A Robot-Based Computer Engineering Module for Manhattan College's Intro to Engineering Course

Robert Mauro
Electrical and Computer Engineering
Manhattan College
Riverdale, New York 10471

Introduction

During the past two years the School of Engineering at Manhattan College introduced a new format for its Freshman Introduction to Engineering course. Instead of centering around a single semester-long engineering design project, the course was modified to include a series of eight Engineering Modules of approximately 5 hours each. The modules introduce the student to each of the College's 6 engineering disciplines along with the specialized topics of Engineering History, and Engineering Ethics.

In an effort to help the Freshman Engineering students better understand the field of Computer Engineering, this year the Computer Engineering Department developed a new module for its portion of this Freshman course which involves student programming of small robots that were designed and built at the College. In 5 hours the students are taught a reduced portion of the BS2 Basic Stamp instruction set[1] and learn how to use the Stamp to create a series of computer programs that range from simply blinking an LED to getting their robot to navigate through a series of mazes. Because the sections of the course are grouped according to the students selected engineering major, each section also gets to spend an additional 10 hours working on a project in their selected field. The Computer Engineering section project consisted of a Robot Sumo Wrestling competition.

The Design of the Robot

Before getting into the details of the student's programming experience, it might first be a good idea to take a look at some of the robot's features. The base of the robot is made from a 4.5" x 6.5" piece of epoxy-glass perforated prototyping board, and the Basic Stamp Board of Education computer[2] is mounted onto this base using four 3/4" nylon standoffs (Figure 1a). The wiring of the Stamp printed circuit board was reworked so that the robot could be powered either from a wall wart, or from batteries mounted on the robot. In particular, a 9V battery is used to power the computer and four AA batteries for the robot drive motors. Although inexpensive carbon-zinc batteries were used, they still lasted for nearly the entire semester in spite of extensive student use.

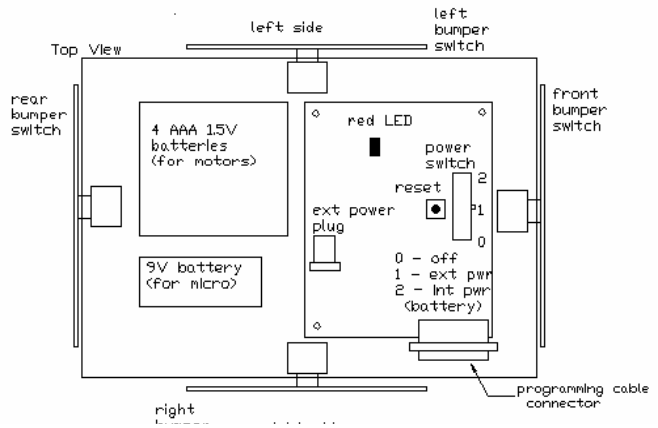

Figure 1a. Top View of the Robot

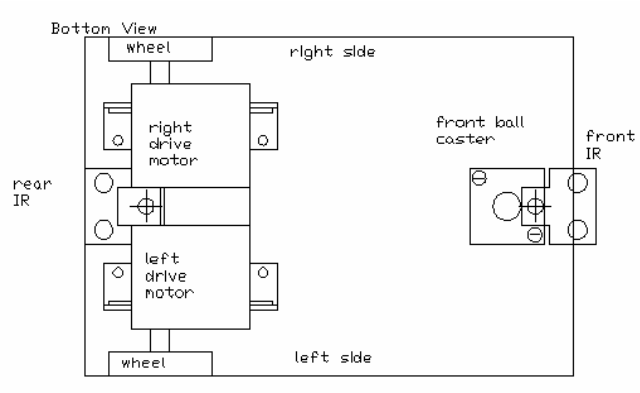

Figure 1b. Bottom View of the Robot

The robot has two sensor systems. The first is a series of four bumper switches located around the perimeter of the robot. For economical reasons these were constructed in-house using switches removed from an old computer keyboard. These switches were then hot-glued onto L-style metal mounting brackets, and the same bracket types were also used to mount and hinge the 1.5" x 3.5" phenolic boards and the attached wire bumpers that served to actuate the switches (Figure 2). The robot also employs two

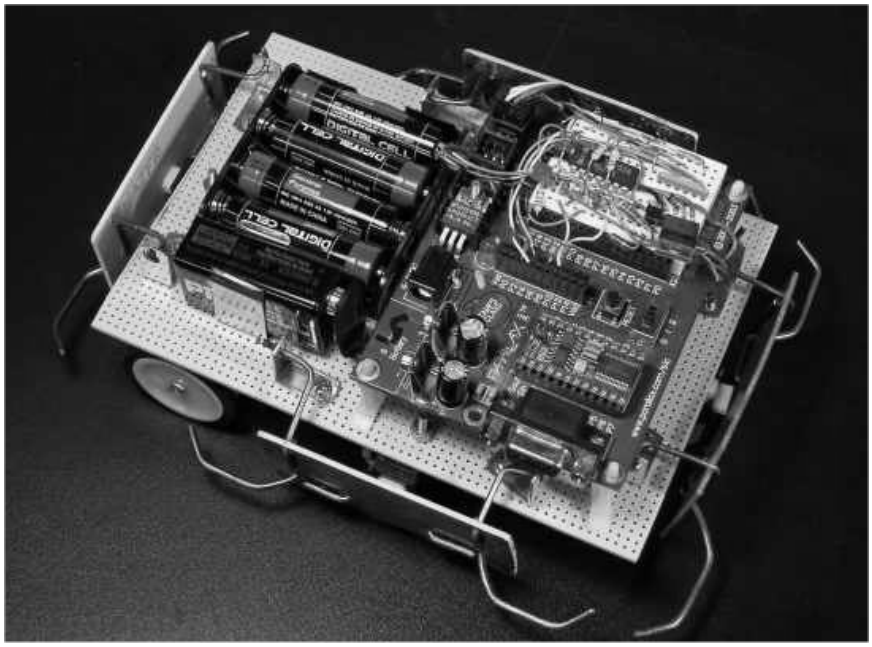

Figure 2. A Photo of the Robot Illustrating the Bumper Switches.

infra-red sensors located on the under-carriage of the robot in the front and the back. The sensors are oriented to detect signal reflections from the floor. Each IR system consists of an Optek OP133 infra-red LED, an OP802 photo-transistor, and an LM-324 Schmitt-trigger. The overall BS2 hardware interface schematic is given in Figure 3. Table I indicates the connection specifics between the Basic Stamp and the robot hardware.

BS2 I/O Pin	Function	Function	BS2 I/O Pin
2	Front Switch	Front IR Sensor	9
5	Back Switch	Back IR Sensor	11
3	Left Switch	Left Motor	13
4	Right Switch	Right Motor	12

Table I. Basic Stamp Interfacing to the Robot Hardware

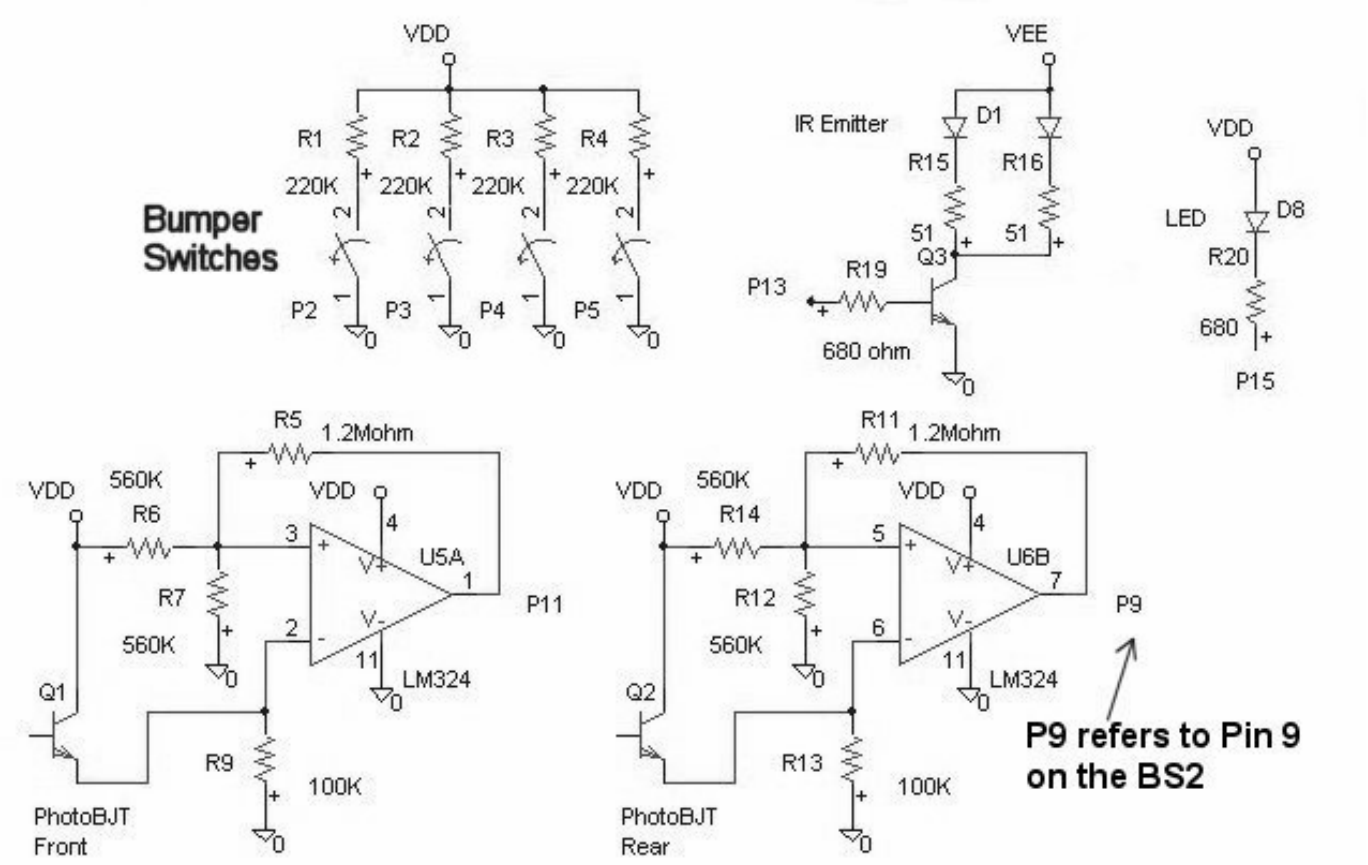

Figure 3. Electronics for the Interfacing of the BS2 to the Robot Sensor Hardware.

The robot drive and steering system consists of a steel ball caster in the front and two independently controlled motors in the rear (Figure 1b). The rear wheels used for the robot are inexpensive VCR idler wheels costing about $0.30 each. The drive motors are Futaba RC servo systems modified to allow for continuous rotation[3]. These units were chosen because they have excellent torque characteristics and directly interface to the BS2 micro-computer. They also allow for direct pulse-width-modulation control of the motor direction and speed (Figure 4).

T = 1000 us = 1 ms	Full speed clockwise
T = 1500 us = 1.5 ms	Zero speed
T = 2000 us = 2 ms	Full speed counter-clockwise

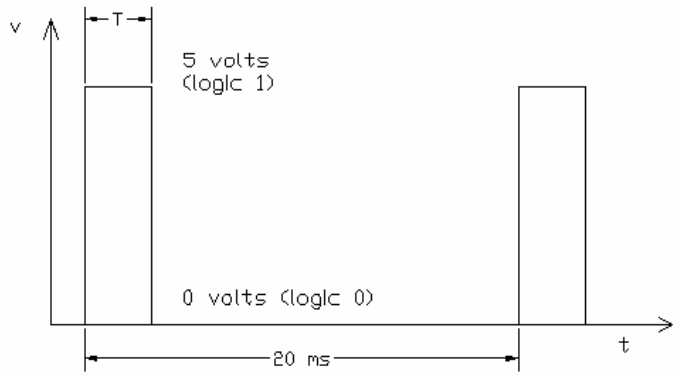

Figure 4. Type of Signal Needed to Drive the Modified Futaba Motors.

The Design of the Robot Mazes

The robot Mazes were constructed from 2" plywood and painted black in case later use was made of the robot's infra-red sensors for navigating through the maze. The plan, if this method were to be employed, would be to place white IR reflective strips on the floor, perhaps with one strip indicating the need for a right turn, and two strips the need for a left. However, in this, the first semester of experience with these robots, only timing loops and/or bumper sensing were used for navigation through the mazes. In constructing the maze the center partition was made removable so that it was a simple matter to convert between maze #1 and maze #2 (Figure 5).

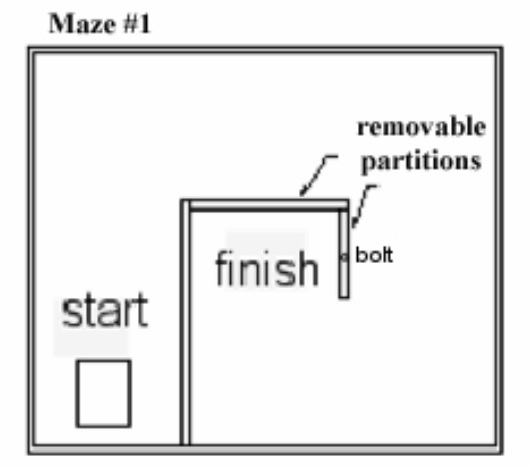

Figure 5a. Maze #1 - All Right Turns.

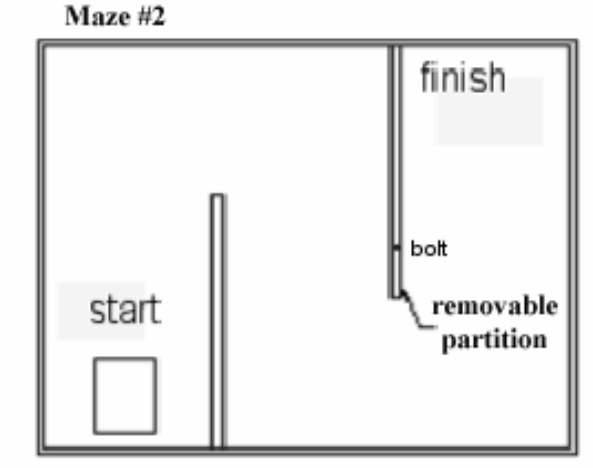

Figure 5b. Maze #2 - Two Right Turns and Two Left Turns.

Student Introduction to the Computer Engineering Module

Because only 5 hours is allocated to this module it is necessary to carefully select the portions of the BS2 instruction set to be taught[4,5]. A list of the specific instructions introduced is given in Table II. The method for teaching these instructions as well as the programming methods and hardware interfacing techniques employed, focus on the introduction of these concepts through a series of carefully selected examples (Table III). From the beginning, extensive use is made of flowcharts to illustrate the programming ideas being discussed. An example of this flowcharting approach is illustrated in Figure 6.

Instruction	**Description**
DEBUG ***OutputData***	Output Format a) Text - Debug ***"text to output"*** b) Variable - Debug ***{DEC, HEX, or BIN} variable {, ...}***
LOW ***Pin***	makes pin an output LOW
HIGH ***Pin***	makes pin an output HIGH
TOGGLE ***Pin***	if output pin is HIGH make it LOW and vice versa
PAUSE ***Period***	wait for indicated period (in milliseconds)
PULSOUT ***Pin, Period Count***	at 2 us/count for the BS2
GOTO ***Address***	Program branches to the label ***Address***
END	stops computer program execution and places the BS2 in the low-power standby mode
GOSUB ***Address***	Program goes to subroutine at the label ***Address***, and comes back after a RETURN to location below GOSUB
RETURN	use at end of subroutine to return to the main program
IF ***Condition*** THEN ***Address***	if ***condition*** is true then branch to label

IF ***condition*** THEN
 statement1(s)
 { ELSE
 statement2(s) }
 ENDIF
	if ***condition*** is true then execute ***statement1(s)***
	 if ***condition*** is false then execute ***statement2(s)***

FOR ***Counter*** = ***StartValue*** TO ***EndValue*** {STEP ***StepValue***} ***statement(s)*** NEXT	set variable ***Counter*** to the value ***StartValue*** initially, execute **statements**, and then increment ***Counter*** by ***StepValue*** (1 if ***StepValue*** not used), repeat this loop again and again until ***Counter*** exceeds ***EndValue***

Table II. Selected BS2 Instruction Set Command Reference.

Generally, the students took about 2-3 hours to complete the programming tasks given in Table III, after which they are ready to get the robot to accomplish the main programming task of getting it to navigate through Maze #1 (Figure 5a) and then through Maze #2 (Figure 5b) using only the bumper switch sensors. For extra credit, students were also encouraged to get their robot to accomplish the following additional tasks:

i. Automatically stop when it has completed navigating through Maze #1 and/or Maze #2.
ii. Be able to navigate through both Maze #1 and Maze # 2 running the same program.
and iii. Return to the starting point and stop after finishing the maze navigation.

Example	Description	Instruction Introduced
1	turn the LED on that is connected to pin 15	LOW, GOTO, END
2	blink the LED ON for 1 second and then OFF for 1 second continuously	HIGH, PAUSE
3	blink the LED ON for 1 second and then OFF for 2 seconds continuously	
4	turn the LED ON when the front switch is hit on the robot.	Reading of input pins, use of ***Inxx***
5	Turn the LED ON when the front switch is hit and OFF when the rear switch is hit.	IF-THEN
6	Output the following messages to the Debug Window when the following switches are hit: front `C>` `A`Front`@`, back `C>` `A`back`@`, left `B>` `A`Left`@`, and right `B>` `A`Right`@`	Debug, If-Then
7	get the right robot motor to run full-speed forward	PULSE
8	get both robot motors to go forward	Multiple Pulse in loop
9	get the robot to run forward for 3 seconds and then stop	Instruction timing and use of IF-THEN loop
10	Get the robot to go forward until the front switch is pressed, and then have it go in reverse	
11	Get the robot to turn continuously in the clockwise direction and record the time for the robot to make 10 clockwise revolutions. Estimate the time needed for the robot to make a 90 degree right-hand turn.	GOSUB, develop RightTurn and LeftTurn subroutines

Table III - Program Development Sequence Prior to Navigating the Mazes.

In preparing the students for the maze navigation portion of the module, the professor ***strongly suggested*** that they write three subroutines prior to beginning this work - a ***LeftTurn*** subroutine, a ***RightTurn*** subroutine, and lastly a subroutine called ***Forward*** that returned to the calling program after the front collision switch was hit. Student teams using this approach found it easy to perform the required navigation of mazes #1 and #2. In addition, nearly all teams working on this module opted to complete one or more of the extra-credit tasks. Furthermore, it was even commonplace for many of the students to stay long after the class was over to get the robot to accomplish the assigned tasks. This was truly an unusual teaching experience. The instructor also found that this particular type of programming problem's strong emphasis on using subroutines vividly demonstrated their importance in writing successful computer programs.

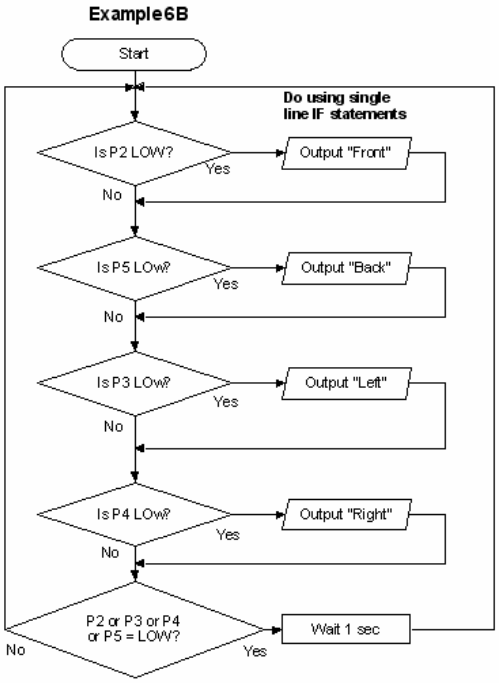

Figure 6. Flowchart for Programming Example 6 Given in Table III.

The Computer Engineering Student's Additional Project Work

Students enrolled in the Computer Engineering program got to spend an additional 10 hours working on a project in the Computer Engineering field during the course of the semester. This year the project they worked on made use of the same robot described earlier. The project was called Robot Sumo Wrestling[6,7,8] and employed the 4' diameter competition ring shown in Figure 7.

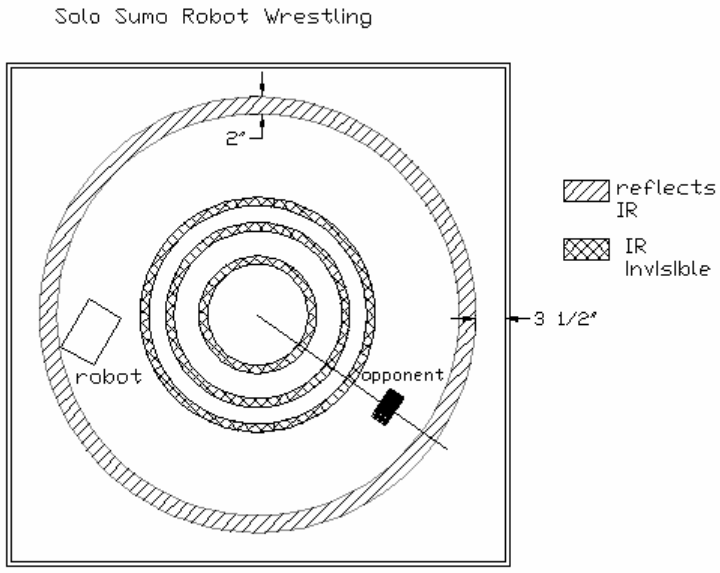

Figure 7. The Robot Sumo Wrestling Ring

Using their previously developed programming skills, the student groups needed to complete three tasks during this portion of the course. Task #1 was a geometric problem which required that the robot park itself in the center ring after having been placed anywhere inside of the large outer circle.

Successful completion of this task involved the use of the IR sensors on the front and the rear of the robot to detect when the robot had reached the outer white circle boundary.

Two methods were envisioned for successfully completing this task.

Method I

i) Have the robot go forward until its front IR sensor trips.
ii) Then, have the robot reverse direction and count the time required for the rear IR sensor to trip. This time is proportional to the chord length traversed.
iii) Turn the robot through a small angle and repeat steps (i) and (ii) until a maximum chord length is reached (the diameter).
iv) Divide this length in half (the radius), go forward this distance and stop. The robot should now be at the center of the circle.

Method II

i) Have the robot go forward until its front IR sensor trips.
ii) Then, have the robot reverse direction and count the time required for the rear IR sensor to trip. This time is proportional to the chord length traversed.
iii) Divide this distance in half (bisect the chord), go forward for this distance and make a 90 degree turn (the robot has just bisected the chord).
iv) Go forward until the front IR sensor trips, then have the robot reverse direction and count the time required for the rear IR sensor to trip. This time is proportional to the length of the diameter.
v) Divide this number in half, go forward this amount (the radius) and stop. The robot should now be at the center of the circle.

In completing this portion of the project, most groups used Method II, and were able to figure it out on their own in the space of 2 or 3 class hours (with a little help).

Task #2 is called Solo Sumo Wrestling and matches the robot against a wooden block opponent. In this case the robot is required to find the block, push it out of the ring without going out of bounds, and then return to the center of the circle and stop (Figure 7).

In task #3, two robots are placed inside of the Sumo ring and the object is to maximize the points received during each match. The competition rules are summarized in Figure 8.

Assessment of the Overall Student Experience With the Course

Student response to the robot programming experience as a way to understand about Computer Engineering were extremely favorable. Even students who did not like programming and who had no intention of entering the Computer Engineering field, still said that they still found the module to be useful as a way to understand what this type of engineering is all about.

Over the course of the Fall, 2003 semester, a group of selected student comments were collected and are presented below with only moderate editing of these comments:

1. At first the task given to us seemed impossible. However, after the concepts were

Sumo Wrestling Competition Rules

1. Each team competes in three 2 minute matches.
2. The match starts with both rear wheels of each robot on opposite sides of a diameter of the circle
3. Points are awarded in the following way:
   a) 25 points for knocking an opponent out of the ring. The definition of "out" is any wheel or caster ball outside of the white circle for any amount of time. The match ends when this happens, and points accumulated before this point still count towards the overall score.
   b) 15 points if an opponent goes outside of the ring by itself. The match ends when this happens as in (a) above.
   c) 2 points are awarded for each hit scored against an opponent (front, back or sides). However, for a team to receive the points the LED on the team's robot must go on and remain on for three seconds. During the time that the LED is on the robot may continue to move but additional hits by either team do not count.
   d) The overall winner is the team having the highest total score from the three matches.

Figure 8. Summary of the Rules for Robot Sumo Wrestling**.**

explained we realized that it wouldn't be too difficult.

2. The lab was extremely difficult, beneficial, and enjoyable. It challenged the group to think through problems and glitches, and work to correct them. It was rewarding to watch the robot complete the maze after we got the programs to work properly.

3. This module was very difficult at first; it was like learning a different language. Learning to program a robot was incredible and simple. It was astonishing to see a piece of machinery do exactly what you told it to do.

4. At first, as we moved from each example to the next, we weren't sure where to start or how to go about it. However, as we progressed we soon learned the importance of the flowcharts; they enabled us to see where we should begin the program. Each time one of the programs worked, we would become overjoyed.

5. The module was a fun learning experience. The programming was easy to grasp because of the flowcharts and the instruction list sheet. Overall the programming of a robot was an exciting way to demonstrate how a Computer Engineer uses software to make an inanimate object complete a task.

6. The programming experience as a whole was methodical and logical. The programs that built up to the final maze run were ideal in that each was a useful tool in building the final program. This technique allowed us to learn a great deal about programming in a short time as we enjoyed ourselves by manipulating the robot.

7. The experience and time spent in this Computer Engineering module was wonderful and very insightful. We enjoyed the company of a programmable robot that taught us the basics of computer programming as well as valuable techniques in engineering such as patience, commitment, team work, and overall pride in a job well done.

Bibliography

(1) *Basic Stamp Programming Manual V2.0C*, http://www.parallax.com/dl/docs/prod/stamps/BasicStampMan.pdf, November 2000.

(2) *Board of Education Rev C Manual*, http://www.parallax.com/dl/docs/prod/sic/boeman.pdf, July 2003.

(3) Buse, L., Variable Speed Control Modification to the Futaba S3003 RC Servo, http://www.seattlerobotics.org/encoder/200009/S3003C.html, August 2000.

(4) Edwards, S., *Programming and Customizing the Basic Stamp Computer , 2nd Edition*, McGraw-Hill, 2001.

(5) Williams, A., *Microcontroller Projects Using the Basic Stamp, 2nd Edition*, CMP Books, 2002.

(6) Whillock, R., *Sumo Robot Building Hints,* http://www.tcrobots.org/articles/a03.htm, July 2003.

(7) Sumo-Robot Wrestling Competition and Robotics Expo 2002. http://www.robots.org/2002_Spring_Robot_Games.htm, March 2002.

(8) Tilden, M., *Eastern Canadian Robot Games Sumo Robot Wrestling Official Contest Rules,* http://www.robotgames.ca/rules/MastersMiniSumo2003.pdf, November 2003.

ROBERT MAURO is a Professor of Electrical and Computer Engineering at Manhattan College.