Player/Stage: A Unifying Paradigm to Improve Robotics Education Delivery

Monica Anderson, Laurence Thaete, and Nathan Wiegand
Department of Computer Science
The University of Alabama
Box 870290
Tuscaloosa, AL 35487-0290
{anderson | lthaete | nwiegand}@cs.ua.edu

***Abstract*— Training students in intelligent robotics requires a large investment both in time and cost. Intelligent controller development requires proficiency in many areas including math, programming, electronics. Students that are not patient with themselves can be frustrated by the large learning curve. This frustration increases when the robot equipment used for assignments is not available (broken or limited in number). Frustration and a sense of failure can cause students to avoid robotics in the future.**

In this article, we propose a new approach to teaching robotics to new students. This approach uses a general purpose, open-source package called Player/Stage. This package provides a hardware abstraction layer to several popular robot platforms. In addition, the Player/Stage provides a simulator for controller testing prior to robot implementation. With Player/Stage as both the simulator and the implementation API, the students are able to apply robot automation concepts in simulation as well as to low and high cost platforms. We believe that Player/Stage, when coupled with robust hardware, is a viable paradigm for classroom instruction that moves the students up the learning curve in small, digestible chunks. A bonus is that experience with Player/Stage can be leveraged into more complex applications and research.

I. Introduction

It is the interdisciplinary nature of robotics that attracts many computer science students. However, the breadth and depth of knowledge required often scares potential students away. Hardware failures frustrate students. Software interfaces are either simplistic and inflexible or too complex for simple learning activities.

Cost factors can also create problems [1]. Schools that choose to add robotics classes to their curriculum make a large investment with a particular vendor for specific hardware and software. Because of the investment, the cost dictates available resources and therefore class size. Student interest is not the primary driver. Many schools opt for less-expensive, less sophisticated solutions that do not grow with the students knowledge and ambitions [4].

When choosing a classroom robotics platform, cost is not the only factor to consider. A successful undergraduate introductory robotics platform should have a low-learning curve for simple activities but should expand to meet the needs of advanced classes and research. To improve scalability, a simulation component should be available that uses the same interface as the robot [2]. Ideally, simulator-based controllers can be reused as the hardware controller with minimal modifications.

In this article, we present an approach to teaching robotics hardware/software that addresses many of these issues. Our approach is three-pronged. First, we introduce the basic concepts involved with hardware kinematics and sensing by making use of a software abstraction layer that eliminates the need to work directly on hardware (Section II). Second, two platforms are selected for student assignments: a smaller number of high-cost, high-fidelity platforms and a higher number of low-cost, lower-fidelity platforms (Section III). Third and finally, the concepts are presented through lecture and a series of exercises designed to strengthen knowledge of navigation and sensing in simulation and on robotic platforms (Section IV). The work in summarized in the conclusion (Section V).

II. Player/Stage

Overall, working with robots involves an interesting and often challenging interaction between software and hardware elements. Often, it is necessary to have a fairly deep understanding of both software and hardware when attempting to successfully develop and design intelligent robotic solutions. In order to manage the steep learning curve of robotics, we make use of the environment collectively known as Player/Stage. Player/Stage [3] is an open source package that provides robot control and simulation.

In particular, Player is a software component suite that forms a general purpose abstraction layer for defining and controlling robot platforms. To facilitate our analysis we further subdivide player into two main components, client libraries and a server layer. The client libraries (available in several languages and extendable to almost any language) allow a student to write software in a familiar paradigm and language. In addition, students will be able to connect to the Player server in order to communicate with robot hardware. The server represents the abstraction layer between a student's code, augmented with library functionalities, and the commands necessary to interface with robot hardware, either real or as simulated by a simulation environment.

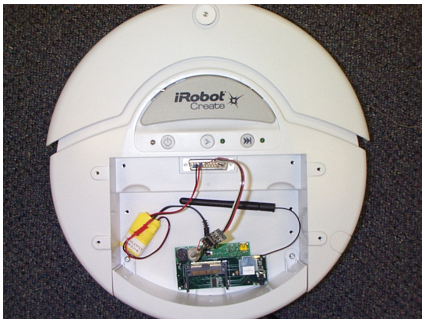

Fig. 1. iRobot iCreate robot

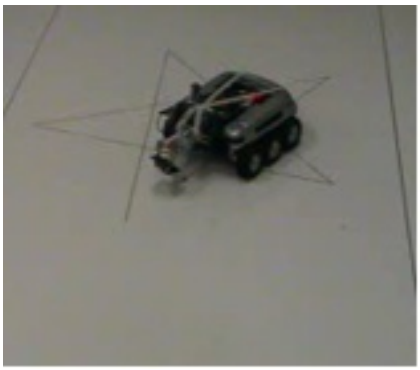

Fig. 2. K-Team Koala robot demonstrating the Logo-enabled PenBot.

Stage is the simulation environment designed to interface with the Player server. Stage functions as a two-dimensional test environment for the simulation of robot hardware in a sufficiently close approximation to a real-world environment. Stage, like Player, is fully customizable. It supports the straightforward definition of new test environments and robot hardware variables. The Stage package is designed to couple with the Player interface in order to allow students to simulate robots and robot behavior without the need for physical hardware. The robot models in Stage are designed so that they replicate important physical attributes such as turning radius, robot size, and shape. Students who develop code for even a simulated robot must be aware of these limitations. This awareness was necessary when moving to the hardware platforms.

Together, Player and Stage provide a deeper understanding of kinematics and the capabilities and limitations of sensors, without the overhead of dealing with actual hardware. Introducing hardware control via a simulation environment has the advantage of focusing students on the algorithms and not any hardware issues. Access to hardware is not an initial factor which in turn reduces lab time at the beginning of the term. Students could work at home at their own pace and often experimented with different implementations. Time for multiple implementations would have been limited if students started in the lab on hardware.

III. Platforms

A. K-Team Koala: Complex Research Platform

The research platform, Koala (see Figure IV), was used as the primary class robot. Although too costly to provide in high numbers, this platform has many sensors mounted and available in the off the shelf offering. The platform includes a proximity sensor ring and ambient light sensors as well as values for battery-level and temperature. Odometry and motion commands are accessible via a serial interface. A PC104-based 500Mhz computer is used for the controller. This computer runs Player/Stage and makes its interfaces available via 802.11b for remote control.

B. iRobot iCreate: Low-cost Platform

The robotics platform, iRobot Create (see Figure III), provides a simple yet versatile robotics platform for beginning to advanced robotics students. This platform was chosen as the base of our robotics platform, however we opted not to use the optional command module, an 8-bit 20 MHz (Atmel ATMega 168 microcontroller). Instead we used the Gumstix platform, a 400MHz Intel PXA255 driven platform along with a 802.11b wifi extension module and serial port extension board. The small size of the Gumstix (4 inches by 2 inches) along with low power consumption (approx. 0.6A) and a preinstalled version of linux provides a powerful robot control platform, that is compatible with Player / Stage robot simulation and control platform.

Inherent in the use of the Player/Stage platform is some cost flexibility. Since a Koala robot is a research quality platform, high-fidelity motors and sensors justify the high cost. However, other platforms, such as the iCreate robot can be used at a much lower cost. A ten-robot class pack is $1000. To run player as a robot interface, a single-board computer (SBC) with 802.11b and Bluetooth were added which raises the per unit cost to approximately $500. The player interface accepts commands from controllers running locally on the SBC or anywhere on the network.

The first step entailed building a version of Player that would run on the Gumstix architecture. This turned out to be the easiest step as a mostly configured buildroot is provided by the Gumstix manufacturer. With only a few changes to the standard buildroot, a working version of Player was integrated into the Gumstix file system image. This image was then flashed to the Gumstix memory using Hyper-terminal on Windows. Alternatively, minicom may be used from a linux environment.

The next step was to provide 5 VDC to the Gumstix board. Unfortunately, the iRobot platform does not provide 5VDC at enough current to power the Gumstix motherboard and associated extension boards. However, there is 12VDC provided at sufficient current, so a simple 5VDC voltage regulator was used to bring the 12VDC down to 5VDC to power the Gumstix rather than using a separate 5VDC battery to supply the Gumstix with power. This has an added benefit of allowing the power switch of the iCreate platform to also

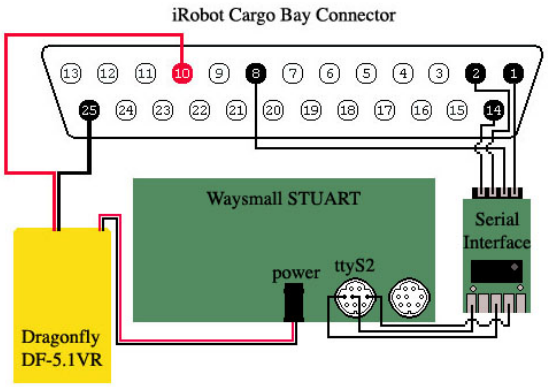

Fig. 3. Wiring information for interfacing the iCreate to the Gumstix SBC

TABLE I
PARTS LIST FOR GUMSTIX-ENABLED ROOMBA

iRobot Create [5]	$129.00
gumstix basix 400xm [6]	$129.00
wifistix [7]	$79.00
console-st (Waysmall STUART)[8]	$20.00
Dragonfly DF-5.1VR voltage regulator [9]	$28.95
Serial Interface [10]	$11.75

control the power of the Gumstix motherboard. With this configuration, we can easily get more than an hour of actual running time from a single battery charge cycle.

The final and most challenging step to complete was setting up communications between the iCreate and the Gumstix platform. The Gumstix provides a serial port that runs at a logic voltage of 3.5VDC and the iCreate exposes a serial interface that operates at 5VDC. We found that we could send messages to the iCreate but could not receive data due to the differences in the operating voltages. Our solution was to use the serial port normally reserved for the terminal. The console terminal port was configured to another port. A serial to ttl voltage stepper was then used to connect the Gumstix to the iCreate.

The final interface wiring harness is simple and requires a minimal amount of soldering, further reducing the learning curve and difficulty of creating the robot platform. All the required parts for the harness are "off the shelf" (see Table I) and a simple schematic of the final wiring harness is provided in Figure 3.

Using the iCreate robot illustrates two of the most important advantages to Player/Stage. First, iCreate robots and controllers can be simulated via Player/Stage even though one is not available from the vendor. A simulator is an important addition to a class as it can reduce frustration in the initial stages of learning by separating learning the interface from hardware issues and inconsistencies. Second, a student that learns on Player/Stage and the iCreate robot can move algorithms and knowledge to other Player/Stage enabled platforms.

IV. CURRICULUM

The INTRODUCTION TO AUTONOMOUS ROBOTICS class consisted of both lectures and workshops. Lectures focused on the modeling aspects of robots and sensors as well as presenting popular obstacle avoidance, mapping and navigation algorithm implementations. Lectures were not sequential but were iterative. Simple obstacle avoidance and navigation, introduced early to facilitate class project design and homework assignments, were expanded later in the class. Three workshop sessions were held early in the semester to familiarize the students with the robot hardware, operating system, and programming tool chain.

The robot-related semester workload consisted of three homework assignments and a class project. Of the three homework assignments, the first two were simulation-based and due within the first 8 weeks. The first assignment was designed to introduce students to the simulation environment, and for some students, Linux. The focus was to simply move the robot around an a priori map. Although this assignment was easy, it assured a "level playing field" before the subsequent more challenging assignments.

The second homework assignment asked students to move the simulated Koala as far as possible within a limited number of time steps from the initial start position. A single controller had to accommodate any start point within the environment. Students that moved the robot at least 10 meters for all test runs using four different start points were considered successful. However, the assignment required that the robot move as far as possible and not stop at 10 meters. Such open-ended assignments are not usually well-received by computer science students. However, students tackled it as a challenge. The assignment created a lot of offline discussion about algorithms and approaches. It also fit very nicely into subsequent lectures about heuristic and map-based navigation strategies.

The second half of the semester focused on hardware implementations. The third homework assignment required students to implement obstacle avoidance (see Figure IV) and wall-following (see Figure IV) on the Koalas. The robot had to stay within 20-40 cm of the wall using sensors that measured about 30 cm. Most students reused code modules for wall-following from the simulations as starting points. Once students tested the range and accuracy of the range sensors, they modified their code to handle the new data.

Although the homework assignments had specific requirements to exercise knowledge in different areas, the class project was open-ended and students were left to self-select groups and design the project. Three projects were created: a room-sized version of the Atari Pong game, navigation using a topological map, and a Logo-enabled penbot.

Using Player/Stage had the effect of enabling more sophisticated projects than most education-based platforms. The projects would not have been possible on a less-capable interface. This has the advantage of creating students that are research ready. In addition, implementing creative solutions is easier when the tools are not limiting.

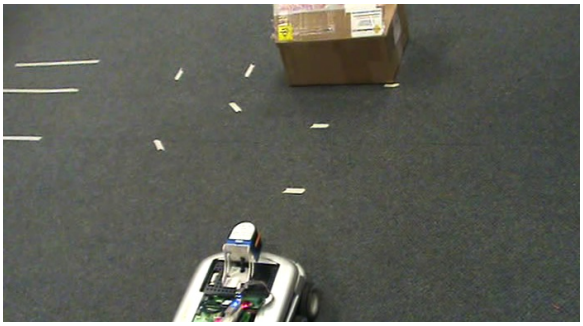

Fig. 4. Koala robot demonstrating obstacle avoidance during homework 3.

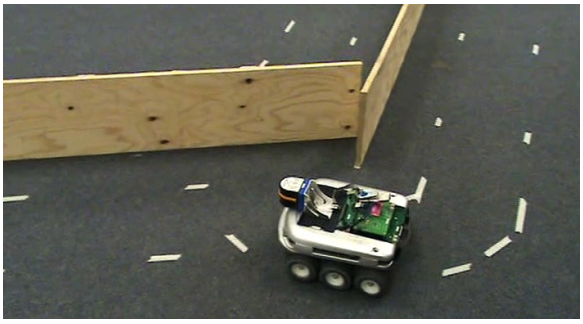

Fig. 5. Koala robot demonstrating wall-following during homework 3.

One major unexpected accomplishment was the application of robot concepts and a player implementation to the iCreate robot platform. This platform was not formally addressed in class but students used the iCreate robot as the puck in the Pong game. More importantly, the students were not afraid to work with two different platforms knowing that the interface was the same.

Using Player/Stage allowed students to focus first on the programming API and then second, on the real world, non-deterministic aspects of controller programming. Learning all these concepts at once can be daunting. Our approach creates early success through simulation-based implementations. Some students need early success to build confidence in an area that is often seen as complex and difficult. Subsequent problems with hardware were seen less as failures of personal knowledge once students understood the programming and modeling aspects.

V. Conclusion

In conclusion, we suggest that teaching college students robotics does not require a simplified interface, but the correct abstraction. The major advantage is the portability of knowledge and code. Students can program new platforms without extensive training allowing the them to implement more advanced projects and solutions. In addition, algorithms developed for one platform can be moved to another, provided the correct sensors are available. Ultimately, the goal of increasing robotic knowledge is best served through paradigms like Player/Stage that reduce the learning curve to digestible chunks without stifling the ability to implement creative, advanced solutions.

Fig. 6. Koala and iCreate engage in automated game of Pong where two Koalas act as paddles and the iCreate is the puck.

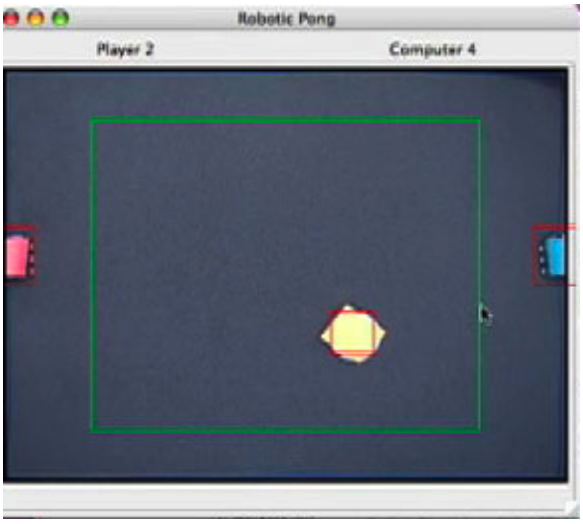

Fig. 7. Screen shot: Playing area with robot locations built from an overhead camera. Color blobs are used to track robot positions.

The students in the class had advanced programming skill but no prior robotics experience. This class was considered a success because students had favorable attitudes toward robotics at the end of the class. More importantly, students reported better attitudes toward hardware than before the attending the class. This assessment was significant for the two female students. There was a 18% drop rate (2 students) but the drops occurred before the hardware portion of the class. Anecdotally, they were related to too many classes in one instance and falling behind on class presentations in the other instance. Drops due to frustrated students would have been a signal that the approach was not appropriate. However, one student that dropped is enrolled in the next class offering.

Future classes will incorporate the structure of simulation and hardware implementations using Player/Stage. More advanced classes will explore higher level behaviors such as exploration, mapping and localization.

REFERENCES

[1] J. Challinger. "Efficient use of robots in the undergraduate curriculum," *SIGCSE 05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, 436440, New York, NY, USA, 2005. ACM Press.

[2] B. Fagin, U. Academy, and L. Merkle. "Measuring the Effectiveness of Robots in Teaching Computer Science," *SIGCSE Bull.*, 35(1), 307311, 2003.

[3] B. Gerkey, R. Vaughan, K. Stoy, A. Howard, G. Sukhatme, and M. Mataric "Most valuable player: A robot device server for distributed control," *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 12261231, 2001.

[4] U. Wolz. "Teaching design and project management with lego rcx robots," *SIGCSE Bull.*, 33(1):9599, 2001.

[5] http://store.irobot.com/product/index.jsp?productId=2586252&cp=2600059.2591511 &parentPage=family

[6] http://gumstix.com/store/catalog/product_info.php?products_id=154

[7] http://gumstix.com/store/catalog/product_info.php?cPath=31&products_id=171

[8] http://gumstix.com/store/catalog/product_info.php?products_id=98

[9] http://www.rctoys.com/rc-products/DF-51VR.html

[10] http://www.acroname.com/robotics/parts/S13-SERIAL-INT-CONN.html