Saturday Science Robots for Junior High Students

Marjorie Skubic

Dept. of Computer Engineering and Computer Science
University of Missouri-Columbia
Columbia, MO 65211
skubicm@ missouri.edu

Abstract

Saturday Science is an outreach program designed to introduce 8th and 9th grade students to career opportunities in the physical sciences. The students participate in a hands-on programming session in which they learn to control simple lego robots with differential motors and bump sensors. The robots are controlled by a Tiny Tiger microcontoller which is programmed in BASIC. Low-level functions are provided to give the students an intuitive, high-level programming interface. The program has been effective in encouraging math and science and in some cases has even increased confidence in science for adolescent girls.

Introduction

Saturday Science is an outreach program designed to introduce junior high students (grades 8 and 9) to career opportunities in the physical sciences. On three Saturday mornings, we invite 30 students into our lab and teach them how to program robots. The students are grouped into teams, given a handout and verbal instructions. Because of the limited time with the students, they do not build the robots. Instead, the program focuses on introducing programming concepts and teaching students how to program the robots, while immediately seeing the effects of their programming efforts.

The Saturday Science sessions are part of a broader program at the University of Missouri, entitled "Promoting Young Women in the Physical Sciences", which seeks to motivate young women to consider math and science careers. Although Saturday Science is open to both boys and girls, there is a strong interest in encouraging the girls to participate, and efforts are made to achieve at least a 50% female representation. Students are recruited by the science teachers at the three junior high schools in the Columbia Public Schools. Female students, selected by their science teachers, are targeted for recruitment through letters mailed to their parents. Approximately 60 females per school receive letters; brochures and information are available in school for other females and the male students. Participation by the students is completely voluntary. Although most of them have used computers for web searching, word processing, and games, this is the first experience with programming for almost all of the students.

The educational program includes a combination of discussion and programming; however, the main emphasis is the hands-on programming. There are 10 workstations, each with a robot, so the 30 students are grouped into teams of 3 each. As the students progress through the programming exercises, the instructor interjects occasional discussion on how the robot mechanics or the program itself relates to math and science.

Robot Design

The 10 robots used in the program are built using legos. The two designs are shown below in Figures 1 and 2. "Freddy" is built with treads. "Willie" has two active wheels and one passive wheel. Both are controlled with two motors and a differential drive strategy, which is explained to the students. Three bump switches are included as binary touch sensors; two are located in the front with a connecting center bar, and one is in the rear with an attached bar to increase the sensing area.

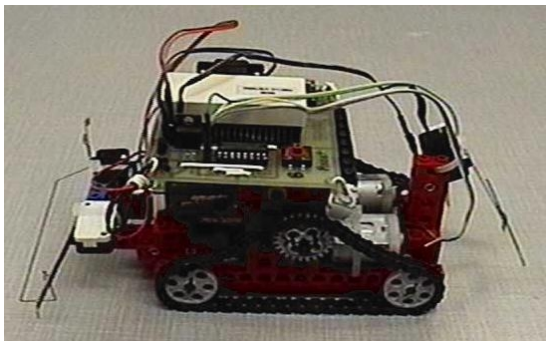


Figure 1. Freddy

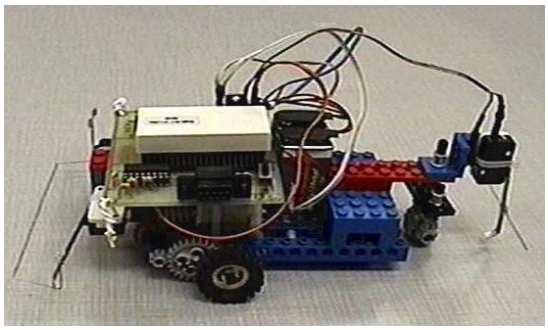


Figure 2. Willie

The brain of each robot is a Tiny Tiger micro-controller (Wilke 2001), programmed in BASIC. The Tiny Tiger chip is a compact module, which includes 2 serial channels, 4 channels for A/D input, and 2 channels for pulse-width-modulation. Different configurations are available. The model used for the Saturday Science robots is the TNN-R/1, which has 32K RAM and 128K Flash ROM and is available commercially for about US$60.

Programming is accomplished through the Tiny Tiger Development System, available commercially for a PC computer. Programs are edited on the PC and downloaded to the Tiny Tiger microcontroller. An MU undergraduate student designed a controller board for the Tiny Tiger chip; the board includes a serial connector to interface with the PC. The printed circuit controller board was manufactured in-house by the MU Electronics Shop. For the Saturday Science program, the PC-based development environment is an advantage, because the students are familiar with PCs and the windows environment.

The intent of the programming exercises was to provide an intuitive interface to the novice programmers. Low-level subroutines (which we call the robot functions) were written prior to the students' visit and provided a form of high-level interface. An effort was made to connect the function names and parameters to the physical concept. For example, GO_FORWARD() turns both motors in an equal velocity and propels the robot forward indefinitely. There is also a TIMED_FORWARD(time) which executes the forward movement for a specified time in milliseconds. The complete list of functions provided to the students, as well as program examples, is shown in the Student Handout in Appendix A.

Sensor input was accomplished with one function READ_SENSORS() which set the appropriate sensor flags for LEFT_BUMPER, RIGHT_BUMPER, and BACK_BUMPER. In this way, programs could be written in an intuitive style using typical programming control structures, as shown in the example below:

```
CALL READ_SENSORS()
IF LEFT_BUMPER = 1 THEN
      CALL TIMED_BACKWARD(500)
      CALL TIMED_RIGHT_TURN(500)
      CALL GO_FORWARD()
ENDIF
```

The student handout and the granularity of the robot functions were prototyped with 2 test students, both 8th graders, prior to the first session. The experience of the test students has generally agreed with the large group response. Students seemed to easily grasp the meaning of the robot functions and control structures such as IF THEN and FOR loops, as long as the commands were executed in a sequential fashion. The subroutine concept seemed to confuse the students in part because the code had to be edited at the end of the main program.

Program Content

The Saturday Science session is held in one of the general purpose PC labs at the University. Ten workstations are set up, each with a robot; the development environment is running and a simple program is loaded. The students come into the lab and sit in groups. The instructor introduces the robot and describes the robot control strategy and the sensors. Using the handout as a guide, the students start by downloading the program which has been provided.

Although there is a serious goal, we try to keep the atmosphere light and fun. After all, the students have given up a Saturday morning's sleep to participate in an educational program. The students can progress through the exercises at their own pace. In addition to the main instructor, there are 4 student assistants that can offer help if the students have questions or get lost or have problems with their robots. Occasionally, the instructor interrupts to introduce new concepts or to offer suggestions. The students who have participated thus far seem to be intensely engaged in their programming projects, and there is lots of noise and activity as the students try out their robot programs. One group is shown in Figure 3.

Figure 3. Students at work.

In addition to the exercises on the handout, the students were offered other robot activities. In Spring, 2000, students were encouraged to write a program which used all of the sensors to escape from obstacles. A large corral area was provided, and the students could put their robot in the corral and watch how it interacted with moving obstacles. Although this was fun for a short time, especially when the robots collided or got stuck together, it did not provide a lengthy, engaging exercise. In Spring, 2001, we have traded the corral for a robot maze, shown in Figure 4. Now the students must devise a strategy for navigating through the maze, which seems to be providing greater interest. Also, the maze provides a more purposeful task that is interesting and has more than one possible strategy.

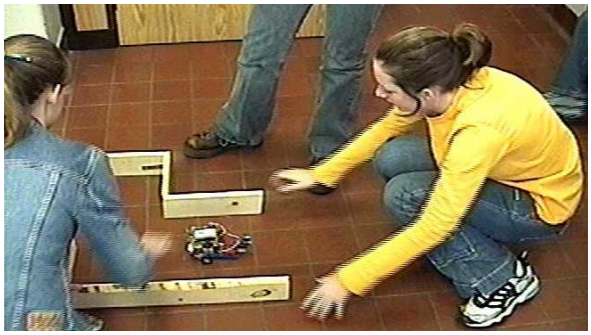


Figure 4. Running the Maze

At the end of the programming session, the students have a wind-down period in a neighboring auditorium. To relate their new experiences to math and science, a few short videos and demos are shown and a discussion is led on the many disciplines involved in creating successful robot technologies. The student assistants (2 MU undergraduates and 2 grad students) take turns addressing the group, describing how they got interested in math and science and what type of projects they are currently working on. The female undergraduate student provided a particularly positive role model to the female students.

The MU students have been instrumental in the success of the program, especially providing an exploration of the career possibilities. Each student shares his or her own personal story, which often facilitates a lively discussion of interesting questions. For example, one of the student's questions prompted a discussion of how one could automate and coordinate various household appliances. Another student asked whether robots could repair themselves, which started a discussion on robots building new robots.

Discussion

At the conclusion of the session, the students filled out evaluation forms. On a scale of 1 (*It was boring*) to 7 (*It was great*), the robot programming session rated an average of 6.0 in Spring, 2000 with a standard deviation of 1.0. The students apparently liked the session but it could have been better. However, the statistics do not tell the whole story. The science teachers have reported that the 9th grade girls who participated as 8th graders have a noticeably higher level of confidence in science compared to the general student body. This is difficult to document but an important result nonetheless and perhaps not surprising. After all, when you have successfully programmed robots, you have accomplished a pretty awesome feat. The students may also have a better appreciation for how math and science are used, and thus may be more motivated to learn those subjects.

From the instructor's perspective, we did notice a change in the repeat students. In the one session held thus far this year, there were 6 returning girls from last year, which were formed into two groups. The returning girls all found their spots quickly and immediately started programming. They had arrived with a purpose. Having already seen the exercises in the handout, they tackled the maze problem much faster than the other students and with greater success. Interestingly, they did not seem to be bored but rather were even more engaged and more forceful than the first-time students.

After noting the positives, let us also relate the negatives and the potential for improvement. The time period allotted for the program was only 3 hours. Slightly more than 2 hours were spent programming the robots with the remaining time spent in the final discussion and demo period. This is long enough for a single session, especially considering there is no break. However, a continuous program with multiple sessions would provide an opportunity for greater depth and for tackling more complex problems. The students could take a break, think about the problem, and return with new ideas and new enthusiasm. However, for those students who are not able to participate in other programs or are not willing to commit to lengthy programs, a single Saturday morning session may provide some benefits. We note that the Robotics lab is one of three visits in the Saturday Science program (the other two are to local science-related industries), so students get to sample three different science–related careers through the program.

Within the Saturday morning format, there are other improvements that could be made. First, a group of two students would be better than three. With two students, one can type the program and one can offer suggestions, but the third is often not actively engaged in the process. Second, the students were frustrated if their robot did not operate perfectly as expected, e.g., if their robot did not go straight forward with the GO_FORWARD() function. The students thought it was their fault or their program was incorrect rather than a simple lack of wheel encoders to ensure that the robots really did go forward. Certainly, the robots could be made more robust. In some cases, we had to make quick repairs on the spot.

Overall, the program has been a success. It would be interesting to do a quantitative evaluation on the effectiveness in learning programming skills as this has not been done. In any case, we believe that the program has a positive effect on the students' attitudes toward technology, especially computers. Perhaps a similar program could be used effectively for other age groups.

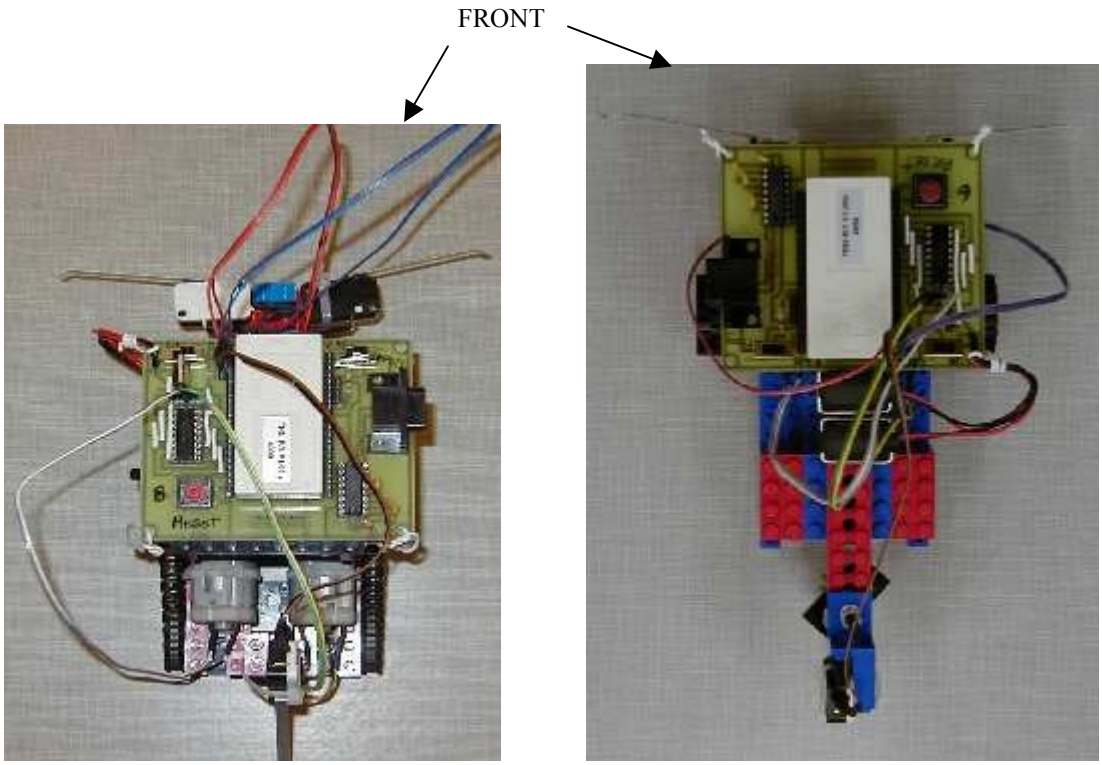
Acknowledgements

Saturday Science is sponsored by the NSF under grant NSF-HRD-96-19140, Meera Chandrasekhar, PI. The program could not have been run without the help of the MU students, in particular George Chronis, Byron Dill, Matt Aubuchon, and Kristi Hummel.

Appendix A. Student Handout

Welcome to Saturday Science at MU's Robot Programming Lab

Today we will be exploring mobile robots and programming them to move around our world. To begin, we have 2 types of robots today, as shown below:

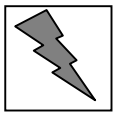


Freddy has treads **Willie has three wheels**

Notice that both robots have 2 bumpers in the front and 1 bumper in the rear. They each have 2 motors, and we can control them by writing a program that drives the motors, telling them when to spin, how fast to spin, and in what direction to spin.

Each robot is connected to a computer by a cable. We program the robot by typing our program on the computer and then compiling it and downloading it to the micro-controller (a very small computer) located on the robot. Let's try a simple program that we have already written for you:

Directions to Download and Run the Robot Program:

1. Connect the computer cable to the robot
2. Set the right switch to **Prog** (for Program)
3. Set the left switch to **ON**
4. Press the **Reset** button
5. From the Tiny Tiger Window on the computer, press the Lightning Bolt icon 
   (The program will compile and download to the robot – you will see this in the window)
6. When it has finished the download, set the right switch to **Run**
7. Press the **Reset** button
8. Disconnect the cable
9. Put the robot on the floor to test the program (give the robot some space)
   To start the program, press the vertical (rear) bumper and the robot will start moving!
   To stop the robot, press the **Reset** button

Let's take a look at our sample program:

```
''''''''''''''''''MAIN LOOP''''''''''''''''''''''''''''''''
CALL TIMED_FORWARD(1000)
CALL TIMED_RIGHT_TURN(1500)
CALL TIMED_FORWARD(1000)
''''''''''''''''''''''''''''END MAIN LOOP''''''''''''
END
```

What does the program do?

Tasks for You

1. Write a program to make the robot move in a square pattern.

2. Write a program to make the robot move in a triangle pattern.

3. Write a program to make the robot do 3 squares. (See program example #1.)

4. (ADVANCED) Write a subroutine to make the robot do a square. (See program example #4.)
   Use your subroutine to make the robot do 3 squares.

Using Sensors

5. Look at program example #2. What do you think the program will do?
   Type the program into the computer and test it with the robot.

6. Type in program example #3 and test it. What does the robot do?

7. Expand program example #3 to include the right bumper. Make the robot backup and turn left if the right bumper is hit.

8. Expand program #3 again to include the back bumper. Make the robot stop and go forward if the back bumper is hit.

9. Write your own program that uses the bump sensors.

Robot Functions

```
GO_FORWARD()                              TIMED_FORWARD(time)
GO_BACK()                                 TIMED_BACKWARD(time)
STOP()

RIGHT_TURN()                              TIMED_RIGHT_TURN(time)
LEFT_TURN()                               TIMED_LEFT_TURN(time)

LEFT_MOTOR(direction, speed)              READ_SENSORS()
RIGHT_MOTOR(direction, speed)             TIMED_REVERSE(time)
```

Sensor Flags

```
LEFT_BUMPER
RIGHT_BUMPER
BACK_BUMPER
```

Program Examples

Example #1

```
FOR COUNT = 1 to 3
    CALL TIMED_FORWARD(1000)
    CALL TIMED_LEFT_TURN(1000)
NEXT
```

Example #2

```
STOP_FLAG = 0
CALL GO_FORWARD()
WHILE STOP_FLAG <> 1
  CALL READ_SENSORS()
  IF LEFT_BUMPER = 1  THEN
     CALL STOP()
  ENDIF
ENDWHILE
```

Example #3

```
STOP_FLAG = 0
CALL GO_FORWARD()
WHILE STOP_FLAG <> 1
  CALL READ_SENSORS()
  IF LEFT_BUMPER = 1  THEN
     CALL TIMED_BACKWARD(1000)
     CALL TIMED_RIGHT_TURN(1000)
     CALL GO_FORWARD()
  ENDIF
ENDWHILE
```

Example #4

```
SUB MY_FORWARD()
    CALL RIGHT_MOTOR(FORWARD,255)
    CALL LEFT_MOTOR(FORWARD,255)
      END
```

References

Wilke (2001). Kg Systems web site.
http://www.industrialcontroller.com/wilke/

Skubic (2000). Saturday Science web page.
http://www.cecs.missouri.edu/~skubic/kids/ss/