Design • Build • Program • Learn • Compete

SERVO

FOR THE ROBOT INNOVATOR

www.servomagazine.com

MAGAZINE

December 2011

PR LITE

Four High School Roboticists Roll Their Own PR2

✦ **Wonder Wheels**
Put a new spin on your robot base with omni or mecanum style wheels.

✧ **Pimp My Hexapod**
A Frankenstein mixture of parts is transformed into a state-of-the-art six-legged walker.

PR Lite — Build Your Own PR2 Running ROS

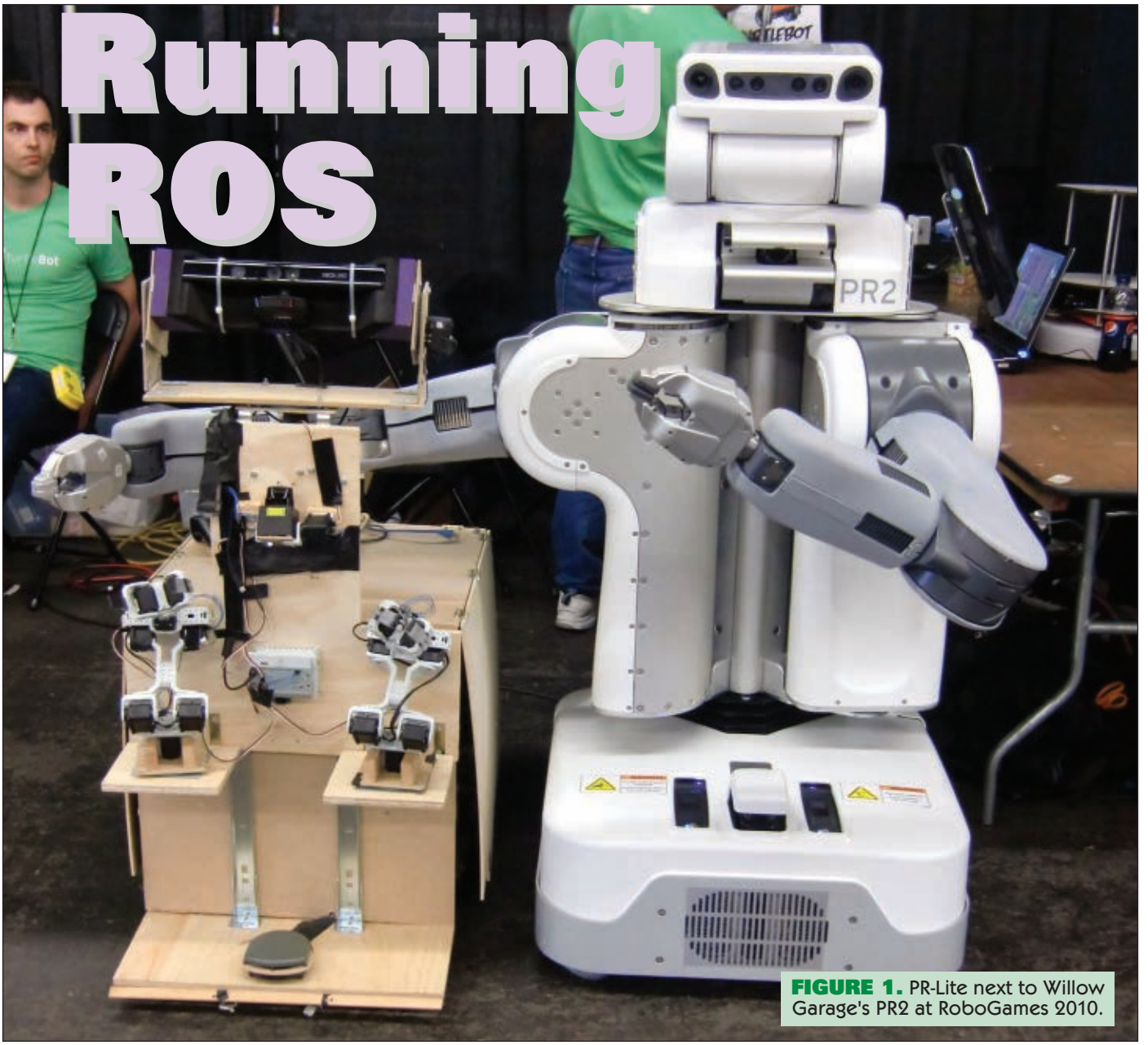

FIGURE 1. PR-Lite next to Willow Garage's PR2 at RoboGames 2010.

by Alan, Andrew, and Matthew Downing, Nathaniel Lewis, and Frank and Robert Ou

How would you like a robot that can play pool, fold towels, and bring you a drink from the fridge? We'd like one too, but don't want to spend $400,000 for a PR2 from Willow Garage. PR2 is built with high-end components so that research on robotics software can progress in anticipation of ultra-low cost components becoming available down the road.

In many ways, that day has already arrived, as embodied by the $150 Kinect and the low cost LIDAR used by the Neato robotic vacuum cleaner. These new sensors and servos with feedback — like the Dynamixel AX-12+ servos — enable a personal robot with PR2-like capabilities to be within reach of a robotics club. Furthermore, the Robot Operating System (ROS) software provides much of the intelligence and glue to make this desire a reality. Such was our goal when we combined the resources and capabilities of four high school roboticists and their mentor fathers to produce PR Lite.

Comparison With PR2

As shown in **Table 1**, the capabilities of PR Lite are analogous to specifications of the PR2, but are lower quality.

	PR Lite	PR2
Head Camera	Kinect (two cameras with LED texture projector for 3D vision)	Three cameras with LED texture projector for 3D vision
Base LIDAR	Neato	Hokuyo UTM-30LX
Tilting LIDAR	Hokuyo URG-04LX-UG01	Hokuyo UTM-30LX
Arm DOF	5 (shoulder pan, tilt, elbow, wrist rotate, left finger, right finger)	7 (includes forearm and upper arm roll)
Arm Payload	2.5 lbs	4 lbs
Number of Arms	Two	Two
Drive System	Forward with differential steering; spin in place; sideways with differential steering	Holonomic
Torso	12" telescopic	12.3" telescopic
Computer	i7 quad-core with hyper-threading, 6 GB memory, 60 GB solid-state drive	Two i7 quad-core, 24 GB memory, two TB disk space (all 2x)
Network	Eight USB ports, I²C	EtherCAT
Batteries	Five 12V lead-acid; 30 min runtime	Lion; Two hour runtime
Software	ROS	ROS
Joystick Control	Yes	Yes
Wireless	Yes	Yes
IMU	SparkFun six DOF*	Microstrain 3DM-GX2
Arm Cameras	Webcam*	Ethernet camera

* Part currently not installed on robot.

TABLE 1.

Open Source Hardware and Software

Our team experience has proven valuable for implementing the initial version of PR Lite. We hope that our efforts will make it easier for others who try to build their own PR Lite. Like with PR2, we embrace the open source philosophy. Our software, schematics, URDF (Universal Robot Description Format) definitions, and CAD drawings are all available online. All other parts are commercial off-the-shelf (COTS) hardware. The other key software is ROS which is open source. PR Lite's PC and AVR code are in online Git version control repositories hosted by Github.

ROS

ROS contains a plethora of functionality that continues to expand due to a large open source community. At its heart, ROS is a message passing system which allows robotics software to be more modular and reusable by defining and allowing the community to define standard interfaces. As a result, many robot components now have ROS interfaces. For example, ROS has Simultaneous Localization And Mapping (SLAM) capability that can be used by both PR2

FIGURE 2. Robert Ou, Andrew Downing, Matthew Downing, and Nathaniel Lewis work on software and CAD drawings for PR Lite.

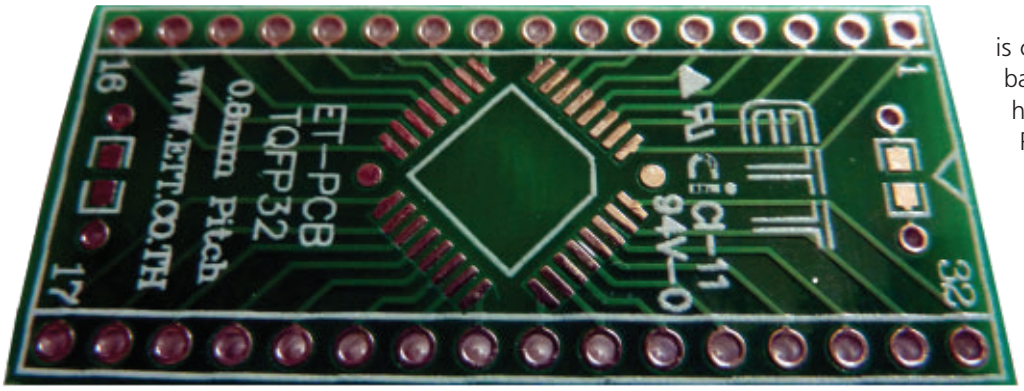

FIGURE 3. Example of breakout board used with the ATmega chips.

and other robots. ROS has drivers for Hokuyo LIDARs, AX-12 servos, the Sparkfun IMU, and many other hardware devices. To help overcome a large learning curve, ROS has many tutorials to explain the functionality available, but these tutorials take time to run, understand, and retain.

After getting over a significant initial learning curve, we were able to easily integrate the Kinect and the Neato LIDAR soon after their release. The Kinect uses the OpenNI library and PrimeSense's NITE library to find a subject's skeleton. This skeleton is transformed into spherical coordinates (with a radius), and inverse kinematics are used to position PR Lite's arms. ROS packages from the University of Arizona for their CrustCrawler arms are customized to control the AX-12+ servos for our very similar arm configuration. Existing drivers for the Hokuyo and Neato LIDARs are used.

PR Lite's ROS navigation stack is customized to handle its unusual base controller to emulate holonomic drive and use the Parallax odometry information. The pr2_teleop_general driver is modified so that PR Lite can be fully controlled by a PS2 joystick using an aftermarket adapter that can easily be found online (rather than the PS3 joystick using Bluetooth).

The ROS simulation software also has a large learning curve. Manipulating XML files to create the robot model is tedious. Our robot is still undergoing significant transformation, so we decided to concentrate more on the design and building of the physical robot. While in theory, the simulation allows software development to continue in parallel with modifying the physical robot, the amount of time to get meaningful results from the simulation is almost as high as getting the physical robot up and working. We eventually put our simulation work on hold so that we could complete our demo for RoboGames.

In retrospect, the amount of software that we wrote to get PR Lite to perform various complex capabilities such as navigation and mimicking human arm movements is surprisingly small. However, the amount of software that we had to understand to get to this point was huge. In the longer term, we'd like to be able to reuse higher level code to perform PR2 demonstrations, but we'll have to modify many of the PR2 services to generalize numerous PR2-specific assumptions.

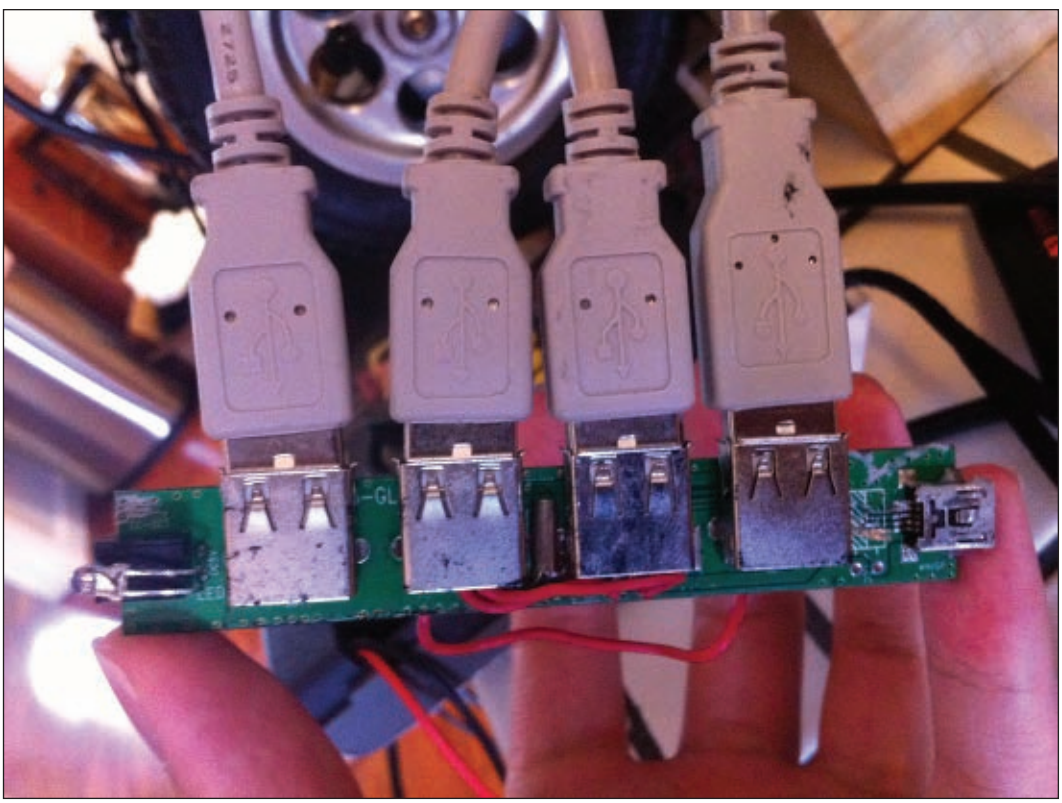

FIGURE 4. USB hub that was modified to connect I²C devices.

Hardware Overview

PR Lite is actually a conglomeration of some of our previous robots. Our RoboMagellan robot — which won silver at RoboGames 2010 — was gutted for parts including its two pairs of Parallax wheel kits, HB-25 motor controllers, and its sensors. A Bioloid comprehensive kit

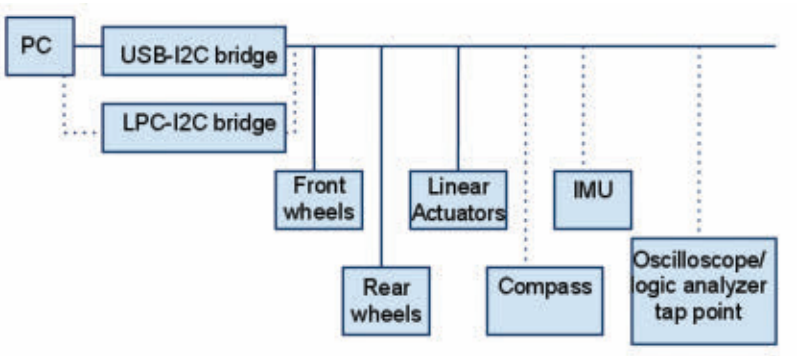

FIGURE 5. Devices that are attached to the I²C communications bus (dotted lines represent devices that are not currently installed).

was repurposed from a newer incarnation of Pooh Bear Bot. The Dynamixel AX-12+ servos have position, temperature, voltage control, and feedback, and are used for arms and pan/tilt functionality. A LIDAR was removed from a used Neato that was purchased on eBay. Our RoboMagellan robot was controlled by an OLPC XO-1 which performed poorly when task switching and experienced high latency when accessing USB devices. Learning from our mistakes, PR Lite is equipped with an i7 quad-core processor and other spare computer parts to make our high-end "Frankenstein computer."

PR Lite is designed to offload all timing-critical functions such as PID speed control loops, linear actuator controllers, and the emergency stop capability onto microcontrollers rather than using the Linux kernel real time capabilities like PR2. All our microcontrollers use the ATmega328p because it is sufficiently powerful, used in the Arduino, and has hardware support for I²C. The TQFP surface-mount version is used because we intend to manufacture custom PCBs. Cheap TQFP breakout boards are used for prototyping.

To interconnect the microcontrollers and the PR Lite computer, we chose I²C. However, instead of using I²C in the traditional device and register protocol, it is used to transmit packets of data like UDP. Because the I²C repeated start condition effectively allows a node to hold the bus — forcing other nodes to wait — support for request-response packets is also added.

We utilize the wire-OR nature of I²C to construct a low latency multi-master bus. All the microcontrollers can transmit broadcast or unicast packets onto the bus at any time. The wire-OR allows collision detection, and packets are re-transmitted. The bus runs at 400 kHz.

Each packet has a checksum. If the checksum is wrong, the receiver will discard the packet. The transmitter should then re-transmit the packet, but this has not yet been implemented. The theoretical latency of this bus is in the range of milliseconds which was considered low enough.

I²C operates at the TTL level and is susceptible to noise. We place all the microcontrollers close together with short wires to reduce the noise interference. We use modified USB cables which are shielded for this purpose. We modified a USB hub to be our I²C bus hub by removing the USB hub IC from it and soldering wires to form an I²C bus inside the USB hub chassis. The noise interference is low with this construction but the USB hub IC is surprisingly fragile and unreliable.

To connect the i7 central computer to the microcontrollers, we used a USB-to-serial module to connect to yet another microcontroller — the bridge controller. This bridge controller takes one mega baud serial from the PC USB module and converts the commands to the modified I²C bus for all other microcontrollers. The bridge controller also receives all the status from the microcontrollers and relays them back to the PC.

Drive System

The Parallax wheels are distance controlled. These needed to be modified to be velocity controlled to be compatible with ROS. The Flash in the microcontroller embedded in the Parallax wheel is near full, so we added our own faster microcontroller with larger Flash for velocity and PID control. To utilize the existing Parallax wheel encoders, we did not remove the wheel controller board. Instead, wires are soldered onto the existing optical encoders and connected to our own microcontroller board, bypassing the Parallax controller.

We use two microcontrollers to control four wheels; one for the front wheels and one for the back wheels. Each wheel is individually controlled by the central i7 computer. The long wires from the optical encoders pick up noise from the motors and can cause problems for our microcontrollers. We use optoisolators to isolate the encoder wires from the microcontroller digital inputs.

The two linear actuators are controlled by one ATmega328p microcontroller. We use two relays for controlling each of the two linear actuators. The two relays allow the microcontroller to provide positive 12V, negative 12V, or no power to the linear actuator. The microcontroller can control the linear actuator to move in forward direction, in reverse direction, or to stop.

The linear actuator has a variable resistor as a feedback mechanism. The variable resistor is connected to the input of the microcontroller's built-in analog-to-digital converter. By reading the A/D value, the microcontroller software constantly monitors exactly where the linear actuator is positioned. When the i7 central computer programs or commands the linear actuator microcontroller to move the linear actuator to a desired position, the microcontroller drives the relays to provide proper power to the linear actuator. When the linear actuator reaches the desired position, the microcontroller tells the relays to provide no power and thus stop the linear actuator at the desired

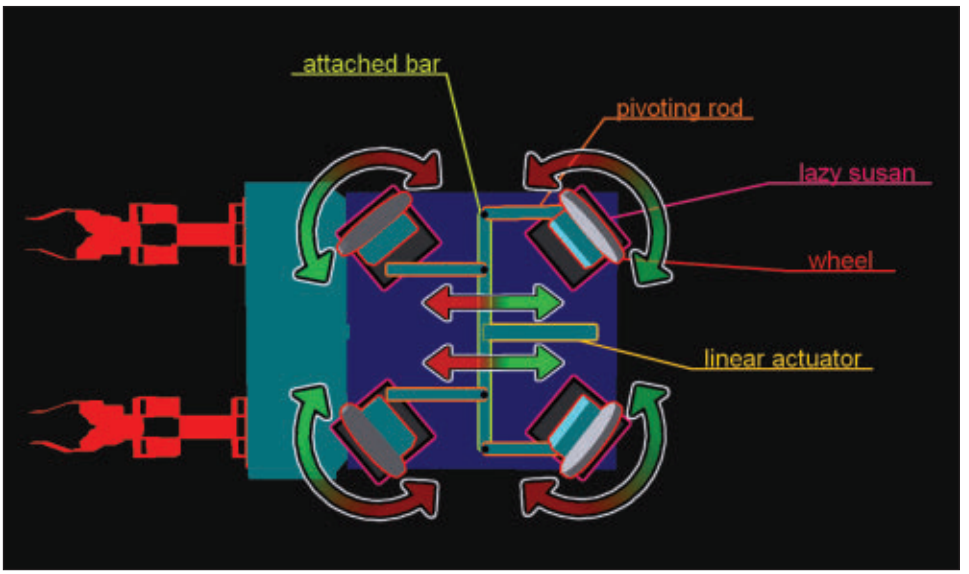

FIGURE 6. Diagram of the mechanism used to rotate the wheels.

position. Unfortunately, the relays and the linear actuators generate a lot of electrical noise. To help reduce this, the control of the relays from the microcontroller is opto-isolated. A separate DC/DC power circuit is used to provide power to the relays.

Many different power adapters are needed to power different components. Originally, we tried to build DC/DC power sources for each voltage type. Later, we decided to just use AC power for many of the components via a 200W DC-to-AC inverter. We can connect the battery to this inverter and allow many components to run with their original AC power adapters. The i7 computer runs on its own 24V DC power supply, which is powered by two 12V lead acid batteries.

The Base and Torso

We used wood for the base of our initial PR Lite prototype. Wood is cheap and easy to cut and drill with hand tools, allowing us to tweak the design continuously. The base contains a bottom shelf with a tongue that extends beyond the main cabinet and holds the Neato LIDAR. A telescoping torso is attached to the front side of the cabinet.

Inside the main cabinet are three shelves. The bottom shelf contains the rotating wheels, a linear actuator, and batteries. The top and middle shelfs hold the computer, all the microcontrollers, the DC-DC and DC-AC converter, and the Wi-Fi router. The Parallax wheels are encased in acrylic structures modeled with CAD software which are each attached to a 4" lazy susan. The four wheels are all simultaneously pivoted by a single bar attached to a linear actuator. The linear actuator has a 4" throw with feedback and is controlled by a microcontroller. An H-bridge is formed via relays and transistors. As illustrated in **Figure 6**, the wheels are pivoted in place to one of three positions for the wheels to move forward, pivot in place, or move sideways. When moving forward or sideways, differential drive is used for further turning control. The wheel controller will not simultaneously move the wheels while pivoting them via the linear actuator.

The torso is attached to the base via two 12" drawer slides and can be moved up and down using a linear actuator. Attached to the top of the torso is a Kinect that can pan and tilt through a pair of servos and three lazy susans. The arms are made from a Bioloid comprehensive kit and are constructed to be similar to two Crustcrawler arms. Each Bioloid arm is attached to a shelf on the torso via a lazy susan which is rotated by a servo. A 4" lazy susan provides a stable base that can handle a heavy load so that the servo only has to deal with the rotational force. The servo hub is centered within the hole of the lazy susan and attached to a shelf on the telescoping torso.

Because the AX-12+ wires are daisychained, one wire with poor connectivity can make the whole downstream chain perform poorly or not function at all. The wires in our Bioloid kit are modified to be customized lengths. Many needed replacement until we produced reliable arm performance.

Conclusion

Our completed PR Lite prototype was demoed at RoboGames 2011 and featured PR Lite's arms mimicking a person's arms. Many enhancements are planned to make PR Lite even more like PR2 in both dimension and capabilities. The main addition will be to add upper arms, each with pan capabilities (using two lazy susans with dynamo servos) and powerful lift capabilities (using a linear actuator). Using our many lessons learned, our next iteration will be fully sketched out using CAD software and made from laser-cut ABS at the TechShop. We also want to harden the prototype against loose wires and bolts by fabricating some boards, and replacing many of the Bioloid construction kit parts with simple bent aluminum bars.

Much of the remaining work is software. The robot torso will be adjusted so it's similar in shape and dimensions to PR2, allowing us to more easily reuse code intended for PR2. We will try to hook PR Lite arm planning directly into the PR2 arm planning. We'll revisit the simulation for the next iteration of PR Lite (PR2 Lite). We can envision PR Lite eventually being able to perform many PR2 capabilities. We look forward to doing our own hackathons! SV