

Activity 4.2.7 Variables and Functions

Introduction

A program can accomplish a given task in any number of ways. Programs can quickly grow to an unmanageable size, so variables and functions provide a technique to reduce the size of the program. A variableis a space in your robot’s memory where you can store data, such as whole numbers, decimal numbers, and words. Functionsgroup together several lines of code, which can then be referenced many times in task main or even other functions. The use of variables and functions allows for complex control of a system.

Equipment

* Computer with ROBOTC software
* AE VEX®testbed
* PLTW ROBOTC template

Procedure

1. Form groups of four and acquire your group’s AE VEX® Kit under your teacher’s direction.
2. Within your four-student group, form a two-student team known as Team A and a two-student team known as Team B.
	1. Team A will construct the VEX® Testbed with the exception of the ultrasonic and the light sensor.
	2. Team B will construct the VEX® Testbed with the exception of the servo motor and flashlight.
3. Connect the AE VEX®testbed Cortex to the PC.

|  |
| --- |
|  |
| AE VEX® Testbed |

**Part 1: Using a variable to loop n times**

1. Open the PLTW ROBOTC template. Click File, Save As, select the folder that your teacher designated, and then name the file A4\_2\_7\_Part1.
2. In this activity you will use the green LED, right motor, and bumper switch. Leave the previously connected motors and sensors wired to the Cortex. Go to the Motors and Sensors Setup window. Configure the Motors and Sensors Setup to reflect the inputs and outputs to be used. Note that additional motors and sensors that are physically attached may be configured; however, these are not required to be configured. Click OK to close the window.

|  |
| --- |
| **Cortex Wiring Diagram** |
|  |

1. Use the program below in the task main()section of the program between the curly braces.

|  |
| --- |
| intmotorCount;motorCount=0;while(motorCount<3){startMotor(rightMotor,95);wait(2);stopMotor(rightMotor);wait(2);motorCount=motorCount+1;} |

1. Save the program, power on the Cortex, ccompile, and download the program. If you have any errors, check with your instructor to troubleshoot your program.
2. Press Start to run the program and observe the behaviors.
3. Document what this program would look like as pseudocode simple behaviors.
4. Create a program that will run a motor back and forth 20 times, 0.5 seconds each way. Test the program and troubleshoot until the expected behavior has occurred. Make sure your code is documented with a task description, pseudocode, and line-by-line comments. Save the program. You will reuse this program in Activity 3.1.6.

**Part 2: Using a variable to remember a maximum or minimum value**

**and using debugger windows**

1. Open the PLTW ROBOTC template. Click File, Save As, select the folder that your teacher designated, and then name the file A4\_2\_7\_Part2.
2. Use the program below in the task main()section of the program between the curly braces.

intbiggest;

while(1==1)

{

biggest=0;

while(**SensorValue**(bumpSwitch)==0)

{

**if**(**SensorValue**(potentiometer)>biggest)

{

biggest=**SensorValue**(potentiometer);

}

}

 }

|  |
| --- |
|  |

1. Select View, Preferences, Compiler, Code Generation, andthen select Force ‘Classic’as the Code Generation For Stack. Click OK. Compile and download the program to the Cortex.

|  |
| --- |
|  |

1. Use the Robot and Debugger Windows menu to make sure that both the Global Variables and Sensors tabs are checked. See the figure for how the menu should appear.

|  |
| --- |
|  |

1. These debugger windows can be placed wherever they are convenient for you on the ROBOTC screen. To read about this, choose Help, Open Help, and then in the Contents tab, expand ROBOTC Debugger by clicking on its plus sign. Select Docking Debugger Windows. Steps 16-18 repeat the information from the built-in help. If you are able to learn from the built-in help, just make your debugger windows look like the picture in step 17, and then continue with step 19.
2. To move the debugger windows, click on the title bar of the debugger window and drag it to the desired location. If the Global Variables and Sensors debugger windows are sharing a single pane with separate tabs, you can tear one of the tabs off and then make it into its own window by clicking on the tab and dragging it to a new location. Make two separate windows, one for the Sensors debugger window and one for the Global Variables window.

|  |
| --- |
|  |

1. Place the Global Variables and Sensors debugger windows on top of one another so that you can easily see the value of the variable “biggest” and the value of the sensor potentiometer at the same time. The windows might look like the image below.

|  |
| --- |
|  |

1. Later, you can make the debugger windows dock again at the bottom of the code editing pane. Drag the debugger window’s title bar and release the mouse on one of the icons shown in the figure below. The debugger window will dock at one edge of the ROBOTC window or of the selected ROBOTC pane. A debugger window can also be placed back into a tab within another debugger window this way.

|  |
| --- |
|  |

1. Start the program and slowly turn the potentiometer back and forth, using the debugger windows to monitor the value of the variable “biggest” and the value of the potentiometer. Press the bump switch and note the effect. Continue to turn the potentiometer, press the bump switch, and monitor the variable and sensor values until you think you understand how the code works. What does this program do? Annotate the program with a task description, pseudocode, and line-by-line comments.
2. Click File, Save As, select the folder that your teacher designated, and then name the file A4\_2\_7\_Part2\_ Modify\_A.
	1. If you have the ultrasonic distance sensor, modify the program so that a variable “closest” will remember the closest distance detected by the ultrasonic distance sensor. The limit switch should reset the record.

OR

* 1. If you have the light sensor, modify the program so that a variable “brightest” will remember the brightest light detected by the light sensor. The limit switch should reset the record.
1. Test your program and troubleshoot until the expected behavior has occurred. Make sure your code is documented with a task description, pseudocode, and line-by-line comments. Save the program.

**Part 3: Define a function**

1. Open the PLTW ROBOTC template. Click File, Save As, select the folder that your teacher designated, and then name the file A4\_2\_7\_Part3.
2. Use the program below in the task main()section of the program between the curly braces.

voidLEDControl(); //this is a function declaration

task main()

{

while(1==1)

{

LEDControl();//function call

}

 }

voidLEDControl()//function definition

{

if(SensorValue(bumpSwitch)==1)

{

turnLEDOn(green);

}

else

 {

turnLEDOff(green);

 }

 }

1. Test the program and troubleshoot until the expected behavior has occurred. Save the program.
2. Describe the behaviors observed.
3. Follow teacher direction and either print the programs or submit electronically with this activity.

Conclusion

1. Describe any challenges that you encountered while developing the program.
2. Describe applications for whichvariables and functions could be used.