MINDSTORMS LOBOTOMY: ROBOTIC WIRELESS COMMUNICATION, COORDINATION, AND CONTROL SYSTEM WITH PARALLEL ACOUSTICAL TRACKING CAPABILITY

Luke Yoder, Mychal Hall, Kyle Madson, Anthony Donaldson, Don Peter

Seattle Pacific University, Seattle, Washington

Abstract

A system for wireless, multi-robotic communication, coordination, and control, based on Lego® MindStorms™ robots and a PC-based Command Center, has been demonstrated at Seattle Pacific University as an example of fruitful undergraduate research and as a powerful extension of MindStorms™ capabilities for use by future students with minimal programming and hardware changes. A custom program resident on a PC allows the human user to initiate command and control decisions and communicate wirelessly to roving robots over its transceiver- equipped COM port, and up to a range of 20 meters. Transceiver units on each robot first of all receive messages, determine which are intended for them, and translate them into the IR signals that are required at their data input ports. Messages are then interpreted and each robot responds according to the specific message. The response may include return messages to the PC, translated back through their IR/RF interface, which can then be used for further coordination and control of the several robots. This allows robots to work together efficiently as a team because only one device is making decisions for them all. A separate ultrasonic tracking system has also been designed that utilizes two microphone 'ears' with accompanying electronics to provide the capability of determining the direction of an ultrasonic beacon. Each robot can then utilize this information to influence how to act – whether to follow, or run away, or make position decisions based on the origin of the sound. An additional technical improvement has been made by replacing the standard AA battery power source with a set of lithium-ion batteries, thus extending operating time to several hours.

Introduction

As a project for the Seattle Pacific University Electrical Engineering Department's Junior Design course, the three-member Mindstorms Lobotomy team created a system for wireless, mulit-robotic communication, coordination, and control, using the Lego® Mindstorms™ RCX™ robots and a PC-based Command Center. This project provides the framework necessary to allow a group of robots to work together to accomplish tasks that are impossible for a single Lego RCX unit.

The RCX Brick is a unit developed by Lego[1] in conjunction with MIT[2] and National Instruments[3] to allow kids to design and program robots built out of Legos. Programs designed in an easy, graphical interface can be downloaded onto the robot using an onboard infrared port, but because the RCX combines the power of a Hitachi H8 microcontroller[4] running analog inputs and outputs with the ease of physical construction inherent to Legos, the platform has caught the

interest of hundreds of hobbyists. While the RCX is a great piece of equipment for easily experimenting with the basics of robotics, and many hobbyists have demonstrated its versatility, the system was never designed to accomplish complex tasks. Any robot built with the system is limited by a short-lived, unreliable power supply, a small instruction memory, and weak computational power. The result is an isolated unit that, while easy to use for simple tasks, is unaware of its location and unable to handle more complex problems. These limitations are what the Mindstorms Lobotomy project aims to solve.

Lego RCX Brick

The project is a multi-pronged attempt to create a communication platform that expands the functionality of the RCX to facilitate future, multi-unit projects. The goal was to create a system that was easy enough to use that freshman engineering students could experiment with its options, but powerful enough that it could be a solid launching point for further projects in robotics. Our work was concentrated in four main areas:

- Developing wireless communication between multiple robots and a base station
- Designing a flexible, PC-based command-center to carry out computationally intensive and decision making tasks
- Designing a separate but integrated system to provide longer-lasting, more reliable power
- Ultrasonic direction detection for tracking

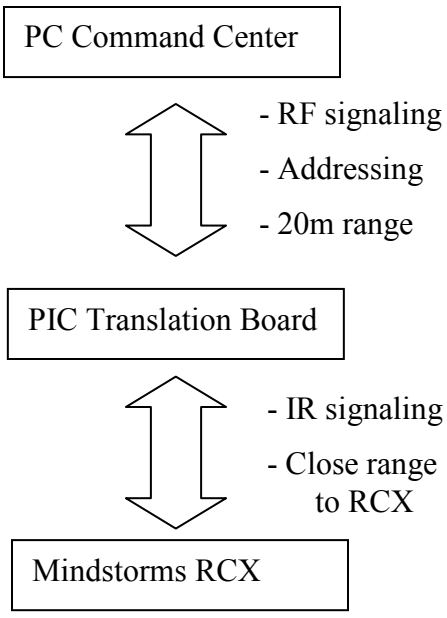
This paper evaluates each of these areas in more detail, looking at our motivations, the solutions we developed, and the difficulties we encountered along the way. It also seeks to analyze some of the things we learned about teamwork, documentation, and larger-scale projects. After all, in any project, the technical design is only one part of a complicated process necessary to produce a final product that is valuable.

Wireless communication between multiple robots

This project began as an effort by a previous SPU student[5] to allow wireless communication between RCX robots. Our project added the command center for increased functionality. The initial project went through several iterations to end up with a device that could parse data from the Mindstorm's IR port, pack it with address information, and send it over radio frequency to other robots. This provides the communication infrastructure necessary to design a system that can take advantage of the communication between different robots.

The solution developed was designed around a Microchip PIC microcontroller[6] that handled the address filtering and IR-RF translation. A received message over RF would be demodulated by an extremely user-friendly Linx[7] wireless transceiver (www.linxtechnologies.com) that transmits the data as a simple TTL signal to the microcontroller. The microcontroller would search for the header, compare the address to verify that it was the intended recipient, then re-encode the data for the RCX protocol and transmit it to

the RCX's IR port, modulating the signal at 47kHz using timers. The RCX would then respond with either a confirmation or the requested data, and the entire process would happen in reverse.



The PIC Translation Board translates between RF and IR signaling

Getting the original communication translation was a formidable task. The protocol of the RCX's IR port is quite complicated and undocumented. It was understood only after long hours in front of an oscilloscope. Once the message was translated, it still had to make it through the noise of random RF signals that were constantly being picked up. To solve this problem, every message sent has a 3-byte header that the receiving PIC searches for to decide it has a real message and not just noise.

We implemented this design on a printed circuit board so that future copies could be made as easily as possible. In the process we found that the schematics left from the previous work on this circuit were not necessarily correct. The previously built prototype worked fine, but was inconsistent with the final copy of the schematic that we received. In our rush to complete the project in 10 weeks, we failed to take the time to properly inspect the design documents for errors, and as a result spent a considerable amount of time simply debugging the board once it was printed and populated. Fortunately, none of the problems required a completely new board; they just needed a few fancy wiring tricks.

When we took up this project we also spent a lot of time refining the communication algorithms to reach a satisfactory level of robustness. Although this system had been designed to handle a large number of robots, it was only tested with two. As a result, communication between more units required some modifications. The biggest issue that we dealt with was that of recovering from lost communication. Every once in a while the RCX would fail to respond back to the microcontroller after it received data. This would cause the microcontroller to freeze, waiting for a return response. We solved this problem by adding some timeouts to the original code.

Once it was completed and debugged, this system worked quite well. With the addition of timeouts and acknowledgements, the solution was tolerably robust, but there is still room for improvement. The group as a whole can always recover from a lost message, but there are still too many lost messages. We did not have enough time to carefully characterize where and why communication did fail, even though we could always recover. Further debugging could lead to a system that failed less often, and could recover more gracefully by attempting to repeat failed tasks before moving on.

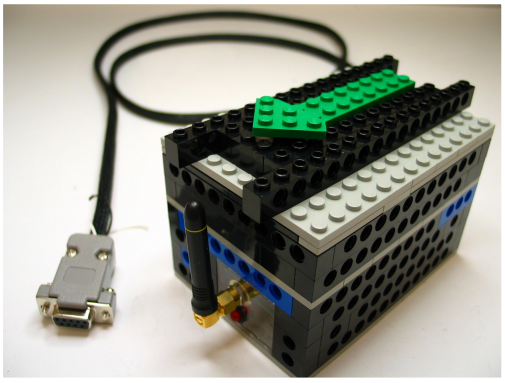
We learned a lot in this section about the importance of good documentation, verification and communication. We just assumed the documentation was correct from the previous project, since it had obviously worked. On top of that, different group members just assumed that

someone else would verify the schematic before laying it out on a PCB. We didn't take the precaution of checking the footprints on the schematic with the component spec sheets, a quick task that seems like common sense now. If we had been more careful in this area, we could have saved hours of debugging and modification.

PC Command Center

Many of the limitations of the RCX cannot be solved simply by creating a conduit for inter-robot communication. The processors in each of the robots are still relatively small. To handle more complex computational and decision-making tasks, we needed a command center with more memory space and the ability to handle floating-point math. The goal was to have the command center that would lay the groundwork for organizational hierarchy among a collection of semi-autonomous robots in a flexible framework that gave the user a single point of access to control every robot in the fleet and was easily expandable for future development. One of our biggest interests for development was in developing some semblance of artificial intelligence on the PC. A system that would allow for this would be open to use in many applications.

Our solution to this was to add a radio frequency transmitter to a regular PC. Since PCs are cheap, ubiquitous, powerful, and extremely flexible, they provided just the platform we needed. We connected one of the Linx wireless transmitters already on the robots to the serial port of a PC, using RS-232 control lines to enable the Linx transmit/receive lines. The rest of the command center was implemented in C++, as a group of classes that could be easily reconfigured by a skilled programmer.



The serial-based transmitter box

We made most of our software architecture decisions around the goal of flexibility. The commands sent to the RCX are stored in a text file that can be modified at any time to add functionality to the program without even looking at the code. A class reads this file each time the program is started and translates all commands sent to and from the robots, screening out noise, handling addressing issues, and recovering from lost connections. This class has high-level methods sending and receiving messages, that can be used by either our current text interface, or any other future class attempting to add artificial intelligence.

The text interface, harnessing the power of the command-interpreting class, allows the user to type in commands such as "forward 3" or "right 30" (degrees) and watch as the chosen robot beeps in submission and responds. The user can look at a list of entered commands, see the values returned by the robots' sensors, and read help strings on each command. This interface is an extremely easy way to harness the power of this wireless communication system. But the functions it calls are available to any interface or controlling algorithm designed to decide each robot's mission. Thus, while the current interface is simple to use, its power remains

accessible with only a few lines of code. The goal is to have a system that can be harnessed by a freshman with very limited programming skills and technical knowledge.

While using a serial interface is not that difficult, the tools available for doing so are different between different operating systems. For this reason, all basic calls to the port are handled through a class that has been written for Windows XP, since that is the most common PC operating system today. A competent programmer could easily write a version of this class that would work in Linux or any other OS without modifying any other parts of the program.

The main issue we encountered once the system was running was that of handling communication failures. How long do you wait for a response from the contacted robot, and what do you do when communication fails altogether? Our program returns after a timeout, with a flag specifying whether or not the reply was received. This leaves the decision about how to respond up to the user or AI, but prevents the program from ever getting hung up by miscommunication.

The overall command center product worked out very well. The RS-232 interface, while slow in comparison to other options, was plenty fast to send and receive short commands to several robots, and was a much easier solution than many other computer interface options. The text file that contains commands provides the flexibility we had hoped for, allowing the addition of new commands in seconds. And, by calling a handful of functions, one can create a program harnessing our provided classes to choreograph an autonomous dance troupe of Lego robots.

This is not to say that the work is done. While the text interface works well and is easy to use, there are many superior options. The most obvious might be to upgrade to a graphical user interface with some image of a remote control, allowing a user to graphically “drive” robots around a room. Another useful option would be to connect this program to a remote computer through TCP/IP, allowing a user anywhere in the world to take the controls of a team of robots that could enter hazardous areas.

Perhaps the most interesting addition to this system would be some semblance of artificial intelligence. Everything is in place to allow a programmer with little knowledge of the circuits and systems involved to experiment with different algorithms for teamwork and communication, all using the compilers and debuggers available for PCs, and avoiding any overhead of dealing with assembly language, limited instruction space, and weak computational power. The possibilities that this platform offers in this exciting area of Computer Science are endless.

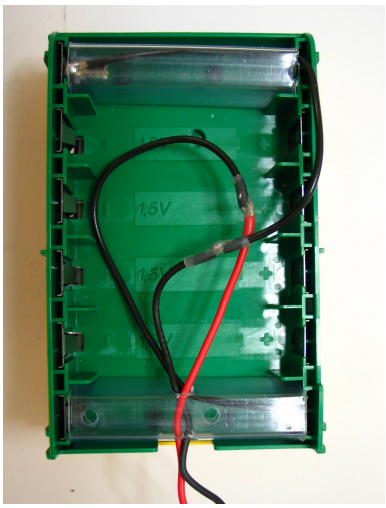
Long-lasting, Regulated Power

When experimenting with the Mindstorms for earlier projects, one of the biggest frustrations that we faced was that of the power supply. The RCX quickly drains six alkaline AA batteries. But now, as digital cameras, PDA’s, and laptops have permeated society, there are much better solutions available than alkaline AA’s or even some of the older rechargeable AA’s.

After struggles against quickly dying batteries that forced a lengthy reload of the RCX firmware and motors speeds that changed with the power left in the batteries, we decided that this project needed a newer, more reliable power supply to truly be useful. On the other hand, with the explicit goal of creating a user-friendly system, we wanted to implement a power supply that was "plug-and-play;" in other words a system that could be swapped in for normal batteries in a manner of seconds.

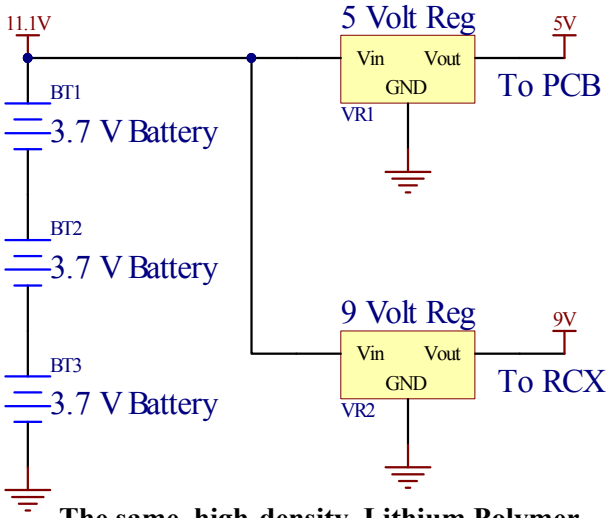
We arrived at a great solution through the donation of several dozen 3.7V Lithium Polymer batteries. We immediately connected up three of these 1600 mAh batteries to each robot, using linear regulators to assure that the 9V to the RCX motors and the 5V to PIC were longer-lasting and well regulated.

This power was fed to the RCX by a plastic insert that connected directly into the standard battery case, with two wires running out the back to main board where the regulators were connected. With all the equipment we designed, retrofitting an old RCX with the new power supply simple meant replacing the standard batteries with this plastic insert. The RCX and the PIC translation board both run off the same set of batteries, reducing the need for extra weight and expense.



The standard, six AA batteries are replaced by this simple insert that runs to the regulated power supply

We had originally hoped to fit the actual stack of three Lithium Polymer batteries inside the RCX cavity designed to hold six AA's, but the batteries were slightly too wide, and only fit with an unacceptable amount of modification to the original plastic case. Leaving the batteries outside allowed us to use an RCX without any quickly reversible modifications to the unit itself.



The same, high-density, Lithium Polymer batteries power both the RCX and the protocol translation board

However, the batteries themselves are not the ideal solution. In order to reap the maximum benefits of rechargeable, high energy density batteries, special chargers are needed. The batteries we had require a charger that switches from supplying constant current to constant voltage at a specific level. Thus, we were stuck with either spending time building a complicated charger or money buying one from somebody else.

On the whole, this approach gave much better results in terms of run time and consistency of the Lego motors, but was not perfect. One of our biggest frustrations was the original need to reload the RCX firmware every time the batteries are replaced. While our solution made that unpleasant occurrence less common, we still had to reload the firmware with

each battery changeover. We discussed possibilities for a live changeover, with one promising option being supercapacitors, but never implemented a solution. The consistency of the robots was increased, but was not perfect either. A lot of inconsistency in the area of robot movement is due to the different levels of friction on the standard Lego parts we used. A robot that thinks it is driving straight will still pull to one side just like a car because of imbalances mechanically. This can only be overcome through some kind of feedback from a system that is aware of it position, and able to correct for deviations in course. That is the goal of the final piece of this project.

Ultrasonic Direction Detection

As mentioned above, perhaps the greatest limitation of the RCX and many simple robots is their lack of awareness of their position. Robots that get off course little by little soon are wandering way off course, completely unaware of their waywardness. Solving this problem is not easy; even the best image processing tools are dealing with very basic issues. Nevertheless, our goal was to create a system that would allow a robot to ”know” its position with respect to several fixed transmitters or conversely to track the other objects in the room.

We decided to implement this tracking system using two ultrasonic “ears” that would measure the time delay between two 40kHz band transducers receiving the same ultrasonic signal. During the interval between the two events, a capacitor would charge from a constant current, the voltage of which was proportional to the time delay and was read by one of the inputs on the RCX. The direction of the origin of that sound pulse could then be estimated using the trigonometry or a look up table for several points where the math is already calculated.

Using this method, a robot could either follow a moving target or calculate its actual position based on signals from several transmitters in known locations. Either approach is an opportunity for the robot to continually receive feedback on its course and make the necessary adjustments. All of sudden a robot that used to constantly get off course can now accomplish tasks that require precise movement to and from different locations.

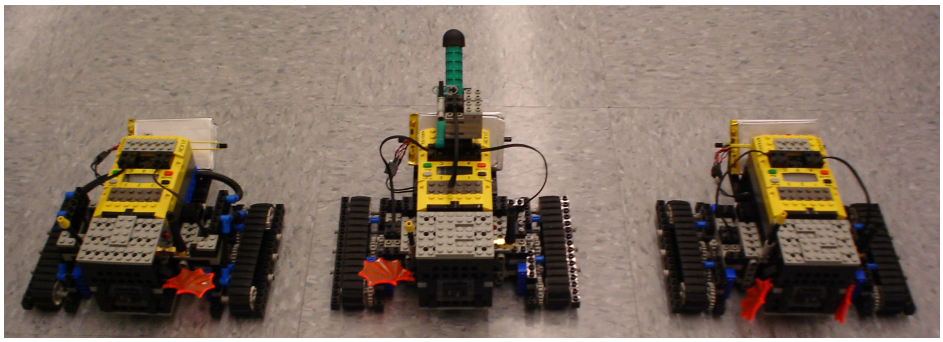
While the entire project was technically sound and each part was demonstrated to work separately, the ultrasonic tracking was never completed as a whole. This was due largely to the ultrasonic transducers chosen. We designed the system with omnidirectional transducers in mind, but ended up with very directional transducers that attenuated very quickly. We briefly experimented with different cones positioned to deflect the sound in all directions but the gains were not enough to save the design.

Even though the transducers were the technical difficulty we encountered, they were not the complete reason that this part of the project failed. The root of the problem lies in the fact that our ambitions were much greater than a 10-week, 5 credit class could handle. This section of the project was of the least priority, and much of the integration work was pushed into the last couple weeks. With another couple weeks to try out some different transducers and experiment more with our differential amplifiers, we could have delivered a much better system. It would

have also helped to not just hope that attenuation and directionality would not be a problem. We assumed that since audible sound can easily travel that far, 40kHz should be able to do the same.

Conclusions

Despite the setbacks and problems we encountered, this project as a whole was very successful. We were able to achieve quite a lot of functionality in a matter of 10 weeks. But a lot of the value of this project came not from what we accomplished, but from what we learned in the process.



Three Mindstorms Lobotomy Robots driving in formation

The work that we accomplished still certainly has value. We successfully built three robots that could operate independently or as a group receiving commands from a user via our text user interface or from C++ script files. The robots ran off our regulated power supplies that lasted longer and were more reliable than the standard AA option. The ultrasonic tracking system never worked fully as a complete system. In the end, we would demo our product first by typing in commands such as "forward 6" or "right 30" and watch the chosen robot follow the command. Then we would run a script file that would cause three robots to drive in formation, with turns that mirrored each other, and ending with all of them firing a small projectile in concert. This is just a small glimpse of the possibilities of this platform with a very small amount of work.

The many different problems and solutions that we encountered often fell under the over-arching idea of communication. Over and over we found that the most important aspect of a successful project was team communication. Some parts we did well, such as working together at the same time on Tuesdays and Thursdays so that each team member was aware of how other parts of the project were progressing. We also did a decent job of defining the interaction points among various parts of the project, allowing integration of sections to go rather smoothly. Our biggest snags also resulted from the breakdown of communication.

One big part of communication is proper documentation. When we started working with documents that were not correct, we were left reverse engineering and modifying circuits and code that we had never seen before. If the time had been taken to properly get those documents in order, it would have saved my team countless hours of frustration. At the same time, we quickly found how easy it is to be rushing at the last minute to fix bugs and let documentation slide in quality. There is a very difficult balance there.

Another area that didn't always work perfectly was inter-member relations. When harried team members become frustrated about different issues, it is important that those are communicated clearly and tactfully to keep the team working together as a unit. Once frustrations start to build, it can be hard to get the technical details communicated. Non-technical problems must be solved quickly to allow the technical work to continue efficiently.

Although we often planned more work than we could achieve, we kept rather close to our original goals. The planning that we worked out early on gave us a good meter to see how far along we were at any given time. It is difficult to estimate the length of different tasks when you are not sure how they will be accomplished, but each attempt gives us a little more of the experience we need to succeed in the future. Unexpected events always arise, either a problems on the project, or outside issues that take time away from the project, but we kept more of less to the schedule that we had planned.

We started out with ambitious goals for a short project, but were really committed to making something interesting and useful happen in that time. Although we spent much more time than average on our project, we found the rewards were proportional. It is always good to know that you set out to accomplish something, and largely succeeded. And, it may just possibly be even more exciting to know that more could be done to make it even better.

References

[1]Lego RCX Brick: Robotics Invention System 2.0 (www.legomindstorms.com)

[2]MIT Media Lab (http://llk.media.mit.edu/projects/cricket/about/index.shtml)

[3]National Instruments (www.ni.com)

[4]Hitachi H8 microcontroller (www.hitachi.us)

[5]Gene Hancock is an Alumnus of SPU (class of 2003)

[6]Microchip PIC microcontroller (www.microchip.com)

[7]Linx wireless transmitter (www.linxtechnologies.com)

LUKE YODER, Senior electrical engineering student, Seattle Pacific University, lyoder@spu.edu

MYCHAL HALL, Senior electrical engineering student, Seattle Pacific University, mhall@spu.edu

KYLE MADSON, Senior electrical engineering student, Seattle Pacific University, mmadson@spu.edu

ANTHONY DONALSON, Director of Engineering Programs and Professor of Electrical Engineering, Seattle Pacific University, ald@spu.edu

DON PETER, Associate Professor of Electrical Engineering, Seattle Pacific University, donp@spu.edu