Designing and building Logobot

Last edited August 6, 2008

Abstract

Logobot is a small, affordable robot which understands the programming language Logo. Logo uses simple commands like `forward`, `right`, `left`, `pen up`, `pen down`, and so on.[1] to control a virtual or physical sprite. Logo aims to make programming physically intuitive by capitalizing on the way we already think about motion to introduce advanced programming, mathematical, and epistemological concepts. The pedagogical versatility and value of the language was explored in Seymour Papert's *Mindstorms*.

Logobot's size and cost will make it a candidate for exploring swarm computing and emergent phenomena in a physical environment. Currently, swarm computing is prohibitively expensive and relatively unapproachable. Environments like StarLogo[2] have made it possible (though not particularly easy) for students to play with complex, emergent systems. Logobot can bring physicality to the process.

Logobot will interface with the Scratch programming environment[3], which means there will be a tremendous market primed for using the tool. Through the plugin we write, people familiar with Scratch will be able to write programs for Logobot in Scratch and run them.

We have not yet confirmed a sponsor school or classroom, but we will do so in the coming week.

1 What are we doing?

We are designing a small, affordable robot that can easily opearate on a desktop and obey Logo commands which are either loaded onto Logobot or run wireless from Scratch. We will be developing a plugin for Scratch that allows us to add commands for Logobot's operation. We will also be creating documentation and support materials for teachers and students interested in constructing or incorporating Logobot into their educational programs. We will

2 Who are we working for?

TO BE DETERMINED

3 What are their needs?

TO BE DETERMINED

[1] `http://en.wikipedia.org/wiki/Logo_programming_language`
[2] `http://education.mit.edu/starlogo/`
[3] `http://scratch.mit.edu`

4 How does it help us?

Logobot is a marketable product and kit that targets a well-established sector: educational robotics. It has the potential to be an extensible platform for robotics to which we can add options of sensors and software at a later date. It is a natural, engaging choice for a product around which we could grow a community of practice that modifies, extends, and evangelizes our product and name.

There are many contexts in which Logobot is a prepackaged educational solution. Logobot would provide us with an easy option to create a program that focuses on any of the following, on-the-fly:

- Constructing Logobot from scratch is a great context for teaching basic electronics
- Logobot as a Scratch addon opens up the possibility of Scratch workshops with MIT and the Lifelong Kindergarten group[4]
- Teaching programming to students in the classic *Mindstorms* style
- Robot competitions
- Teaching math and geometry at any grade or school level with Logo
- Teaching college students differential geometry (see Abelson and di Sessa's *Turtle Geometry*[5]
- Community workshops and demos at venues like the Museum of Science

Because the educational angle to robotics is so well-established, there are many existing channels into which we could plug Logobot to advertise and expand the NUBlabs name.

Furthermore, if we can document the unique educational opportunities Logobot offers, we will position ourselves as a leader in the field by capitalizing on existing connections to educational media outlets.

5 When does this need to be done?

December 1st 2008: we want the product and kit ready for the holidays and for schools' spring terms.

6 What skills are needed to make this work?

- **Mechanical engineering**

  Logobot needs to be durable, robust, affordable[6], and buildable.

- **Electrical engineering**

[4] `http://llk.mit.edu`

[5] `http://www.amazon.com/Turtle-Geometry-Mathematics-Artificial-Intelligence/dp/0262510375/ref=pd_bbs_sr_1?ie=UTF8&`

[6] Perhaps this means that it is designed with as few parts as possible.

Logobot's electronics need to be designed with cost and the kit-builder in mind. They should be designed as an extensible platform, not a typical, immutable, embedded device. Logobot needs to be able to navigate arbitray surfaces, translate high level language into movement commands, and receive commands in real time.

- **Software programming**

  A Scratch plugin needs to be written to add the necessary functionality to Scratch to allow for the programming of Logobot from Scratch. A web site needs to be set up to support the community that will grow up around the product, as well as a way to share and document projects.[7]

- **Clear writing and documentation skill**

  This product has been created before. What will distinguish NUBlabs' implementation is our integration with an existing product and its community (i.e. Scratch), and the quality and extent of our support materials (not to mention the degree to which we advertise them).

7 What materials are needed?

- PCB
- Electronic components
- Chassis
- Motors
- Wheels and other moving parts
- Sensors
- Memory
- Batteries
- MORE?

8 What will be the material cost of development?

-

[7] This starts to touch on some broader applications that need to be made to support DIY activity. That means that this piece of software should not be on the critical path for release.

9 How much will a unit cost?

- PCB ($)
- Electronic components ($)
- Chassis ($)
- Motors ($)
- Wheels and other moving parts ($)
- Sensors ($)
- Memory ($)
- Batteries ($)
- MORE?

10 How long will it take?

- Design and testing of prototype board (hours)
- Mechanical design of mechanisms and structure (hours)
- Field testing (hours)
- Design and testing of final board (hours)
- Sourcing electronic components (hours)
- Sourcing mechanical components (hours)
- Documentation of building process (hours)
- Documentation of usage (hours)
- Documentation of use cases and lessons (hours)

11 How do you plan to make this?

1. Design and build the prototype electronics (DATE)
2. Write the Scratch plugin (DATE)
3. Design and build the initial prototype (DATE)
4. Perform tests and revise (DATE)
5. Implement revision and do field testing with students and teachers, gathering feedback (DATE)

6. Implement second revision (DATE)
7. Design and package kit (DATE)
8. Complete product and kit documentation (DATE)
9. Deploy in sponsor school and begin afterschool program/other school outreach and online distribution (DATE)

12 What documentation needs to be written?

12.1 Product documentation

- How do the electronics work?
- How do the mechanisms work?
- Why were the primary design decisions made?
- Likely troubleshooting and frequently asked questions
- Five introductory class activities or lessons
- Support materials for software usage
- Introductory videos

12.2 Kit documentation

- How do the electronics work?
- How do the mechanisms work?
- Why were the primary design decisions made?
- Likely troubleshooting and frequently asked questions
- Five introductory class activities or lessons
- Support materials for software usage
- Introductory videos
- Walkthrough of build process