Infusing Cyber-Physical Systems Concepts into Computer Science Curricular

Module: Binary Search

Release 0.1

May 2, 2013

1 Overview

You will learn how to use a program to control a robotic vehicle to follow a circular track repeatedly. Through experiments, you will learn an algorithm called "binary search", the concept of algorithm efficiency, embedded control using computer programs, and solution evaluation to a technical problem.

1.1 Prerequisite

We recommend that you complete the "Trial and Error" module before you start this module. In the "Trial and Error" module you can obtain experience in programming a Boe-Bot and experimenting with program *circlerun.bs2*. Although not recommended, you may treat this module as a standalone one without trying the "Trial and Error" module. We refer you to the "Trial and Error" module for a basic introduction to programming Boe-Bot and program *circlerun.bs2*.

1.2 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*.

1.3 Programming Boe-Bot

See the "Trial and Error" module.

1.4 Running Circles

We introduced Program *circlerun.bs2* in the "Trial and Error" module. For your convenience, we list the program in program listing 1.

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
```

2 Tasks

You are to solve the same problem in the "Trial and Error" module, i.e., to revise program *circlerun.bs2* so that the robot will travel endlessly following a given circular track whose radius is different from those obtained from previous experiments. Figure 1 demonstrates a circular track and the robot must run within the circular track. As discussed in the "Trial and Error" module, the problem reduces to the finding of an appropriate value of the pulse width in Line 7 of program circlerun.bs2, i.e., the second parameter in Line 7 that controls the revolution speed of the servo controlled by pin 12. However, different from the "Trial and Error" module, you are to use a new method called "Binary Search". Compared to the "Trial and Error" method, the "Binary Search" method is a technical solution with increased complexity, however as we will learn, also with higher efficiency and more controllable accuracy.

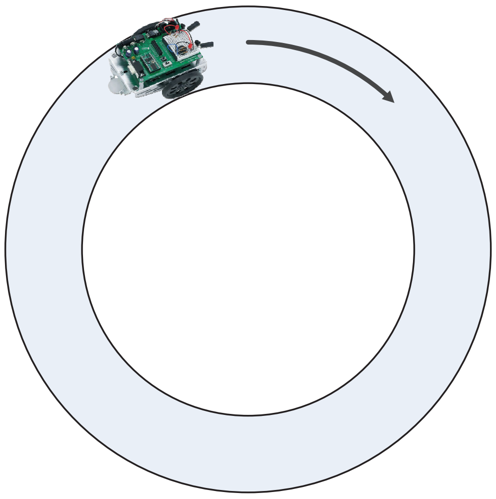


Figure 1: The robot runs within a circular track.

2.1 Task 1: Binary Search Method

From the experiments, you will learn an algorithm called "binary search", the concept of algorithm efficiency, embedded control using computer programs, and solution evaluation to a technical problem.

The "Binary Search" method is inspired from an computer algorithm of the same name, the "Binary Search" algorithm. The method is grounded on the observation,

Observation 1. When the pulse width in Line 7 of program *circlerun.bs2* causes the robot to follow a larger circle than desired, a smaller (or larger depending on factors such as how server motors are connected or mounted) value will make the circle even larger and when a value causes a smaller circle, a larger (or smaller) value will make the circle even smaller.

The "Binary Search" method is a method that systematically narrows the search space (i.e., the interval of two values of the parameter where the good value of the parameter is in) in half. As a result, the method can converge exponentially to a value that leads to a sufficiently accurate pulse width for the problem. For instance, when the pulse width in Line 7 is 700, the vehicle travels along a circular path of moderate radius; when the pulse width is 650, the vehicle travels along a circular path of infinite radius; and when the pulse width is 750, the vehicle travels along a circular path of the smallest radius. If we start with pulse width 700, if the circular path is too large, we can only let the pulse width be smaller; otherwise, we would obtain an even larger circular path. Therefore, we can in effect eleminate the half of the search space of the pulse width, i.e., interval $[700, 750]$ and in next test run, we only need to search the pulse width within interval $[650, 700)$.

Through this task, you will 1) learn to use the "Binary Search" method to find a sufficiently good value for the pulse width in Line 7 of program *circlerun.bs2*, 2) gain more hands-on experience on programming a Boe-Bot from a PC using the BASIC Stamp Editor, and 3) establish an association between the robot vehicle trajectory and pulse width. You should follow the "Binary Search" method as illustrated in Figure 3. We also outline the steps also below,

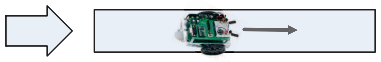
1. Connect the Boe-Bot to the PC.
2. Launch the BASIC Stamp Editor from the PC
3. Input program *circlerun.bs2* as shown in Program Listing 1 to the Basic Stamp Editor.
4. Set up the search space for the pulse width in Line 7. Recall that the valid values for the pulse width in Line 7 of program *circlerun.bs2* must be no less than 650 and no greater than are 750. The search space is in effect interval $[650, 750]$. For your convenience, denote the left boundary of the search space as $w_L = 650$, and the right boundary $w_R = 750$, i.e., the search space is always $[w_L, w_R]$. The solution to the problem must be within the search space.
5. In the "Binary Search" method, you should always try the middle value, we always choose the middle value of the search space for a test. Then you shall calculate the pulse width that is to be tested as follows,

   $$w = \left\lceil \frac{w_L + w_R}{2} \right\rceil \tag{1}$$

   Note that $\lceil\ \rceil$ is a "ceiling" operation that maps to the smallest following integer, i.e., $\lceil x \rceil$ is the smallest integer not less than x.

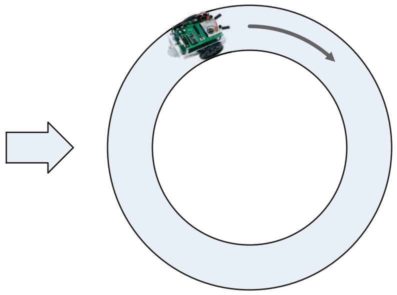
   For instance, if $w_L = 650$ and $w_R = 750$, we calculate the pulse width to be tested as $w = \lceil (650 + 750)/2 \rceil = 700$.

```
1  ' {$STAMP BS2}
2  ' {$PBASIC 2.5}
3
4 DEBUG "Program Running"
5 DO
6       PULSOUT 13, 850
7       PULSOUT 12, 650
8       PAUSE 20
9 LOOP
```



(a) When the pulse width at Line 7 of program *circlerun.bs2* is 650, the vehicle travels in a straight line, i.e., a “circular” path of infinite radius.

```
1  ' {$STAMP BS2}
2  ' {$PBASIC 2.5}
3
4 DEBUG "Program Running"
5 DO
6       PULSOUT 13, 850
7       PULSOUT 12, 700
8       PAUSE 20
9 LOOP
```



(b) When the pulse width at Line 7 of program *circlerun.bs2* is 700, the vehicle travels in a circular path of moderate radius.

```
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
```

(c) When the pulse width at Line 7 of program *circlerun.bs2* is 750, the vehicle travels in a circular path of the smallest radius

Figure 2: The vehicle travels along circular paths of different radius.

6. Replace the pulse width in Line 7 of program *circlerun.bs2* by the value of w. 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.

   You shall observe the radius of the circular path along which the vehicle travels and compare with the radius of the given circular track. For your convenience, denote the radius of the circular path along which the vehicle actually travels as r_p and the radius of the given circular track r_t.

   According to the principle of the “Binary Search”, we can eliminate half the search space and reset the search space for next test run. If the radius of the circular path is greater than that of the given track, you shall make the pulse width greater as a smaller pulse width leads to a circular path of greater radius, i.e., eliminate the half search space that is greater than w and the new search space becomes

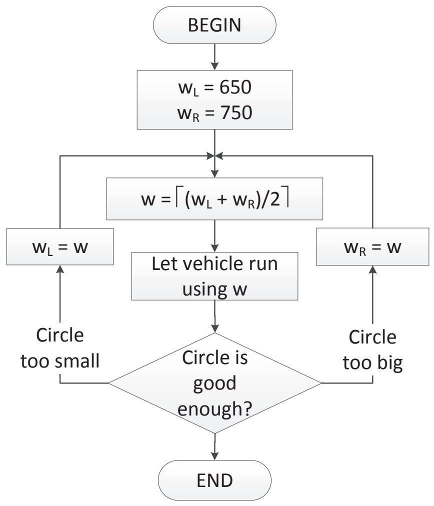


Figure 3: Binary search method

$[w_L, w]$ and vice versa.

7. if $r_p > r_t$, let $w_R = w$; if $r_p < r_t$, let $w_L = w$; if r_p and r_p are sufficiently close, go to next step. Repeat above steps starting at step 5. You must note the values of r_p and w. Note that in the "Binary Search" method, we call steps 5–7 as an iteration. In each iteration, you test a different value and decide which half of the search space you will seach next.

8. The value obtained in previous step is the solution. Halt the experiment.

2.2 Task 2: Evaluation: Efficiency and Accuracy

You are to evaluate how efficient the "Binary Search" method is and the how accurate the solution can be. The efficiency reflects the effort one must make before she finds a satisfactory pulse width value. We use the number of test runs as the efficiency measurement. The same as the "Trial and Error" method, we measure the accuracy of the solution 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 $e\% = |r_t - r_p|/r_t = e/r_t$.

Your task is to fill up Table 1 with the results obtained for at least two circular track of different radius. If you are working on this module without trying the "Trial and Error" module, you may skip the columns in Table 1 for the "Trial and Error" method.

Table 1: Efficiency and Solution Accuracy of the "Binary Search" Method

No.	Radius of Track	Binary Search			Trial and Error		
		# of Trials	Absolute Error	Relative Error	# of Trials	Absolute Error	Relative Error
1	$r_t =$						
2	$r_t =$						
3	$r_t =$						

3 Discussion

Observing Table 1, what would you conclude from it? Are you convinced that the "Binary Search" method is a more efficient method? In fact, the number of the values of the pulse width that you must try is bounded by a constant. We prove it as follows.

The initial size of the search space is $L_1 = \lceil 750 - 650 \rceil = 100$. The "Binary Search" method shrinks the search space in half at each iteration. After the 1st iteration, the search space shrinks in half. The size of the search space for the 2nd iteration becomes becomes $L_2 = \lceil 100/2 \rceil = \lceil 100/2^{2-1} \rceil = 50$. Similarly, the size of the search space for the 3rd iteration becomes $L_3 = \lceil 50/2 \rceil = 25 = \lceil (100/2)/2 \rceil = \lceil 100/2^2 \rceil = \lceil 100/2^{3-1} \rceil$; after the 3rd, $L_3 = \lceil 25/2 \rceil = \lceil 12.5 \rceil = 13 = \lceil 100/2^3 \rceil = \lceil 100/2^{4-1} \rceil$. Figure 4a is an example showing the search space shrinks exponentially. In summary, we observe that the search space shrinks in the patter below,

$$\begin{aligned}
L_1 = 100 &= \left\lceil \frac{100}{1} \right\rceil = \left\lceil \frac{100}{2^0} \right\rceil = \left\lceil \frac{100}{2^{1-1}} \right\rceil \\
L_2 = 50 &= \left\lceil \frac{100}{2} \right\rceil = \left\lceil \frac{100}{2^1} \right\rceil = \left\lceil \frac{100}{2^{2-1}} \right\rceil \\
L_3 = 25 &= \left\lceil \frac{100}{4} \right\rceil = \left\lceil \frac{100}{2^2} \right\rceil = \left\lceil \frac{100}{2^{3-1}} \right\rceil \\
L_4 = 13 &= \left\lceil \frac{100}{8} \right\rceil = \left\lceil \frac{100}{2^3} \right\rceil = \left\lceil \frac{100}{2^{4-1}} \right\rceil \\
L_5 = 7 &= \left\lceil \frac{100}{16} \right\rceil = \left\lceil \frac{100}{2^4} \right\rceil = \left\lceil \frac{100}{2^{5-1}} \right\rceil \\
L_6 = 4 &= \left\lceil \frac{100}{32} \right\rceil = \left\lceil \frac{100}{2^5} \right\rceil = \left\lceil \frac{100}{2^{6-1}} \right\rceil \\
L_7 = 2 &= \left\lceil \frac{100}{64} \right\rceil = \left\lceil \frac{100}{2^6} \right\rceil = \left\lceil \frac{100}{2^{7-1}} \right\rceil
\end{aligned}$$

Since the value of pulse width must be a whole number, at the 7th iteration, the size of the search space becomes 2 and you no longer need to continue. One of the end value of the search space must give you the most accurate circlar path.

In general, as you have observed, the size of the search space shrinks exponetially. Denote L as the size of the initial search space, the size of the search space at the i-th iteration is,

$$L_i = \left\lceil \frac{L}{2^{i-1}} \right\rceil \tag{2}$$

We halt the "Binary Search" method at the the n-th iteration at which the size of search space becomes 2,

$$L_n \left\lceil \frac{L}{2^{n-1}} \right\rceil = 2 \tag{3}$$

which leads to,

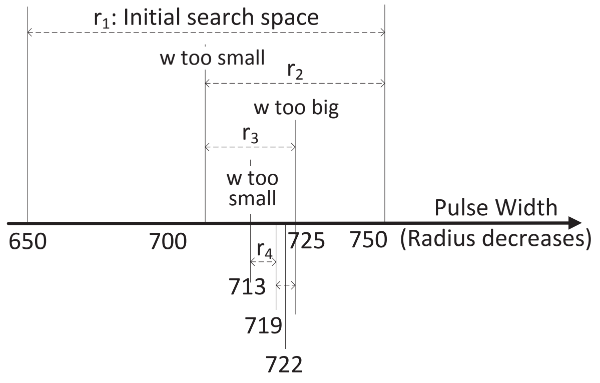
$$1 < \frac{L}{2^{n-1}} \leq 2$$

Multlying 2^{n-1} to the inequality,

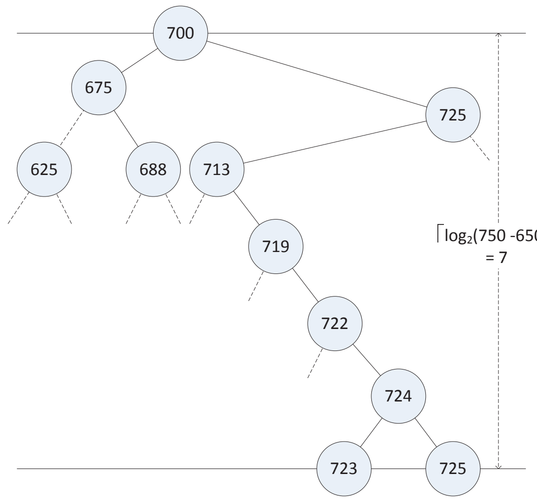
$$2^{n-1} < L \leq 2^n$$

Applying $log_2(\cdot)$ to the inequality,

$$log_2 2^{n-1} < log_2 L \leq log_2 2^n$$



(a) Search spaces shrink



(b) Binary search tree

Figure 4: The Binary Search method

$$n - 1 < log_2 L \leq n$$

According to the definition of $\lceil \cdot \rceil$, the number of iterations (or tests) is bounded by

$$n = \lceil log_2 L \rceil \tag{4}$$

For example, we apply the above result to Task 1, the maximum number of the value of the pulse width must be no greater than $\lceil log_2 L \rceil == \lceil log_2(750 - 650) \rceil = \lceil log_2(100) \rceil \approx \lceil 6.644 \rceil = 7$, i.e., *for any given circular track, we need at most 7 trials to find a sufficiently accurate value of the pulse width using the "Binary Search" method.* This result can also be resulted in a "tree" as shown in Figure 4b.

4 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? Attach Table 1 and explain how efficient the "Binary Search" method is, how controllable the accuracy is, and under what condition the most accurate solution can be obtained.
- Answer the following questions.

  If the size of the initial search space is 10, how many trials do you need at most to find the solution to the problem? What if the size of the initial search space is 200? What if it is 500? What if it is 1,000, 10,000, and 20,000? Can you graph the number of trials versus the size of the initial search space?