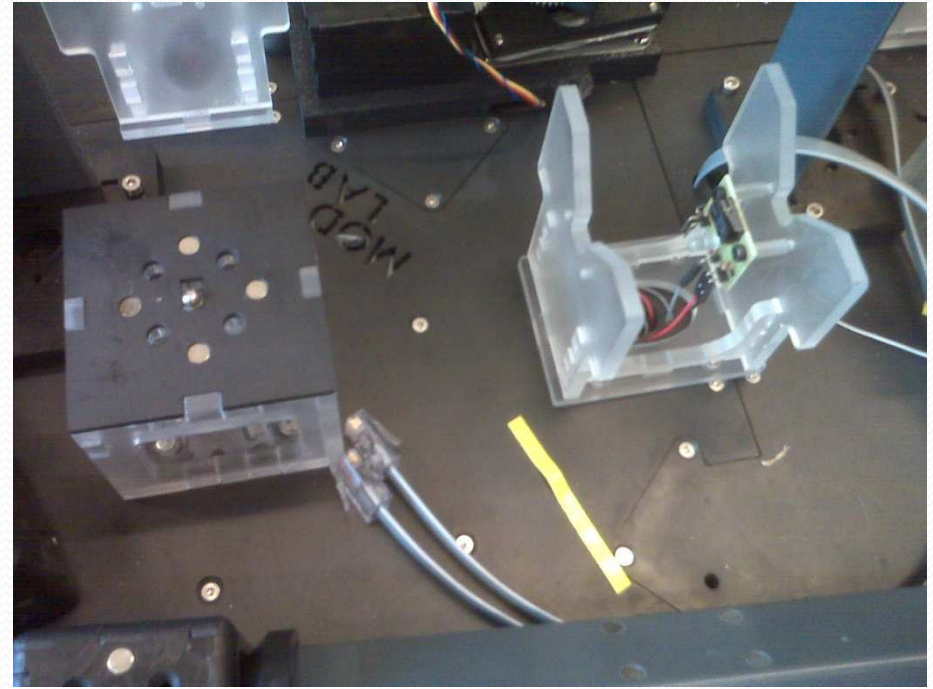
- ICRA <http://icra2010.grasp.upenn.edu/>

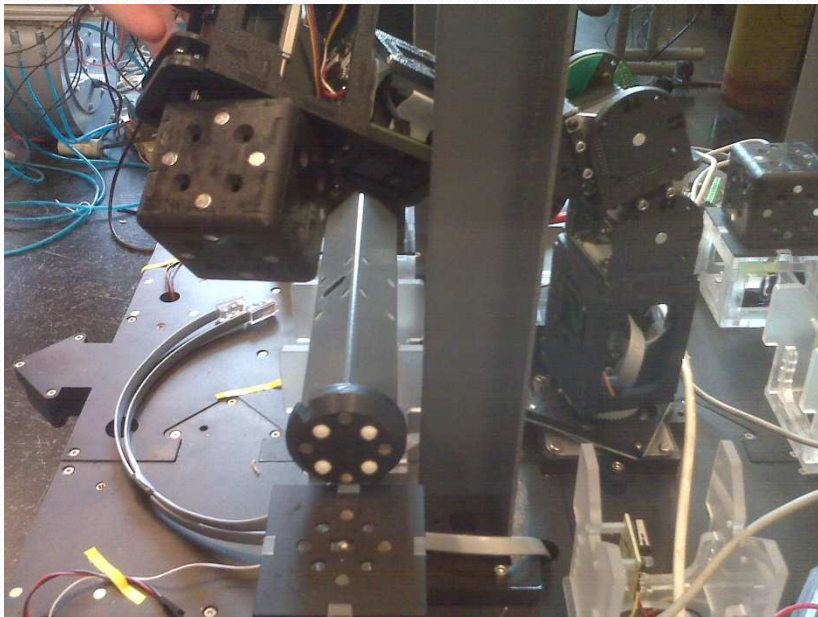


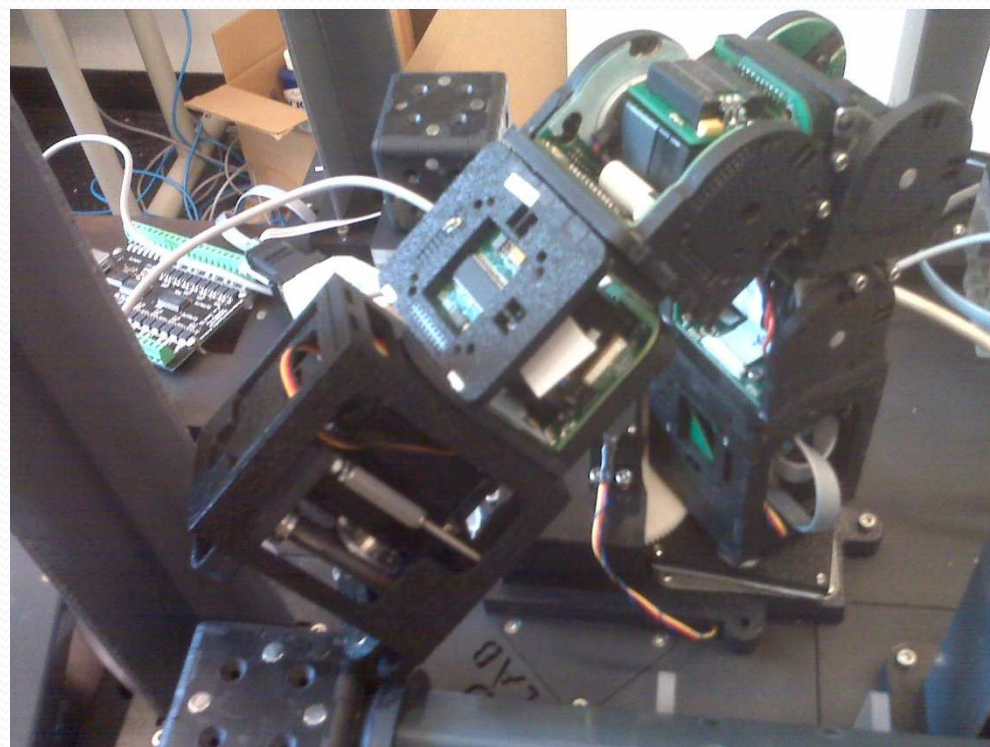
Robotics Competition

- Group is competing at the ICRA Robotic Competition
- Held in Anchorage Alaska, May 3-8
- Robots are built using CKBots, the same modules used to build the robotic arm



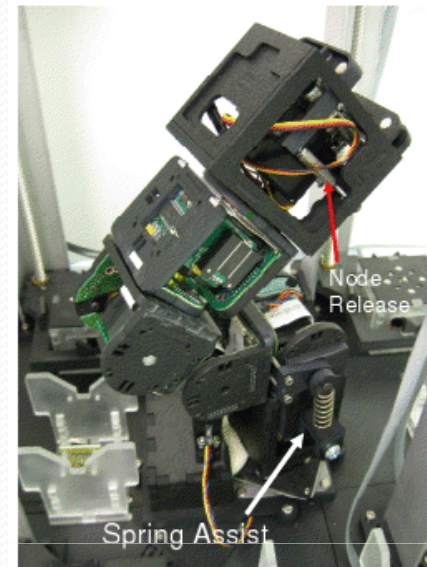






Actuators

- Connector Kinetic roBot (CKBot)
 - U-BAR, $\pm 90^\circ$ rotation
 - L-7, $\pm 90^\circ$ rotation



U-BAR (L) and L-7 CKBot Modules

Property	Value
Mass (per module)	138(g)
Size(per module)	W60xL60xH60(mm)
Batteries	Lithium Polymer 7.4V
MCU	PIC18f2680
Servo	Hitec HSR5990TG
Torque	417 oz.-in.
Transit time	0.17sec / 60 degrees
Reconfiguration	Manual

CKBot Components