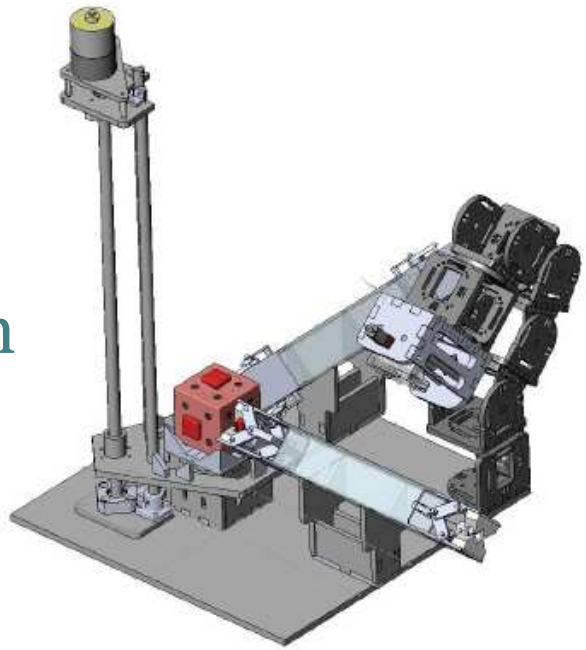
Factory Floor Testbed
MS4

Stefan Kristjansson
Andrew Lawrence
Richard Wood

Project Update

- Currently working on:
  - High level resource management algorithms in CCL
  - C interface from CCL to CKBots
  - Torque minimized path planning using forward kinematics and Matlab simulation developed for MS2.



Achieving Our Goals

- Main objective for the quarter is to complete CCL implementation with high level resource control.
- We are on track to finish this within the next two weeks.
- Concurrently with CCL implementation, we look to finish the torque minimized path planning to increase success rates.
- This will allow for the hardware tile to properly construct its section of the simulation.

Project Demo

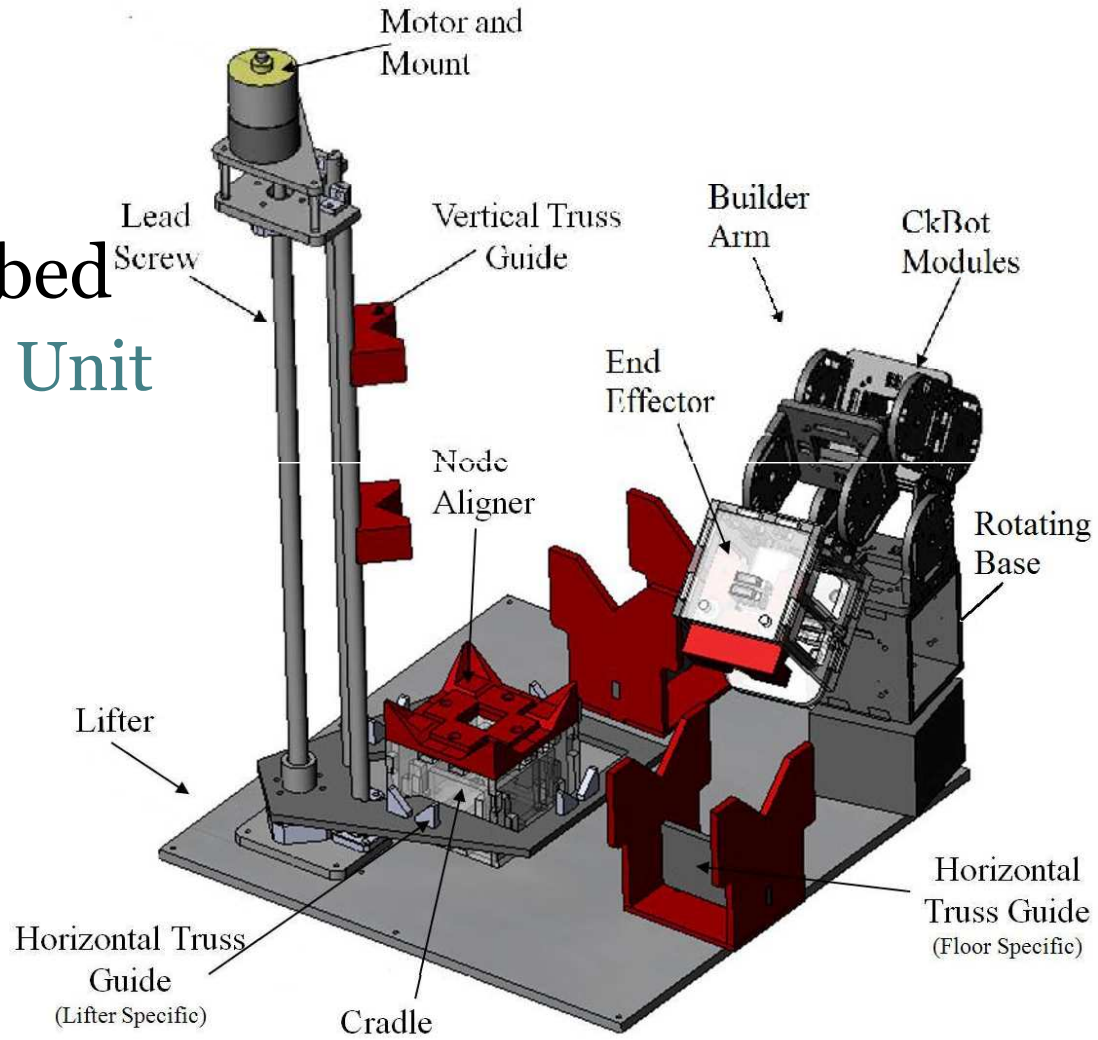
- CCL simulation will construct a multi-tile structure
- Single physical tile will work with simulated tiles
  - Will pass resources to simulated tiles
  - Will properly construct its structure section
- Assembly algorithm will be robust
  - Handle disturbances in resource input
  - Repair breaks/failures in structure

Hardware Details

- Design developed by MODLAB at the University of Pennsylvania
- Slight modifications are necessary to allow the robotic arm to place a truss without a node present
  - This is required to allow a more robust assembly algorithm to be implemented
  - This release can be accomplished through the addition of magnets to the truss cradle

Plant

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



Software - What is CCL

- The Computation and Control Language
- Developed by Professor Klavins
- Useful in distributed algorithms
  - Multiple programs can run in parallel
  - Avoids Critical Sections
    - Larger Expressions are built from shorter expressions
- Allows multiple robots to operate and interact simultaneously from a single program (vs. C)

CCL Basic Structure

- Main program

```
program main() := tileXY(0,0)+tileXY(0,1)+tileXY(0,2)...
```

- Main is composed of multiple smaller programs
- The smaller programs are composed of either a set of guarded commands, function calls, or lists of additional programs

CCL Composition

- Composed of Guards and Commands

- Guard:
  - A boolean expression statement, provides a check on necessary conditions to execute a command
- Command:
  - The desired operation to execute after passing a guard

Functions

- Functions do well defined tasks or manipulations.
- Replace loops with recursion.
- Return values that can be useful for simplifying guarded commands.
- Builds lists

```
fun findEmpty tile i .
    if i < 3 then
        if tile[i] = 1 then
            findEmpty tile (i+1)
        else
            i
        end
    else
        -1
    end;
```

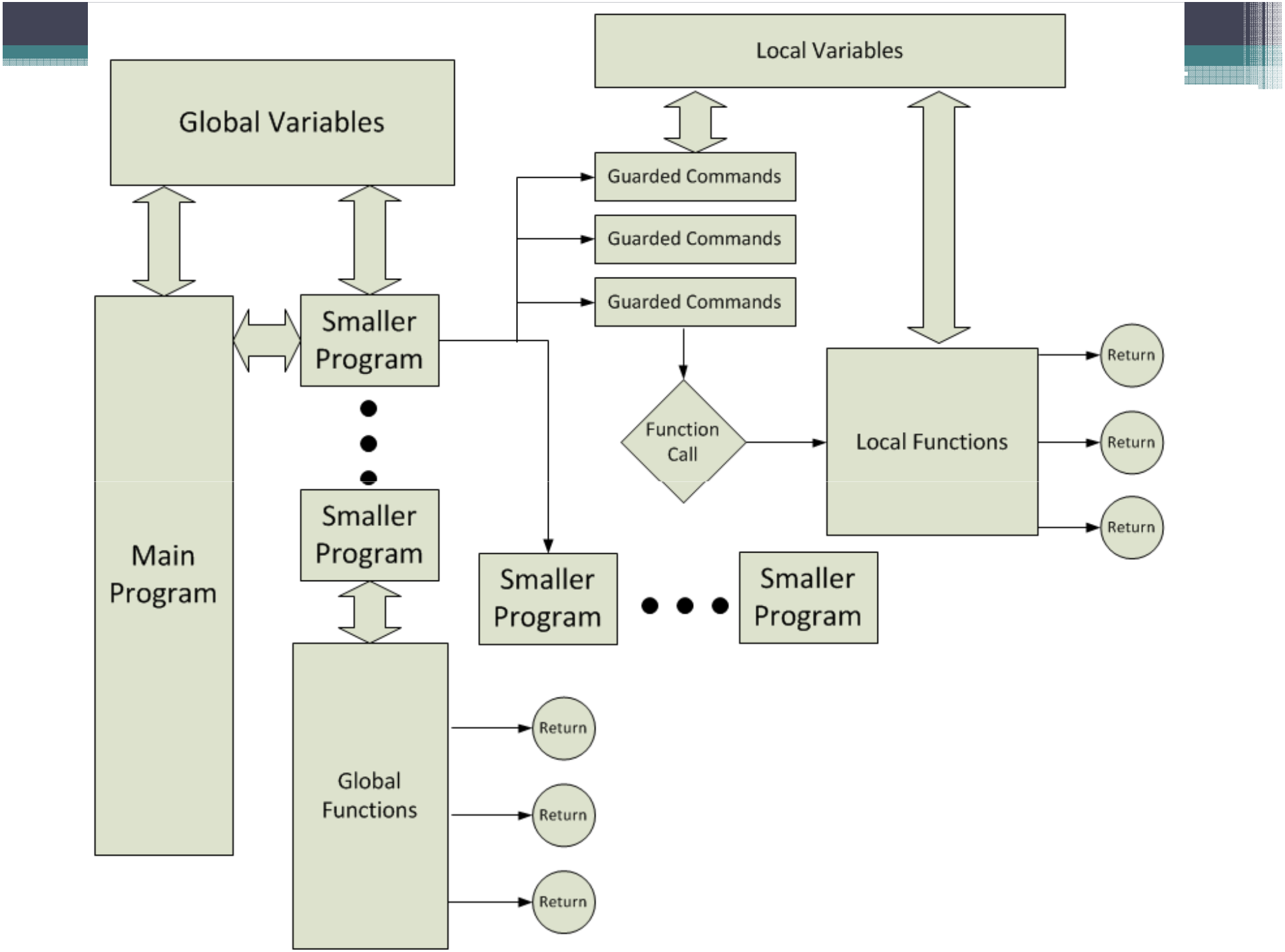
```
fun function parameter .
    if condition then result_A
    else result_B
    end
end;

program PROG1(parameter) := {
    GUARD1 : {
        CMD_A := function ( parameter )
        CMD_B
    };
    GUARD_2A & GUARD_2B : {
        CMD_C
    };
};

program PROG2 () := {
    GUARD_3 : {
        CMD_D
    };
};

program PROG3(parameterX, parameterY) := PROG1(parameterX)+PROG1(parameterY)

program main() := PROG1(parameterZ)+PROG2()+PROG3(parameterX, parameterY);
```


Local Variables
Global Variables
Guarded Commands
Guarded Commands
Guarded Commands
Smaller Program
Main Program
Function Call
Local Functions
Return
Return
Return
Smaller Program
Smaller Program
Smaller Program
Global Functions
Return
Return
Return

```
program assignJobs(x,y) := {
    startup := TotCols;
    // On initialize, make top row builders
    (startup > 0) & (x = 0)  : {
        JOBS[x*TotCols+y] := 1;
        startup := startup -1;
    };
    // Current Tile is Builder, and is Complete, and a next tile in
    // column exists --> pass Builder rights and Floor Complete
    ((JOBS[x*TotCols+y] = 1) & (GAS[x*TotCols+y][FLOOR][2][0] = 1)&
        ((x*TotCols+y+TotCols) < (TotCols*(TotRows-1)+x))) : {
        JOBS[x*TotCols+y+TotCols] := 1;
        JOBS[x*TotCols+y] := 2;
    };
    // Current Tile is Builder, and is Complete, and a next tile in the
    // column does not exist --> pass Builder rights
    ((JOBS[x*TotCols+y] = 1) & (GAS[x*TotCols+y][FLOOR][2][0] = 1)) : {
        JOBS[x*TotCols+y] := 2;
    };

};

program tileXY(x,y) := assignJobs(x,y)+checkCompletion(x,y)+operate(x,y);
```

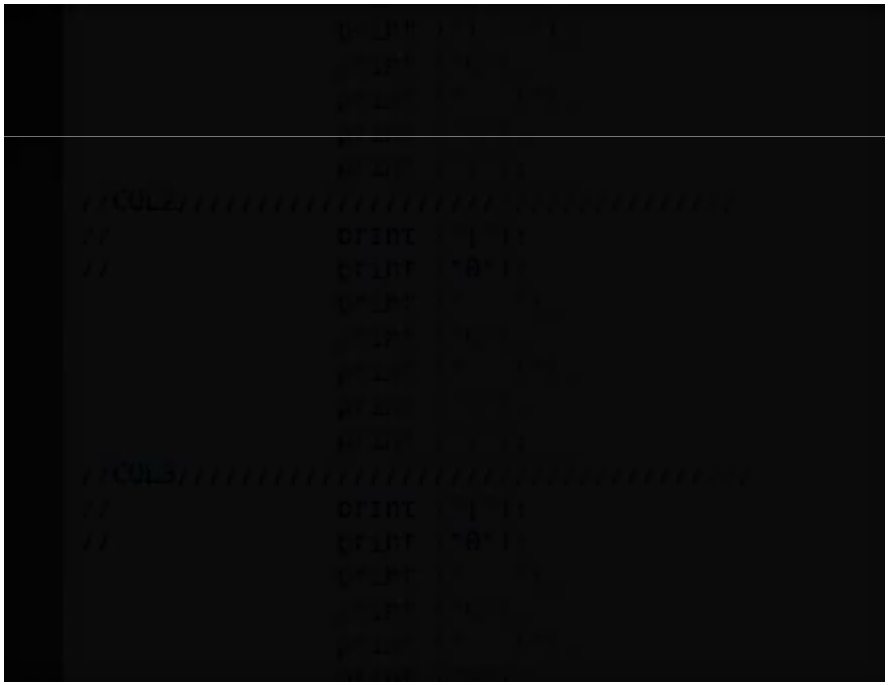
Benefits

- Simple set of guards and commands that can be expanded to distributed systems of arbitrarily large size.
- Programs execute in parallel
- Expressions cannot be interrupted during execution, guards against critical regions.

Drawbacks

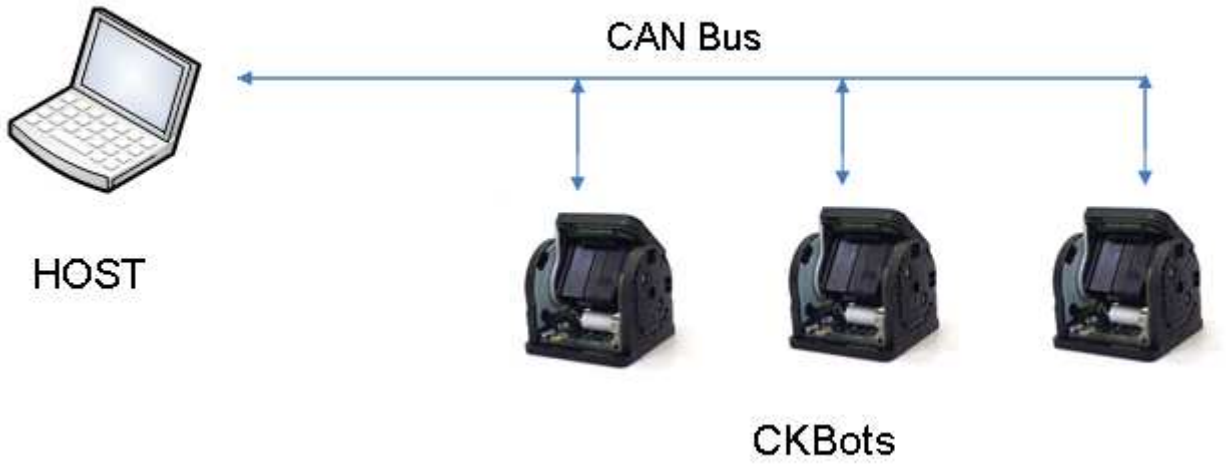
- Initially difficult to program in – much different structure than C or Java
- All larger expressions must be composed of smaller expressions
- Cannot use loops for data iteration - recursion

Factory Floor Testbed Simulation in CCL



Talking to the Hardware

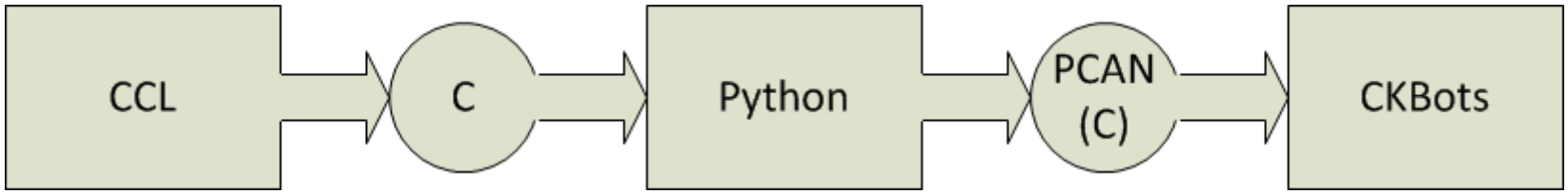
- Computer connects to CKBots through CAN Bus
- Robotics Bus Interface is used to communicate over CAN Bus.



Talking to the Hardware

- High Level Algorithms Implemented with CCL
  - CCL can call C-functions
  - PCAN driver built in C
- UPenn has a software library in Python
  - Implements Robotics Bus
  - Has pre-built commands for CKBots
- Options to connect CCL to CKBot:
  - Re-write Robotics Bus interface in C
  - Write C ‘wrapper’ functions for Python

Full Software Implementation



- Keep High-Level algorithms in CCL
- Keep Low-Level communication in Python
- Couple CCL and Python together with a C-interface
  - Build wrapper functions

Bibliography

- *(1) CCL: The Computation and Control Language. Retrieved April 05, 2010, from University of* Washington, Self Organizing Systems Lab website, http://soslab.ee.washington.edu/mw/index.php/Code
- *(2) Phidgets. Retrieved April 13, 2010, from Phidgets website, http://www.phidgets.com/*
- *(3) Mason, Matthew. (2001). Mechanics of Robotic Manipulation. Massachusetts: The MIT Press.*
- *(4) Modlab CKBot Graphic User Interface Manual. Retrieved April 10, 2010, from UPenn, Modular* Robotics Laboratory website, http://modlabupenn.org/efri/
- *(5) M. Yim, P. J. White, M. Park, & J. Sastra, "Modular Self-Reconfigurable Robots", 2009,* pp. 5618-5631.
- *(6) Nurrat, Richard, & Li, Zexiang, & Sastry, S. (1994). A mathematical introduction to robotic manipulation. Florida: CRC Press.*
- *(7) Craig, John J. Introduction to Robotics: Mechanics and Control.(1989) Reading, Mass.: Addison*-Wesley.
- *(8) IPython Documentation. Retrieved April 12, 2010, from IPython website,* http://ipython.scipy.org/moin/
- (9) D. Gomez-Ibanez, E. Stump, B. Grocholsky, Vijay Kumar, & C. Taylor. The Robotics Bus: a Local Communications Bus for Robots. In *Proceedings of SPIE,* Volume 5690. 2005.