Using Mechatronics to Facilitate Active Learning

Stephen Paul Linder¹
Brian Edward Nestrick²
State University of New York at Plattsburgh
Plattsburgh, NY 12901 USA

Abstract
Nurturing a student’s ability to autonomously acquire knowledge is a primary goal of computer science education. Active learning, learning promoted by interacting with one’s environment, as opposed to lectures, is the most effective form of learning in developing a students ability to acquire knowledge. Our approach to promoting active learning in computer science is to incorporate mechatronics, a holistic approach to designing intelligent software for the control of physical systems, into the curriculum. A context-based, collaborative approach to learning, combined with mechatronics-based projects, provide a near optimal method of teaching students how to acquire knowledge about computer science. However, because of perceived cost and complexity, mechatronics has not seen wide acceptance by computer science programs, but has been dominated by mechanical engineering programs. In order to facilitate the use of mechatronics we have developed a set of inexpensive experiments that uses common electromechanical parts and requires only a simple set of tools for assembly. Our approach makes a wide variety of mechatronics-based projects accessible to all level of courses, and for even the smallest computer science departments.

Introduction
Active learning is the ability to learn through interaction with one’s environment. Collaboration is a rich potential component of active learning, since collaboration involves interaction with other people as well as the physical environment. An effective mechanism to facilitate collaboration and active learning for computer science students are mechatronics projects. Mechatronics is a holistic approach to designing physical systems that are controlled by intelligent software (Bradley 1997). As such it integrates the disciplines of computer science with systems engineering, mechanical engineering, electrical engineering and computer engineering. In the past 20 years, with the invention of the microprocessor, the complexity of mechatronics systems has increasingly been transferred from the physical components of the system to the control software. This trend can be seen in the evolution of the automobile, where early systems were almost completely mechanical, then over time incorporated

¹ spl@alum.mit.edu

² BrianNestrick@netscape.net
more electrical components, until today when a typical automobile contains multiple microprocessors controlling everything from the transmission, fuel injection, wheel traction, etc.

Even though the trend of mechatronics is to vest the software component with a greater share of the system functionality, computer science has been co-opted from a leading role in mechatronics education (Kaynak 1996). Mechatronics, however, can easily be incorporated into mainstream computer science education, both at the undergraduate and graduate level. The synergism between mechatronics and the traditional computer science education will help facilitate constructivist learning, by provide experience with real machines, not the simulated machines so often used in teaching computer science. These real machines give students an opportunity to work with a complex environment where the model of the machine is incomplete (Pfeifer 1997). This complexity allows the emergence of behaviors that were not foreseen, making the mechatronic domain even more fascinating for students (Pfeifer 1997; Arkin 1999). When assigning a project, responsibility for integrating the available knowledge into a meaningful solution should be left to the student (Ohlsson 1995), since the emergent behaviors often allows for multiple correct solutions. Students become aware that writing the software is not the difficult part, but rather the difficulty is with discerning which of many possible solutions is best at solving the problem (Dannelly 1999).

Exposure to mechatronics systems is also important because it involves students in projects where computer science and software are only part of the solution. Increasingly software is no longer written solely for a computer platform, but as part of a system where software is embedded in a much larger system. As educators we need to prepare our students to work on these systems. Systems where knowledge of computer science and software development alone is not necessarily sufficient to solve problems, but where the self directed use of active learning to assimilate and integrate domain knowledge is continuously required.

Also, the use of mechatronic-based projects is also in agreement with the thrust of the new Accreditation Board for Engineering and Technology (ABET) Engineering Criteria 2000(EAC-ABET 1998)\(^3\). This new ABET criteria for evaluation of engineering and computer science curriculum requires assessment of student’s capability to take their formal educations apply it holistically to complex problems. Our projects explicit satisfy many of the ABET 2000 outcomes including the demonstration of an ability to

- design a system of components to meet desired needs
- work on multidisciplinary teams, and
- communicate effectively

Even with the great benefits to students of including robots and other mechatronic projects in the computer science curriculum (Nourbaksh 2000), they have not seen a

\(^3\) can be found at http://www.abet.org/criteria.html
great penetration into the curriculum. This is perhaps because electromechanical systems, including robots, have been perceived as expensive, and if not expensive, mechanically unreliable and providing only a limited possibility for software projects.

Mechatronic systems can be as complex as an automobile or the Martian Rover, or as simple as an automated paddle for hitting a Ping-Pong ball. In our program at SUNY Plattsburgh we have incorporated mechatronic projects that use a variety of mechanical platforms. These platforms include simple inexpensive systems for playing one dimensional Ping-Pong, and playing a simplified version of billiards, to more complex projects using wheeled robots to do beacon-finding and solving a maze.

At SUNY Plattsburgh we have developed an inexpensive robotic car, the Handy Car to facilitate a wide range of projects for undergraduates (Linder, Nestrick, Mulders and Lavelle 2001).

This paper continues with some pedagogic background, an overview of mechatronics education, a description of some of our mechatronic projects, and a description of our inexpensive robotic car, the Handy Car, used in several of computer science courses. Detailed information on the construction of these projects can be found at http://www.plattsburgh.edu/csc.

**Pedagogic Background**

**Constructivist learning** assumes that students acquire knowledge by constructing individual models of knowledge (Jonassen 1998). Constructivist learning involves a *manipulation space* which reflects the effects of the student’s actions. A student acts, the manipulation space reflects the effects of their actions, the student tries something different, and the manipulation space shows the changes resulting from these new actions. Constructivist learning begins with the question, “What if….?” However, it is essential that a student not only poses the question, but actually tests their hypothesis. When students participate in this way, they will retain more information, and develop learning behaviors that are critical for continued intellectual growth (Caine and G. 1994). Students in lectures are not usually the ones posing the questions, and thus do not guide the direction the lecture takes. The rigid sequencing of information imposed by the lecturer may not be the optimal sequence for every student.

Faculty can facilitate active learning by providing students with a rich environment that provides a context for the content of the curriculum. This environment should include an experiential mechanism for exploring the curriculum in the context of the world outside of the classroom. However, the lecture-based courses commonly taken by students rely on the long held bias in higher education for the “the sage on the stage”. This tradition of teaching and learning biases education to the thinking mode, and implies that content is all that is important and context is unimportant.

Excessive context can, at times, be a hindrance to learning. In physics, novice students are often told to ignore friction, since it greatly complicates the situation. A perceived lack of real-world resources, such as time or money can greatly inhibit a student’s brainstorming of potential solutions to a problem. However, there is a time to ignore context and there is a time to revel in it. Computer programs are written to be run; they
are made for a purpose. In the real world, a problem is presented, and then a program is written to solve that problem. Often teachers attempt to teach in the reverse order: first facts and other content is presented, and then example problems are presented which make use of that recently acquired content.

Involving students with industrial problems that provide context before the content is even presented as one way to promote active learning. If students are aware of the context of a problem, then they can immediately see the applicability and relevance of the content when it is presented. Intertwining theory and practice helps students understand the way they will learn in an industrial setting and provides motivation to excel at their studies (Solomonides and Botton 1994). When working in industry they will often have to respond to new challenging practical problems by learning new theory, and then applying the theory to solve the problem. Often faculty have the mistaken impression that they must lead with theory, so that students can then later apply theory to the practice of computer science. Research, however, has shown that leading with practice often provides a greater motivation for students to learn the applicable theory. This practice driven approach allows students to develop their own theoretical models autonomously, and does not rely on models that are taught in classrooms (Ohlsson 1995).

Higher education has had an over reliance on traditional didactic methods of teaching. A review completed by Magnesen (Magnesen 1983) gives the following average percentages of material retained in long term memory based on the modality of interaction:

<table>
<thead>
<tr>
<th>Percent Retained Long Term</th>
<th>Modality of Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>Reading</td>
</tr>
<tr>
<td>20%</td>
<td>Hearing</td>
</tr>
<tr>
<td>30%</td>
<td>Seeing</td>
</tr>
<tr>
<td>50%</td>
<td>Seeing and hearing</td>
</tr>
<tr>
<td>70%</td>
<td>Discussing</td>
</tr>
<tr>
<td>90%</td>
<td>Doing and discussing</td>
</tr>
<tr>
<td>95%</td>
<td>Teaching and tutoring</td>
</tr>
</tbody>
</table>

Retention rate of material is particularly important when studying software engineering. The amount of material that the software engineer must become facile with has grown tremendously since the days thirty years ago when computer programs were encoded on punch cards and carried around in boxes. Whereas the syntactical complexity of computer languages has not changed considerably, the ancillary libraries that must be used by the engineer has grown prodigiously. The increase in productivity of software developers has come about in large part by utilizing the large libraries that are now standard with all new programming languages. The retention of all of this additional information more than ever requires students to take an active interest in their own education. Additionally, students that learn mechanisms for retaining large amounts of technical information early in their career will be better prepared to create software in today’s complex environments.
The higher retention rate of material obtained by doing and discussing are recurrent themes in active learning not only because of their positive impact on information retention. Doing and discussing also promotes collaboration. Collaboration among students and faculty has been shown as one way to facilitate active learning (Courtney, Courtney and Nicholson 1994; Pappas, Krothe and Adair 1998). Collaboration forces students to organize their abstract ideas into concrete sentences in order to convey those ideas to other students. This reworking of one’s own ideas creates a more thorough understanding.

The use of mechatronic projects that use robots facilitates this process because the robot in part becomes a participant. The robot grows through the combined efforts of the students. Complexity results in emergent robotic behaviors that excite and invigorate the students(Pfeifer 1997). Students now feel empowered because they can elicit complex behaviors from another physical entity. This perhaps explains the craze for certain toys that elicit seemingly complex response from simple inputs (e.g. the Pokémon Pikachu and the Furby virtual pets and a cell phone that can be made to act as a virtual fishing rod). Motivation becomes internalized because the emergent behaviors of the robot give the robot a personality. Students have created an offspring, and they suffer when the robot suffers. Made implicit is that the intelligence of the robot reflects upon the intelligence and resourcefulness of the creator.

**Mechatronics**

Mechatronics has traditionally been taught by mechanical engineering programs. San Jose State has one of the few concentrations in the field at the undergraduate level (Hsu 1999). Surprisingly this concentration does not include course in computer science. Whereas computer scientists use discrete math to model problems, mechanical engineers abstract their problems to a system of differential equations (Spong and Block 1995). Two simple examples where the differential equation models fail to give simple solutions are the parking of a car and the backing up of a tractor trailer. These problems are difficult because of the nonlinear nature, but can be solved using by neural networks or rule based control (Tanaka 1994).

An alternative approach to teaching mechatronics is to create interdisciplinary teams which include computer scientists. The Multidisciplinary International Virtual Design Studio (MIVDS) program further enhances diversity by forming multidisciplinary teams from students in Turkey, Canada and the United States (Erden, Erkmen, Ronald B. Bucinell et al. 2000). Design4Practice program at Northern Arizona University (NAU) integrates students from Mechanical Engineering, Electrical Engineering and Computer Science to solve mechatronics problems (Erden, Erkmen, Ronald B. Bucinell et al. 2000; Bero, Doerry and Hartman 2001). Students take a sequence of four design courses, one per year, with each course using a progressively more difficult design project. NAU has found that these multidisciplinary teams are better prepared to synthesize mechatronics solutions then a team from only one discipline.

---

4 See http://www.cse.nau.edu/Design/
Mechatronics is inherently an multidisciplinary approach. Ideally team members should be diverse in their academic backgrounds, problem solving heuristics, and personal experiences. Teams should have students from several academic departments so students can be exposed to the different perspectives each profession will bring to bear on the problem. While a diverse pool of technical students can be found at most universities, even smaller colleges can promote diversity by using students from physics, math, etc.

The Massachusetts Institute of Technology (M.I.T.) offers the exemplar of many introductory robotic-based mechatronic courses: “6.270” (pronounced six-two-seventy). 6.270 is held during the one-month long winter session and is open to students of all majors. Teams of two to three students compete in an elaborate robotics competition. Each teams is provided with a kit with a collection of Lego bricks, electronic sensors, motors, gears, and a microcontroller board.

The University of Queensland offers an freshman level course in mechatronics utilizing the game of Robo-Cricket (Wyeth 1997). In Rob-Cricket, pairs of robots compete on a round playing field in a modified game of cricket. Student teams design and construct a robot from a kit which included a small microcontroller board, the Interactive-C programming language, LEGO™ and a collection of sensors and actuators. The course was structured to promote team work, including the use of challenging problem domain which requires a high level of cooperation to construct a competent robot, and by limiting each team to one parts kit.

Krishnan, Das and Yost from the University of Detroit Mercy describes a mechatronic course that culminates with a team project to build a wall following robot (Krishnan, Das and Yost 1999). They found their course improved student retention and better prepared them for their career.

Computer science students at SUNY Plattsburgh also have a mechatronics-based capstone design course. Counter to the intuition of most engineering professors, computer science students from a liberal arts college were able to construct behavior-based robots with a high level of functionality equaling or bettering those built by the electrical and mechanical engineering students. We attribute this in part to our students having been exposed to discrete math, event-driven software, multi-threading, and software engineering in the core computer science curriculum.

**Mechatronic Projects**

The following section will describe a progression of mechatronic projects that we have developed over the previous two years at SUNY Plattsburgh. All our projects use commodity parts and require a simple set of tools to construct — no machine shop is required. The most commonly used parts are hobbyist servo motors ($15), infrared distance sensors ($16), wood, screws and Lego’s.
These projects were designed to have simple real world analogs, with simple requirements. This approach differs from the 6.270 contests which often have elaborate rules for scoring points.

**Ping-Pong**

Ping-Pong can be used as a basis of a simple mechatronic project. A one dimensional form of the game can be constructed using 2x4 studs. Two studs are bolted together with spacers so form a track between the 2x4’s that is slightly wider than a Ping-Pong ball. As shown in Figure 1, a student designed paddle and gear box is mounted at the end of the track. The ball can be tracked either by using simple infrared break sensors mounted to the rails or to use a camera mounted above the track.

Before the project begins, students are given an absolute grading standard based on their efficacy at returning a ball against a human player. After all the grading is complete students compete in a competition against each other for fame and glory.

**Billiards**

A modified form of billiards can be played using a flat table as the playing surface. Golf balls are placed on the table and the objective of the game is to use a cue ball to knock all the golf balls off the table. The simplest version of the game uses a set of Sharp Infrared Rangers mounted on servo motors to detect the balls on the table. After the balls are detected, the shooter is aimed using a servo motor and a cue ball is ejected.

Students at SUNY Plattsburgh have also implemented this game using a video camera as a sensor. A Java-based application uses machine vision to detect the balls and then aims the shooter. If the cue ball missed, visual feedback is used to correct the aim of the shooter.
**Beacon Finding**

Beacon finding is a common project for robotics and intelligent control courses. Our version of the project used a robot developed at SUNY Plattsburgh, the *Handy Car* (Linder, Nestrick, Mulders and Lavelle 2001), which is based on a Radio Controlled (R/C) car kit and a M.I.T. designed Handy Board. By using a robot car, instead of a robot that can pivot in place, the software complexity of the project is greatly increased. Students must now program behaviors that mimic that of human driver when the vehicle is in tight corridors or corners.

**Maze Mapping**

The capstone Group Programming Project course for the previous two years has designed a distributed system for mapping mazes. Mazes are constructed by using 2x4 studs. A desktop computer then constructs a map of the maze from telemetry collected by a robot while the robot traverse the maze. Our initial project used only Sharp Infrared Rangers and a shaft encoder on the axle to measure, respectively, hall widths and distance traveled. A more sophisticated version of the mapping robot uses a video camera to detect road signs and mile markers.

By distributing the computation load between the robot and desktop computers simpler robots can be used, and larger software teams can be more easily supported. Our ultimate goal is to provide a live online version of this system to attract high school students to our program..

Prof. Illah R. Nourbakhsh at the Robotics Institute of Carnegie Mellon University teaches *Introduction to Mobile Robot Programming*, which also uses a variation of the maze mapping project (Nourbakhsh 2000).

**Handy Car**

The Handy Car was develop as a simple low cost mobile platform for mechatronic projects. As shown in Figure 2, the Handy Car is constructed from a R/C car kit. Our approach is similar to the three-wheeled Palm Pilot Robot\(^5\) developed by the Robotics Institute of Carnegie Mellon. Instead of desktop computer, a Palm Pilot controls the robot through a serial interface to a Pontech SV203 controller. The controller is used to drive the three servo motors and reads output from three distance sensors and three shaft rotation encoders. However, we feel that our design is better because students have better cogitative models of our car-based design, and our software development environment is more flexible. With the Handy Car, software can be developed to run either for the PC or the robot.

If funds do not allow a separate robot for each project group, more than one group can share a robot. One PC is connected to the Handy Car and acts as a server for the robot. Students than access the Handy Car using clients running on separate PCs. This approach allows multiple students to share one robot in an introductory computer science course.

\(^5\) See [http://www.cs.cmu.edu/~pprk](http://www.cs.cmu.edu/~pprk) for more information.
The next two sections describe the mechanical design of the robot and the software API.

**Mechanical Design**

The mechanical base of the Handy Car is an extremely stable mechanical environment because its is constructed from a Radio/Controlled (R/C) car kit\(^6\). These R/C cars were designed to withstand the abuse of children. The car uses a servo motor to steer the front wheels and electronically controlled DC motor for propulsion. Pulse width modulation is used to control both motors.

![Diagram of Handy Car](image)

**Figure 3. Schematic of robot with a Plexiglas electronic deck mounted on a Radio/Controlled (R/C) car base.** The electronic deck contains a Handy Board computer board, IR distance sensors and a breadboard for prototyping simple electronic circuits. The electronic deck is mounted on the mechanical base using standoffs. All electronics, except for the electronic speed control, are mounted on this Plexiglas deck. The electronics include a 68HC12-based Handy Board, an associated battery pack, a proto board for constructing simple electronic circuits, and IR distance sensors. Three connections are made down to the mechanical base: two pulse width control signals for controlling steering and propulsion, and a digital input signal from a Hall Effect switch that counts wheel revolutions. The sensors, breadboard and batteries are mounted using Velcro while the Handy Board is mounted using short standoffs.

---

\(^6\) R/C cars come either as unassembled kits or as prebuilt kits which include a radio controller. We used an unassembled kit because it provides more flexibility and is considerably cheaper.
This simple design allows a Handy Car to be assembled by any student or faculty in less than two days. Only limited facility with tools is required. A soldering iron is required for electrically connecting sensors, battery pack, and motors to the Handy Board; a hacksaw to cut the Plexiglas for the electronic deck; an electric drill to make mounting holes in the Plexiglas and the bumpers of the car, and a hot glue gun to mount the Hall Effect sensor and magnets. Finally, Velcro strips are used to attach the battery pack and sensors to the electronics deck.

A complete Handy Car costs approximately $470 with the following cost distribution:

<table>
<thead>
<tr>
<th>Parts</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mechanical Base</strong> – a bare R/C car, servo motor, electronic speed controller</td>
<td>$160</td>
</tr>
<tr>
<td><strong>Electronic Deck</strong> - Plexiglas, screws and standoffs</td>
<td>$15</td>
</tr>
<tr>
<td><strong>Electronic Controller</strong> – an unassembled MIT Handy Board</td>
<td>$220</td>
</tr>
<tr>
<td><strong>Sensors</strong> – IR distance sensors and a Hall Effect-based rotation sensor.</td>
<td>$75</td>
</tr>
<tr>
<td>Total</td>
<td>$470</td>
</tr>
</tbody>
</table>

The price will be reduced by $100 when we replace the Handy Board with a more reasonable priced microcontroller board. Add $80 for an assembled Handy Board.

**Software API**

Our software API supports a real-time software interface between the robot and a PC using the serial port of the Handy Board and the PC. Communications is done using a message-based, handshaking protocol based on the SEMI Equipment Communication Standard (SECS) standard. Currently, the PC side of the serial interface is written in
Java using the Java Communications (COMM) package: javax.comm. The Handy Board interface is written in C. The actual serial connection can be done using a tethered, wireless IRDA, or radio.

Using our API, control messages are be sent to the robot and either solicited or unsolicited data messages are received from the robot. When running the serial interface at 9600 baud, the interface can support 6 data messages per second. Each message contains, a status byte, five distances sensor values, steering angle, and robot speed.

Our Java interface provides a class wrapper around the serial interface, so the complexity of the interface is abstracted. This abstraction allows students in introductory computer science classes to easily interface to the robot. Students can also inherit from this interface and define new commands and data messages. However, the interface code on the Handy Board must be modified to support the new messages by either the instructor or more advanced students.

Supplementing our Handy Car to PC interface is an additional interface that supports communication between the host PC and other client PCs using the Internet. This interface allows several students to share a single robot through the single host computer, facilitating the of the testing of students’ software projects.

**Conclusion**

The facilitation of active learning is important for the effective education of all computer science students. Students who consistently develop mechatronic projects that interact with their environment learn to obtain and retain information better, and they learn to organize their abstract ideas into more useful models of information. Additionally, mechatronic projects promote the learning of important collaborative skills.
6.270 introduced the possibility of using mechatronics even in a lower-level courses accessible to non-engineering students. Our experience over the last two years has supported this, and mechatronics has indeed invigorate our computer science students and has facilitate active learning. Leading with practice has provided a great motivation for students to learn the applicable computer science theory, and students are now more comfortable in learning autonomously. No longer are students intimidated by a stack of manuals.

Students also find that they have much more to talk about at job interviews. Prospective employees are impressed that students have solved challenging "real" problems in a team environment. Also, demonstrations of our mechatronic projects by upper division students to lower division students and high school students has provided inspiration and motivation to all.

**Acknowledgements**

We would like to acknowledge the supports of the students in our capstone courses in developing much of the software required to make the Handy Car run. Brian Nestrick and Symen Mulders for their undergraduate research project designed, constructed and programmed the robots, and developed the informational website. If you would like help designing similar systems you can find all their resumes online.

**Bibliography**


