

Bonaventure Undergraduate Robotics Laboratory
Technical Report 4
September, 2000

**The Khepera Robot and the kRobot Class: A Platform for Introducing Robotics in
the Undergraduate Curriculum**

Robert Harlan, David Levine, Shelly McClarigan

The Khepera Robot and the kRobot Class: A Platform for Introducing Robotics in the Undergraduate Curriculumⁱ

Robert M. Harlan

David B. Levine

Shelly McClarigan

Computer Science Department

St. Bonaventure University

{rharlan,dlevine}@cs.sbu.edu

Abstract

We discuss a class interface for the Khepera robot that makes the robot an excellent platform for undergraduate robotics courses and robot-based lab exercises in other courses. The interface hides low-level robot-computer communication and permits the building of derived classes that encapsulate related base behaviors relevant for higher-order tasks.

1 Introduction

Historically, the study of robotics has been limited to graduate level classes at large universities, owing to the high cost of buying and maintaining robots. Recently, the advent of smaller, less expensive robots has made it feasible for smaller institutions to integrate robotics into the undergraduate computer science curriculum.

We have discussed the reasons for adding robotics as well as the approach we use to introduce it elsewhere [2]. In brief, we introduce robotics to give students experience with real-time systems programming and to motivate student participation in large, continuing projects that present the challenges of large-scale software engineering. Our robotics course introduces the development of low-level behavior control algorithms and examines two distinct approaches to the design of intelligent robots: the traditional approach, which places emphasis on a centralized planner that manages low-level

behaviors to carry out tasks; and the more recent, behavior-based approach, which holds that intelligent behavior is a property emergent from low-level behaviors tied to environmental stimuli.

This paper focuses on the software we have developed to support the hardware platform selected to support both our robotics course and robot-based labs we plan to introduce in other courses. The hardware platform is the Khepera miniature robot developed in by the K-Team in Lausanne, Switzerland [1,5]. Its size, engineering and cost (approximately \$1,800 per unit) make it an ideal robot for an undergraduate lab. Each robotics workstation requires only seven to eight square feet for experimenting with behavior control algorithms, and students do not need to touch the robot's sensors or effectors.

The software interface provided with the Khepera, however, makes the platform difficult to use in a robotics class and impossible to use for one or two lab experiences in other computer science courses. The development of even simple behavior control algorithms involves not only interpreting robot-environment interaction but low-level, robot/computer communication from which data must be extracted. The low-level communication interferes with the students focus on the appropriate response to environmental stimuli, and errors in the former are sufficient to derail a project.

We have developed a class hierarchy that enables students to concentrate on a specific layer, e.g., controlling a robot, while abstracting from other, typically lower-level layers. The kRobot class hides communication and permits focus on robot-environment interaction. The Serial class is responsible for robot-computer communication, and it can be examined independent of developing behavior control algorithms. Higher order classes can encapsulate basic behaviors and permit focus on high-level AI-type issues such as planning.

The most important class is the kRobot class, which has the goal of enabling students to write basic behavior control algorithms for the Khepera robot. The class hides low-level

details involved with robot/computer communication and allows developers to focus on robot/environment interaction in controlling behavior.

The class has been used to design, implement and test control algorithms for basic behaviors such as light-following, wall-following and obstacle avoidance. It has also been used to build derived classes that encapsulate a set of basic behaviors. BARD [3] is a centralized planning class that is capable of managing the delivery of mail in a small office complex. Students working with BARD can focus on higher-level functions such as planning the best way to deliver mail and managing the execution of the plan, as the relevant basic behaviors are built into the class. Flaky, a robot navigator, consists of behaviors relevant for moving around in an environment and avoiding obstacles. Students are able to focus on higher-level issues such as planning an optimum route in a known environment.

This paper discusses the Khepera hardware and the software we have built for it using two sample behaviors – one low-level and one high-level – and our plans for using the kRobot class to introduce robotics-based labs in other courses in the curriculum. The classes can be obtained from our robotics lab website, <http://web.sbu.edu/cs/roblab.html>.

2. Overview of the Khepera Robot

The Khepera robot is a miniature mobile robot with similar capabilities to larger sized robots that are most often used in research, educational facilities, and other real-world environments. Its small size (55 mm diameter, 30 mm height), light weight (approx. 70 grams), and compact shape are ideal for micro-world experimentation with control algorithms.

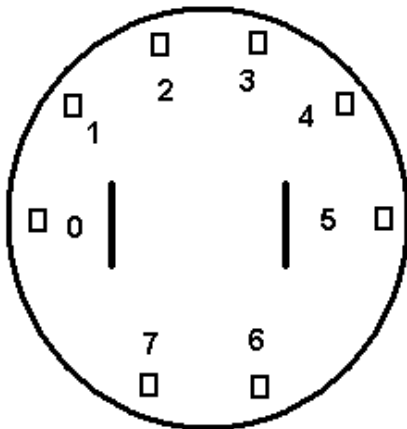


Figure 1: Schematic of the Khepera

The Khepera has sufficient sensors and actuators to ensure that it can be programmed to complete a wide variety of tasks. Its eight infrared sensors can sense both ambient

light levels and proximity to nearby objects. It also has two DC motors that are capable of independent variable-speed motion, allowing the robot to move forwards, backwards, and to complete a variety of turns at different speeds.

The Khepera has several extension modules that can be plugged in to the top of the robot. These include a gripper arm, a two-dimensional vision system and a digital camera.

The Khepera has an onboard Motorola 68331 processor, 256KB RAM, and 256KB ROM. It also has a rechargeable NiCd Battery that allows it up to 30 minutes of high-activity autonomy.

The Khepera can be run autonomously (the control program is downloaded to the robot's processor) or tethered to a host computer. We prefer the tethered mode: the control program running on the host computer is able to query the sensors of the robot and control the robot's effectors. It is much easier to develop and debug control programs in this mode, and the size of the controlling program is not restricted by the size of the robot's processor and memory.

In the tethered mode, the Khepera uses the SerCom protocol for communication with the host computer. The protocol is based on ASCII commands and responses. Commands are in the form of a capital letter, followed by any parameters needed, ending with a carriage return and line feed (represented hereafter as `\r\n`). Responses from the robot are the lowercase character corresponding to the uppercase letter command, then any data it needs to transmit (such as readings from the sensors), and a `\r\n`. For example, the command to read from the proximity sensors is:

```
N\r\n
```

To this command the Khepera might respond with the following string:

```
n,0,59,1023,1023,78,0,0,0\r\n
```

The response is returned as a C-style string and must be parsed to determine the values of each of the proximity sensors.

3. Development of the kRobot Class

The SerCom interface provided with the Khepera was unacceptable for our course for it is neither the product of, nor does it lend itself to object-oriented design, the design methodology utilized throughout our curriculum. Further, the need to extract data from strings really prevented the students from focusing on the robot-environment interaction required to develop behavior control algorithms.

We wanted a clean, intuitive interface for the Khepera robot that hides the low-level communication between the robot and the controlling program and permits students to focus on developing basic behavior control algorithms.

The kRobot class provides this interface. The robot object stores its state information, and the client program can access this information and issue commands to the robot. Methods include opening and closing the serial port connection to the host computer, moving the robot and monitoring the light and proximity sensors. For example, `r.ReturnProxSensor(4)` returns the integer value of proximity sensor 4. `r.ReadSpeed()`, obtains the rate of rotation of the robot's motors and `r.SetSpeed(100,100)` sets the speed of the left and right motors. There are also functions for actions such as moving forward, turning around a left corner, stopping, etc.

The kRobot constructor creates an instance of the Serial class. The Serial class provides a programmer-friendly interface to the serial port through which the robot communicates with the controlling program. The details of opening and closing a serial connection are hidden here and are accessed through the kRobot class's `OpenConnection()` and `CloseConnection()` functions. The `Talk()` function of the serial class manages communication between the robot and the remote computer.

There is no comparison between the simplicity and elegance the kRobot class provides for managing a Khepera robot and the original SerCom interface. To obtain the proximity value of infrared sensor four using the serial port directly, one must:

- Open a connection to robot through the serial port (*not an easy task, takes about 30 lines of code*)
- Send "N" to the robot (*sending a command is also difficult, each individual character must be sent one at a time*)
- Receive a response such as "n,0,59,1023,1023,78,0,0,0" (*commands received from the serial port are also received one character at a time, they must be manually reassembled*)
- Parse the response to get the individual sensor readings from the string (*each string of numbers must be pulled from the main string*)
- Convert the string "78" to the integer 78, and return it

Completing the same task using the kRobot class is accomplished by sending three messages to a kRobot object:

```
r.OpenConnection(); // open a connection to
                    // the Khepera
r.ReadProxSensors(); // reads the proximity
                    // sensors
val = r.ReturnProxSensors(4); // value of
                    // 4th proximity sensor
```

4. Using the kRobot Class as a Student

The kRobot class is designed to permit students to develop low-level behavior control algorithms for the Khepera robot. It was designed and built over the course of a semester as part of an undergraduate honors project following our initial experimentation with the robot.

We demonstrated the ease of use of the class by having students use it to implement basic behaviors such as following walls, turning and recognizing offices that are needed to build a robotic office delivery system (BARD).

We built a physical model of an office complex, a rectangular office suite with offices numbered incrementally from 101 (Figure 2). There was one exterior corridor, and the exterior walls of the complex measured 3' by 2.5'. The base was made of 1/4" pegboard, and walls were constructed of 1.5" high poster board. The complex was set on a table next to a Sun workstation and permitted ample room for the robot to maneuver.

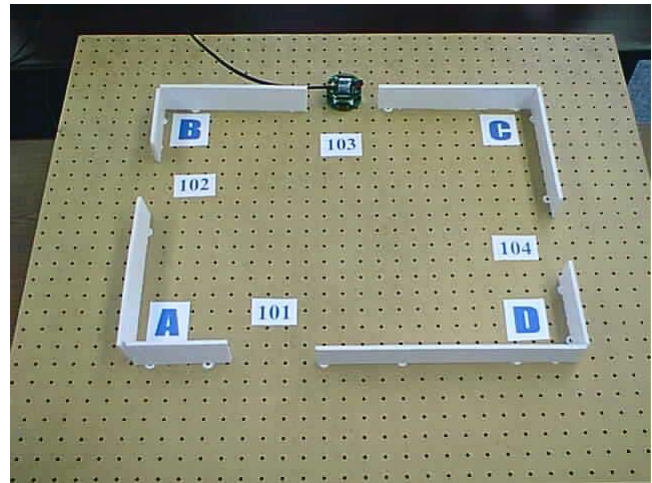


Figure 2: BARD's World

The first behavior developed was wall following: the robot would follow walls until it reached a corner or an office. The ability to follow a wall, however, required first that the robot find a wall to follow. The robot's proximity sensors were used to determine if it could see a wall. If it could then it could begin wall following, otherwise, the robot needed to move a little closer to the wall, and check the sensors again. The `MoveForward()` and `TurnRight()` functions were used alternatively accomplish this.

The `FindWall()` algorithm, illustrated in Figure 3, is:

```
while a wall is not in view
  TurnRight(a little bit)
  MoveForward( slightly )
  CheckSensors()
```

The specific values for “a little bit” and “slightly” are determined by testing.

Once a wall has been found, it is necessary to follow it without either running into it or drifting too far away from it, i.e. beyond sensor sensitivity. Since the robot’s motors are imperfect, and since the wheels may slip on a surface the wall following algorithm will also involve low-level behaviors and can be developed in a manner analogous to the wall finding algorithm.

Once appropriate low-level or base behaviors are defined, they can either be packaged as a derived class or used directly to implement higher-level behaviors. In the case of the delivery system BARD, students developed a central planner that generates a plan for delivering a set of messages in the most efficient manner and then carries out the execution of the plan using the base behaviors. Because the base behaviors had already been developed, students were able to concentrate on the more abstract problems such as how the robot should represent the office complex internally, how it can determine its position in the complex, and how to determine the most efficient way to deliver messages.

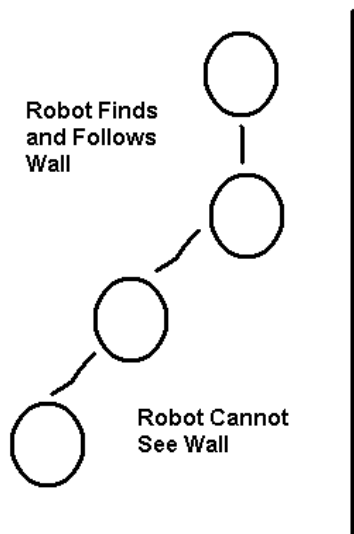


Figure 3: Finding a wall

5. Future Developments of the kRobot Class

The kRobot class is robust enough to permit even difficult behavior control algorithms to be implemented and it can become the basis for derived classes that uses a collection of base behaviors to accomplish higher-order tasks. The software design permits students to focus on whatever level of abstraction the instructor deems appropriate. Over the next two years we plan to use the software to introduce robotics topics in other courses in the curriculum.

We are building a robot simulator that will utilize the same interface developed for the Khepera. The CMU Graphics

Library [6] is being used for the simulator, which will permit its use on a variety of platforms. The simulator can be used, e.g., in an introductory lab that introduces iteration and selection control to enable a robot to follow light.

As indicated above, the robot navigator, Flaky, is capable of planning routes in a known environment. It can be used to evaluate blind and informed searches in either an algorithms or an artificial intelligence course.

Building the equivalent of the serial class, which handles the serial port for robot-program communication, would make an excellent lab for a computer organization lab although we have not yet developed the instructional materials for this.

All of these labs have the positive benefit that they reduce the cognitive overhead that students face as they encounter new material. The uniform interface reduces learning time in subsequent courses and thereby permits more time to be focused on the problem at hand.

6. Conclusions

The kRobot interface we have developed for the Khepera robot makes it an excellent platform for introducing robotics throughout the undergraduate curriculum. The interface simplifies the design and testing of control algorithms by hiding the low-level communication between the controlling program and the robot. Further, it can be used to build derived classes that encapsulate related base behaviors so that higher-level tasks can be implemented

The class hierarchy makes it possible for students to concentrate at an appropriate level of abstraction, making the Khepera platform useful not only for a robotics course but for one or two session lab experiences in other courses.

References

- [1] Baroni, P., Fogli, D., Guida, G., and Mussi, S. Active Mental Entities: A New Approach to Building Intelligent Autonomous Agents. *SIGART Bulletin*, 9, 1, (Summer 1998), pp. 10 - 19.
- [2] Harlan, R. Adding Robotics to the Undergraduate Computer Science Curriculum. *Proceedings of the Fifteenth Annual Eastern Small College Computing Conference*. St. Bonaventure University (1999), pp 112-119.
- [3] Harlan, R. Information on the BARD (automated delivery system), Flaky (robot navigation) and other projects developed using the kRobot interface is available at <http://web.sbu.edu/cs/roblab.html>. April, 2000.

- [4] Kumar, D., and Meeden, L. A Robot Laboratory for Teaching Artificial Intelligence. *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education* (1998).
- [5] Mondada, F., Franzi, E., and Ienne, P. Mobile robot miniaturization: A tool for investigation in control algorithms. *Experimental Robotics III, Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan. (Springer Verlag: 1994) pp. 501-513.
- [6] Stehlik, Mark. CMU Graphics Package. Available at <http://www.cs.cmu.edu/~mjs/apcs.html>. November, 2000.

ⁱ Work on this paper as well as the equipment used was funded in part by National Science Foundation CCLI-AI grant 9980999