Internet of Toys: Julia and Lego Mindstorms

Robin Deits

October 21, 2015

Abstract

The latest version of the Lego Mindstorms robotics system is an excellent candidate for the exploration of distributed robotics. I plan to develop a set of Julia bindings for the low-level motor and sensor interfaces. Using those bindings, I will demonstrate the use of Julia's built-in support for parallel computing by creating a simple collaborative robot mapping system.

1 Background

Versions 1 and 2 of the Lego Mindstorms system (the RCX and NXT) were surprisingly capable and flexible toys, but were fundamentally limited by the underpowered hardware and proprietary onboard software. Fortunately, the situation has been radically improved with the newest Lego Mindstorms release, EV3. With EV3, the brick (the box which houses the battery, computer, screen, and interface ports) is now a small Linux computer complete with USB and MicroSD ports. This means that it is now possible to install and even develop software directly on the EV3 using standard Linux tools. Even better, the EV3 is capable of booting directly from the MicroSD port, which opens the door for custom operating systems.

I will be building my work on top of ev3dev, a custom Debian Linux distribution built specifically to run on the EV3. The ev3dev distro makes communicating with the basic EV3 hardware particularly easy by mapping sensors and motors directly to nodes in the filesystem. For example, instructing a motor to run continuously is as simple as:

```
echo run-forever > /sys/class/tacho-motor/motor2/command
```

The developers of ev3dev also provide higher level bindings for c++, python, lua, and node.js. Rather than building Julia wrappers for any of these language bindings, I will write a new set of bindings directly on top of the filesystem interface provided by the ev3dev OS.

2 Scope

2.1 Low-Level Bindings

I plan to, at minimum, support reading data from the standard Lego Mindstorms EV3 sensors, writing speed and position commands to the stock motors, and reading the motor tachometers from Julia. This should actually not involve all that much new code: the entire c++ bindings provided by ev3dev comprise only 1200 lines of code, much of which is devoted to features like controlling the LCD screen, built-in LEDs, and sound generation. I consider those features to be lower-priority; including Julia bindings for them would be nice, but is not necessary in order to demonstrate the use of Julia on EV3.

2.2 Demonstrating Collaborative Robotics

Once the low-level bindings are functional, I plan to create a simple demonstration of the power of Julia for collaborative robotics. I plan to build two simple Lego vehicles, each with a Lego ultrasonic distance sensor mounted onboard. Each Lego vehicle will run the ev3dev OS with Julia installed locally. A master computer will connect to the Lego robots as part of a standard Julia ssh-based cluster. The master will then instruct all available robots to execute a mapping behavior, in which each robot drives around in a random pattern, collecting range data from the distance sensor and tracking its position via dead reckoning. The master will then wait for the robots to complete their tasks using a standard Julia remotecall_fetch(), and then assemble a rough map of the world from the data collected by the robots.

Having worked in robotics for a few years, I am well aware that dead reckoning and ultrasound sensors are likely to provide a very poor map of the environment, but I'm confident that I can produce a system which is good enough to demonstrate its purpose, and that I can show that controlling robots through remotecall() is a viable idea.

3 Hardware Requirements

To complete this project, I will need at least some access to Lego Mindstorms hardware. The ideal hardware setup would be two complete Mindstorms EV3 systems ($350 each from Lego Education), two MicroSD cards to install the ev3dev OS ($6 each from Amazon), plus two WiFi USB adapters to allow them to act as part of a Julia cluster even when untethered ($35 each from Amazon), for a total cost of approximately $782.

If this is too expensive, then there are many options to scale down the cost of the project. Removing one entire EV3 system cuts the cost of the project in half, but makes it difficult to demonstrate any sort of parallel or collaborative application. On the other hand, the ev3dev OS can be run directly on a Raspberry Pi (available for around $40) instead of an EV3, but additional work will be required in order to connect any sensors or motors. I would be happy to research additional options as needed.