A Low-Cost, High-Performance Robotics Platform for Education and Research

Max Bajracharya and Edwin Olson

3 Ames St. #325
Cambridge, MA 02142
maxb@mit.edu, eolson@mit.edu

Abstract

The authors have developed a new robotics platform for researchers and educators, providing an improvement in processing power compared to existing low-cost solutions while maintaining a low cost. The platform includes a microcontroller board, GNU-based development tools, and a suite of device drivers and basic utilities. The authors successfully used the platform in a one-month long robotics crash-course at MIT.

Motivation for a New Robotics Platform

While several expensive robot kits are available commercially, many researchers cannot afford to spend thousands of dollars on a basic platform. Educators, attempting to improve the robot to student ratio, are even more constrained. Several robotics platforms have emerged that are affordable for both, including Lego Mindstorms and the Handy Board. However, for researchers and educators hoping to tackle harder robotics problems, the features of these boards may not be sufficient. We have developed a robotics platform that is comparable in cost and ease-of-use to the Handy Board yet provides greater computational power and additional features.

Existing platforms have many good traits that are worth emulating. The Handy Board serves as an example of a successful platform that enjoyed a long lifespan. Some of the traits that we believe are responsible for its popularity are simple operation, simple programming environment based on a commonly understood language (C), straightforward electrical interfaces enabling "hardware hacking," and a general immunity to abuse, making it appropriate for use by inexperienced students.

The primary disadvantage of the Handy Board, Lego Mindstorms, and other similar platforms is their limited computational power. For example, the Handy Board, able to execute only about 500k instructions per second, cannot handle video processing—certainly not motion flow. The 32k memory capacity is also a limiting factor; a single 320x200 video frame cannot be held in memory at once. Robots building maps of their environment can also quickly exhaust the available memory.

Another controller board deserving mention is the Compaq Skiff. Based on the StrongArm SA-110 microprocessor, the Skiff has vast processing power (up to 233MHz) and can address 16 megabytes of DRAM. It also has a large number of I/O ports for interfacing to various types of sensors. The Skiff, however, is an extremely complicated board, consisting of two stacked PCBs with high-density parts on both sides of each PCB. The primary bus interface is PCI, which makes "hardware hacking" very difficult for novices. Lastly, the cost of the Skiff is likely to be a hardship for many.

The Compaq Skiff can be programmed using the GNU toolchain, which provides a performance improvement over the HandyBoard's Interactive C, which is interpreted. It also enables users to use existing and highly-optimized libc and mathematics libraries and the powerful gdb debugging suite.

An additional shortcoming of many existing boards is the lack of explicit support for sophisticated peripherals. It was our goal to provide support for high-current (2A per channel) motor drivers and dedicated inputs for quadrature phase optical encoders, for example. While these capabilities can usually be added to existing boards, it often requires specific knowledge of the internals of the microcontroller and the board's other components.

A New Microcontroller Board

After developing several small robots, and with plans of running an ambitious robotics course, we were motivated to design a new microcontroller board that could serve the role that the Handy Board did, while providing the computational power, memory capacity, and other features that we wanted.

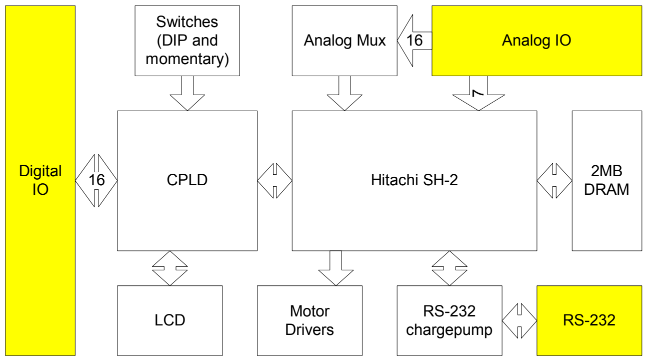

Figure 1. Block diagram of controller board

At the heart of our microcontroller board is the microprocessor: a Hitachi SH-2. The SH-2 is a 32-bit RISC microprocessor with a rich set of on-board peripherals, eliminating the need for many external components. The SH-2 has an integrated DRAM controller, making large memory capacity possible. In addition, the chip supports SRAM-style interfaces, making interfacing with other chips particularly easy. The particular model of SH-2 we selected is available in speed grades up to 30MHz, enabling about 30MIPS of sustained performance. We added 2MB of DRAM, a 16-1 analog multiplexer to increase the number of analog input ports, and an Altera CPLD to implement an interface to an LCD module and provide additional digital I/O ports. We selected a CPLD with a large number of macrocells so that users would be able to add additional functionality; for example, additional counter channels could be implemented. The board should be trivially modifiable to operate without a CPLD, with the SH-2 controlling the LCD directly, in order to reduce the cost of the board.

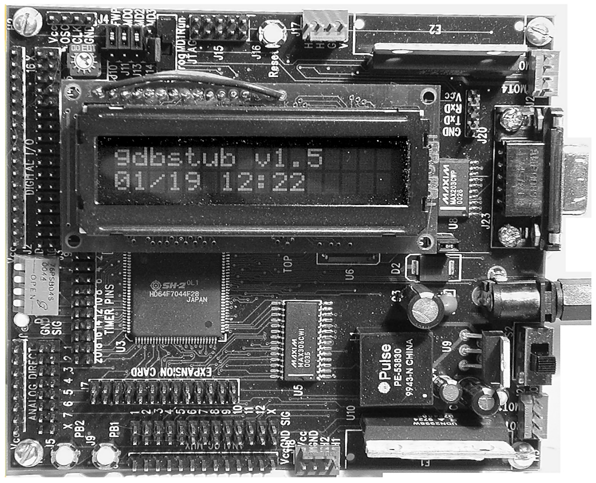

Figure 2. Photograph of controller board

The SH-2 has many on-board timers, which can be used to control motors, servos, and decode quadrature phase signals from optical encoders. In our standard configuration, there are two quadrature phase decoder channels, four DC motor only ports, and four servo or motor ports. The four DC motor only ports are all connected to high current motor drivers providing 2 amps per channel and a current feedback circuit so that current usage can be monitored by software.

The board is physically constructed on a four layer PCB, with two signal planes and planes for power and ground. We had originally hoped that ambitious amateurs would be able to make their own boards, which would imply a two layer PCB and no fine-pitch surface mount components. The reality is that virtually all modern microprocessors are only available in very small pitch packages (making assembly difficult) and their high clock speeds and greater pin counts make a robust two layer implementation very difficult. Using four PCB layers avoids many signal noise problems, while simultaneously making PCB layout an easier task.

Our board is designed with a single power supply in mind. Logic components, sensors, and servos all require a regulated 5V supply, whereas DC motors (which often have a nominal rating of 12V) use a higher, unregulated voltage available directly from the batteries. The regulated 5V source on our most recent boards is generated from the battery voltage. The voltage is heavily filtered to prevent the high-current motors from causing noise problems for the digital components.

We have experimented with both linear regulators switching regulators. Linear regulators suffer from very poor efficiency when the battery voltage is much greater than 5V, resulting in heat problems in the regulator and causing batteries to drain needlessly quickly. With a V_{bat} of 6V, the efficiency of a linear regulator is roughly 83%, but with a V_{bat} of 12V, this drops to about 41%. Switching supplies are bulky, costly, and are less able to maintain a constant voltage when current demands change rapidly, but do yield greater efficiency—about 65% (independent of V_{bat}) in our implementation. However, we observed that voltage drops caused by the slow transient response of the switching regulators caused the board to erroneously reset—a fatal problem. We are still investigating the ideal power source setup, but for our next board revision, we intend to return to linear regulators and provide an option to use separate battery sources for the electronics and motors if V_{bat} is too large for efficient 5V regulation. For example, if a user needs to supply 24V to the motors, they could provide a 6V supply for the electronics and an independent 24V supply for the motors. However, as long as the required motor voltage is close to 5V, a single power source becomes a viable and convenient alternative.

While the board can make use of a wide variety of battery technologies, we have found consumer-grade NiMH batteries to perform very well. 1400mAH are commonly available in a single AA module, and we typically use eight of them to form a voltage source sufficiently high (9.6V) to operate our motors (nominally rated for 12V) and controller board efficiently. These batteries should theoretically last about an hour with the motors running, though in our experience 45 minutes is more realistic. 2200mAH batteries in a size C module are also available, but we have not tried them. A major advantage for users is that NiMH AAs are readily available from almost any electronics store, as are rapid one-hour chargers.

The ideal connector for attaching external devices like motors and sensors would be polarized, inexpensive, high density, easy to assemble, and rugged. Generic 0.100 pitch header is inexpensive and high density but students have trouble assembling reliable connectors. Crimping connectors, easy to assemble and quite rugged, are a tempting alternative, but are very costly and typically take up a lot more room. We have elected to use generic header for analog and digital I/O, in virtually the same way as the Handy Board, but have used crimping connectors for the high current motor connectors. We also use crimping connectors for the quadrature phase connectors, since maintaining proper polarity is essential. We will be using female header exclusively on the next board revision to eliminate the risk of unoccupied header being shorted together by stray pieces of wire or other conductive material.

We have only produced the board in very small runs, so we have not determined how much the board will cost in quantity. Our current estimates put the cost at about $250-$300, putting it well within reach of enthusiasts, and making it reasonable for research groups to build many boards to study robot interactions.

Development Environment

In addition to developing a controller board, we have worked to ensure that the entire GNU toolchain can generate code for it. Gas, gcc (C and C++), and newlib (a libc for embedded systems, including functions like strcmp, printf, and malloc) can all be used, just as they could be on a Unix machine. Code can be downloaded via serial link into either the SH-2's onboard FLASH or the RAM via a gdbstub. The gdbstub also allows programs running on the controller board to be interactively debugged from a PC.

Runtime utilities are also available, including device drivers for a standard HD44780-compatible LCD display, PWM, quadrature phase decoders, and analog/digital I/O. In addition, we have written a lightweight multithreading library.

The complete design of the controller board and all of the software tools have been placed in the public domain. We intend to support the software as best we can, and encourage others to base their robotics projects on our platform. Schematics, software, and other information are available at http://maslab.mit.edu.

Applications in Education

We recently completed teaching a month-long robotics course, known as MASLab or 6.186, at MIT. Like MIT's 6.270 contest, our course is intensive; students meet every day Monday through Saturday. While we envision the course becoming larger in the future, we limited enrollment to three teams (each with about three students) so that we could better deal with any unexpected problems that might arise with our first attempt at teaching a robotics course with a new controller board.

The goal of the contest was to build and program a robot capable of finding and grabbing "targets", then pushing or pulling the targets back to the location where the robot was first turned on. The playing field was a smooth tile floor with a nine-inch high wall around the perimeter. The actual size and shape of the playing was intentionally and dramatically altered from run to run, but was about 100 square feet. Any number of obstacles (also with nine-inch walls) would be placed inside the playing field and random positions. Targets consisted of small boxes with active IR beacons to make them more easily detectable by robots. The walls and obstacles were all constructed out of hardboard, which can easily be detected by IR range finders.

We created a small infrared beacon module that emits an IR signal containing a four-bit ID in every direction, and has a highly directional receiver. The modules have a typical range of 12 feet, substantially more under good lighting conditions. By rotating the IR module around on a servo, it was possible to find the angle to other nearby modules. We also added several modules outside the playing field so that robots could implement an absolute positioning system using triangulation. We statically assigned IDs to each beacon so that when a module was detected, it could be trivially determined whether it was another robot, a target, or a navigation beacon.

The teams were provided with all the basic pieces of a robot. High quality DC motors with integrated gearheads and encoders were obtained from surplus for about $25 each. Each team also received a pair of servos and IR range finders, as well as an IR transceiver module. We provided plexiglass, which can be easily cut and shaped, to form the base of the robot. Additional parts were provided when requested.

The experience of our participants varied dramatically. We had one freshman, three sophomores, one junior, and three seniors participate. Of these, all but one was an EE/CS major. One person had previously taken 6.270, but no one else had any significant experience in developing an autonomous system.

The teams had differing degrees of success. One team managed to successfully find, capture, and drag back a target while avoiding obstacles. They iteratively stopped, scanned their surroundings, and by assigning "points" to various sensor results, used a hill climbing strategy to determine their course. The second team was able to accurately (within a foot or so) triangulate the positions of targets by establishing the baseline and measuring angles to the targets at each end of the baseline. They measured the length of the baseline using odometry and were able to use odometry to drive to the target, but did not have time to incorporate obstacle avoidance into their routines. The third team, the team with the least experience, had significantly more difficulty. They initially invested a large amount of time writing code, underestimating the degree to which real-world non-idealities would impact the behavior of their robot. By the end of the contest, however, they could iteratively stop, scan, and move in the direction of the target, using odometry to return to their starting point in a straight line.

We were pleased to see that all of the teams, even the least experienced ones, were able to build a relatively sophisticated robot—capable of using odometry and IR modules to navigate a very difficult playing field. Two of the teams used velocity feedback control systems (PD) for their drive motors. Students wrote in their course evaluations that they spent the most time gathering together basic behavioral "ingredients" (turning *n* degrees, panning the IR module for targets, driving in a straight line) and had only the last day or two to combine them together into a top-level strategy for retrieving a target. Given an additional week we believe all three robots would have all succeeded in retrieving a target.

It was obvious that the most significant obstacle to the students in succeeding was time. Since the independent activities period during which our course is held will always be a month long, we must make that month more productive. We plan on combating this in several ways in next year's course:

- Hosting a "Build your chassis night". In one case, a team was still doing basic mechanical construction on their robot at the halfway point, and all the teams took at least a week. We believe that students underestimating the total amount of time required to program the robot caused the delay. We plan on designating a day early in the course during which they will build the basic components of the chassis.

- Checkpoints. To keep students from coding hopelessly elaborate control programs without first perfecting basic behavioral ingredients, we plan to set checkpoint dates during which they will demonstrate basic behaviors under contest-like conditions.

- Additional background material. We found that providing background material and demonstrations of what was possible with an autonomous robot not only gave students a more concrete idea of what was possible, but also inspired them to think about the issues involved.

We believe that the environment of our contest, consisting of a smooth surface but with obstacles at unknown locations, is particularly compelling to students; with virtually no modification, the robots could be made to navigate and retrieve targets in real-world environments, such as a dormitory hallway or office.

Future Plans

We are collaborating with the organizers of 6.270 to create the next controller board revision, which will be used in both contests next year. Our work to create a board suitable for both classes will help ensure the applicability of the design to an even broader range of researchers and educators.

Conclusion

We have developed a microcontroller board that offers solid performance at a price that is accessible to students and researchers. By using the GNU toolchain, we leverage existing compilers, libraries, and debugging tools. Multithreading was enabled with a library provided by the authors.

The board was used in a month-long class by MIT undergraduates to build autonomous robots capable of performing predefined tasks in a realistic environment. In addition to providing an educational venue for artificial intelligence algorithms (such as path-finding and map-making), students had the opportunity to learn about embedded and real-time systems programming.

The combination of our controller board and a largely unknown playing field provides a motivating environment for developing control systems for intelligent autonomous vehicles. This is reinforced by an emphasis on software development rather than mechanical engineering.

We believe that our controller board would be a useful asset to those building autonomous robots, as demonstrated by its successful use in our undergraduate course. Future revisions will continue to refine and improve the design.

Acknowledgments. The authors would like to thank our sponsors for making our research and class possible, particularly Hitachi Semiconductor.