Infusing Cyber-Physical Systems Concepts into Computer Science Curricular

Module: Trial and Error

Release 0.1

April 30, 2013

Contact: huichen@ieee.org

1 Overview

You will learn how to use a program to control a robotic vehicle to follow a circular track repeatedly. From the experiments, you will learn the concepts of the "trial and error" method, embedded control using computer programs, and solution evaluation to a technical problem.

1.1 Lab Requirement

- A *Boe-Bot* robot.
- A PC with the *BASIC Stamp Editor* software installed
- A large circular track and a small circular track
- Program *circlerun.bs2*.

1.2 Programming Boe-Bot

The BASIC Stamp Editor[1] is a programming tool for the Boe-Bot. It runs on a PC. Having launched the software, you can edit, check syntax, and run a program. Figure 1 shows a snapshot of the BASIC Stamp Editor.

To program a Boe-Bot, we must connect the Boe-Bot to the PC that has had the BASIC Stamp Editor installed using either a Serial (RS232) cable or a USB A-to-Mini-B cable. We then select "Run" from the "Run" menu to check the synatx, compile the program, install the program to the Boe-Bot, and start the program on the Boe-Bot. Note that we must set the 3-position switch on the Boe-Bot to the center position (i.e., position 1) to allow running any servo motor control programs.

[1] See `http://www.parallax.com`

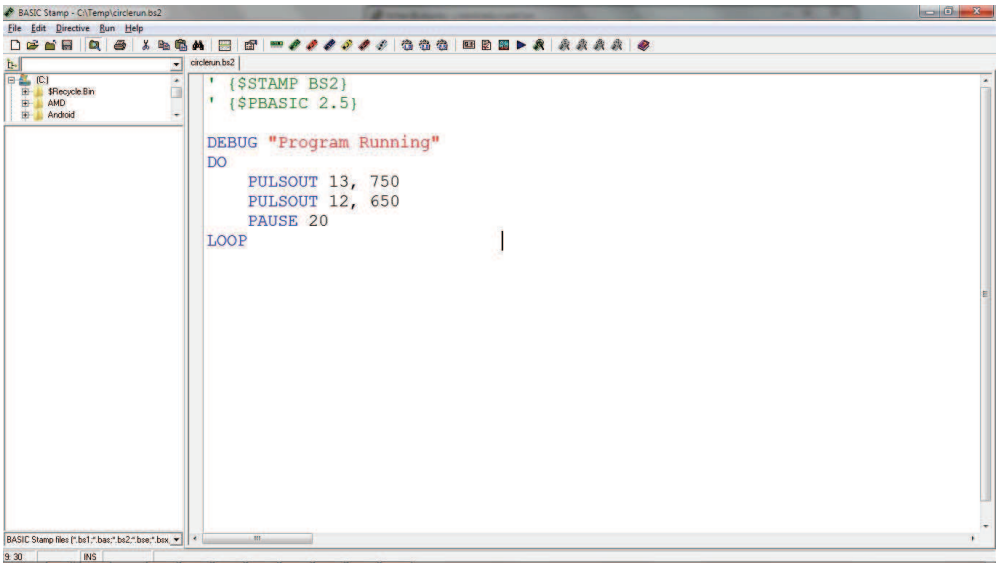

Figure 1: Using the BASIC Stamp Editor to edit and run a program

1.3 Running Circles

Program listing 1 shows the program that controls the robot to run in circles. The program has an infinite loop. Within the loop, Lines 5 and 6 are two statements to send two pulses of different width (called pulse width hereafter, indicated by the second parameter of command PULSEOUT) to pins 12 and 13 (indicated by the first parameter of command PULSEOUT) respectively. The pulses controls revolution speed of the two servo motors on which the two wheels mounted respectively.

Listing 1: Program circlerun.bs2

```
1  ' {$STAMP BS2}
2  ' {$PBASIC 2.5}
3
4  DEBUG "Program Running"
5  DO
6      PULSOUT 13, 850
7      PULSOUT 12, 750
8      PAUSE 20
9  LOOP
```

We denote the pulse width as w. The pulse width is measured in a unit of 2 μs. Given that both of the two servo motors are properly centered, Table 1 provides a description of valid pulse width that a user can use without causing *irreversible damage* to serve motors.

2 Tasks

After we introduce the concepts that the Boe-Bot can be programmed from a PC and that the program controls the behavior of the robot by changing the pulse widths in Lines 5 and 6, you shall proceed with the tasks that follow.

Table 1: Valid pulse width for Boe-Bot servo motors

Pulse Width (w)	Description
$650 \leq w < 750$	The servo motor turns clockwise. The smaller w is, the higher speed the servo motor turns. When $w = 650$, the servo motor turns at full speed.
$w = 750$	The servo motor stalls.
$750 < w \leq 850$	The servo motor runs counterclockwise. The greater w is, the higher speed the servo motor turns. When $w = 850$, the servo motor turns at full speed.

2.1 Task 1: Behavior of Boe-Bot and Pulse Width

Through this task, you will 1) gain hands-on experience on programming a Boe-Bot from a PC using the BASIC Stamp Editor and 2) establish an association between the robot vehicle trajectory and pulse width. You should follow the steps below.

1. Connect the Boe-Bot to the PC.
2. Launch the BASIC Stamp Editor from the PC
3. Input program *circlerun.bs2* as shown in Program Listing 1 to the Basic Stamp Editor.
4. Click the "Run" menu entry form the "Run Menu".

   If the 3-position switch on the Boe-Bot is at the center position (i.e., position 1), the Boe-Bot should be running and its trajectory is the smallest circular path since only one wheel is turning. An individual can choose to disconnect the Boe-Bot from the PC to allow it runs more freely.
5. Change the pulse width in Line 7 to 650 and run the revised program.

   The Boe-Bot should now go along a straight path, which is considered as a circular path of the greast radius (i.e., the infinity).

   Note that due to many factors, such as imperfect centering of the servo motors, imperfect build of the robot, and the unevenness of the surface, the robot may eventually deviate from the straight line path.
6. Change the pulse width in Line 7 to a value between 650 and 750, and then run the revised program.

   You may choose any value between 650 and 750. You shall observe that the robot travels in a circular path whose radius is greater than that observed at step 3 and less than that observed at step 5.

2.2 Task 2: Trial and Error

In this task, you are to revise program *circlerun.bs2* so that the robot will travel endlessly following a given circular track whose radius is different from any one obtained from Task 1. Figure 2 demonstrates a circular track and the robot must run within the circular track.

We know from the experiments in Task 1 that program *circlerun.bs2* actually makes the robot travel endlessly in a circular path. We also know from the experiments that the robot will travel at circular paths of different radius when the pulse width in Line 7 of the program takes different values. The problem then reduces to finding an appropriate pulse width in Line 7. One simple solution is that we try a guessed value of the pulse width in Line 7 if the pulse width value results in a circular path either too large or too small, i.e., an *error*, we then try a different pulse width value. We keep trying different values of pulse width until a pulse width value leads to circular path with a radius that is close enough to the radius required by the circular track. This solution is an example of the *Trial and Error* method. The advantage of the method is that it requires little knowledge of the problem and from the *Trial and Error* process an individual can gradually gain insight of the problem.

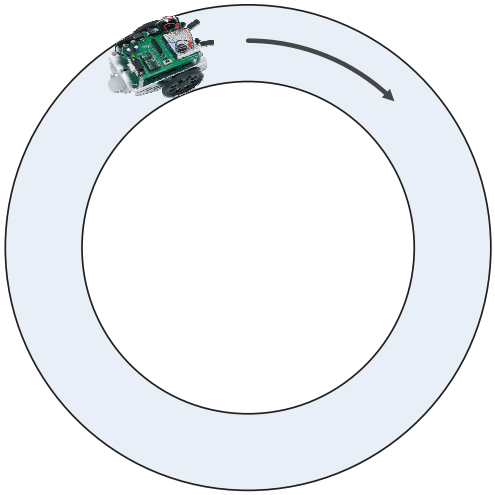

Figure 2: The robot runs within a circular track.

To make the solution comparable, you are not only to experiment with the *Trial and Error* method but also to evaluate the effectiveness of the technical solution, i.e., the *Trial and Error* method from two aspects, the efforts required to find an appropriate pulse width and the accuracy of the circular path. In other words, you are to answer two questions, 1) how accurate is the circle? 2) and how many trials have been conducted? To answer question 1, you should indicate the radius of the given circular path and that of the resulting circular path and their difference. It is favorable that you express the difference in two different ways, absolute error and relative error. Denote the radius of the given circular track as r_t and the radius of resulting circular path as r_p. The absolute error is $e = |r_t - r_p|$ and the relative error is $e\% = |r_t - r_p|/r_t = e/r_t$.

The intent of this course module is not to write the program itself. Instead, the intent is to allow you through experiments to learn that the behavior of a physical system (i.e., a robot) can be controlled by a program, that the quality of a technical solution needs to be evaluated, and that some problem does not require an exact solution (for instance, the final radius of the circle that the robot follows does not have to be exact; however, it needs to be approximate enough, which is a departure from many programs that deal with discrete structures).

3 Submission

The instructor requires you to submit the following items,

- A brief recitation of the problem description and the technical solution;
- A journal of each trial with the value of pulse width and whether the resulting radius is too big or too small;
- The answers to the two evaluation questions 1) how accurate is the circle? 2) and how many trials have been conducted?