ASSIGNMENT 4

GENE-121-2017-3F-ASSN-04-Description

Fall 2017

CONTENTS

INTRODUCTION

This assignment is to be completed in pairs.

In this assignment, you will be introduced to the Lego Mindstorms platform and real-time programming.

At the start of the lab session please sit at the robot kit with the number you have been assigned from the partner list. Please do not alter the physical construction of the robot. When you are finished, do not dismantle the robot. Simply return the robot in its original configuration to a TA.

Note that you can complete Question 1 of this assignment without needing a robot (i.e. can be done before the tutorial session).

LEGO ROBOT SETUP

You do not need hand in any code for this part of the assignment. These instructions are just to get you up and running on the EV3.

STARTING ROBOTC

Turn on the Lego EV3 robot by pressing the center button. It will take a minute to boot up.

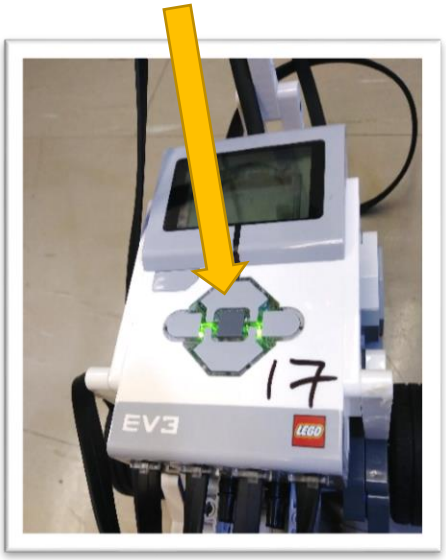

Figure 1. Power on button

Connect the EV3 robot to the computer with the USB cable. You should see a USB connectivity indicator (circled in image) on the EV3 display.

Figure 2. USB connectivity indicator

Start the RobotC software called "ROBOTC for LEGO MINDSTORMS". (Note that you do NOT want to run the "Graphical ROBOTC for LEGO Mindstorms 4.X" version of the app.)

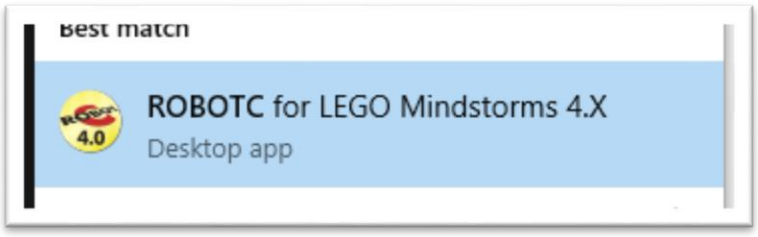

Figure 3. ROBOTC app

SAMPLE PROGRAM

Verify that the two drive motors are connected to motor ports A and C.

Click on "File" -> "New File" to start a new program.

Enter the following code (which contains two errors):

```
task main()
{
  for (run_five_times = 0; run_five_times < 5; run_five_times++)
  {
    motor[motorA] = 75; motor[motorC] = 75; wait1Msec(2000);

    motor[motorA] = 0 motor[motorC] = 0; wait1Msec(1000);
  }
}
```

Save the program.

Compile the program. The compiler will indicate that it was unable to compile the program. You should see the window at the bottom of the screen indicate "Compiler Errors". There will also be red 'X's in the margin at the lines where the errors are located.

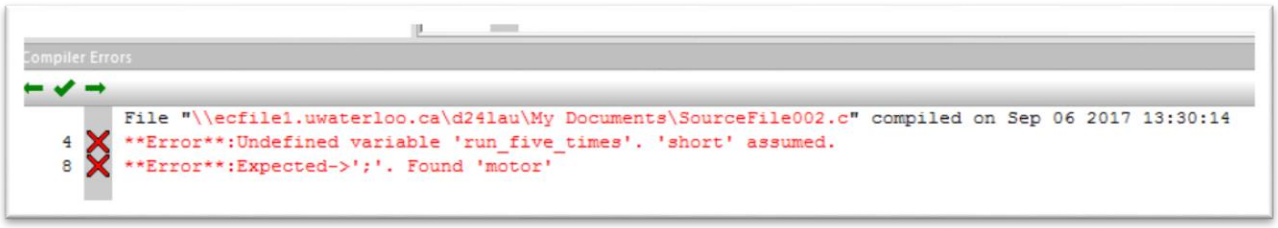

Figure 4. Compiler errors

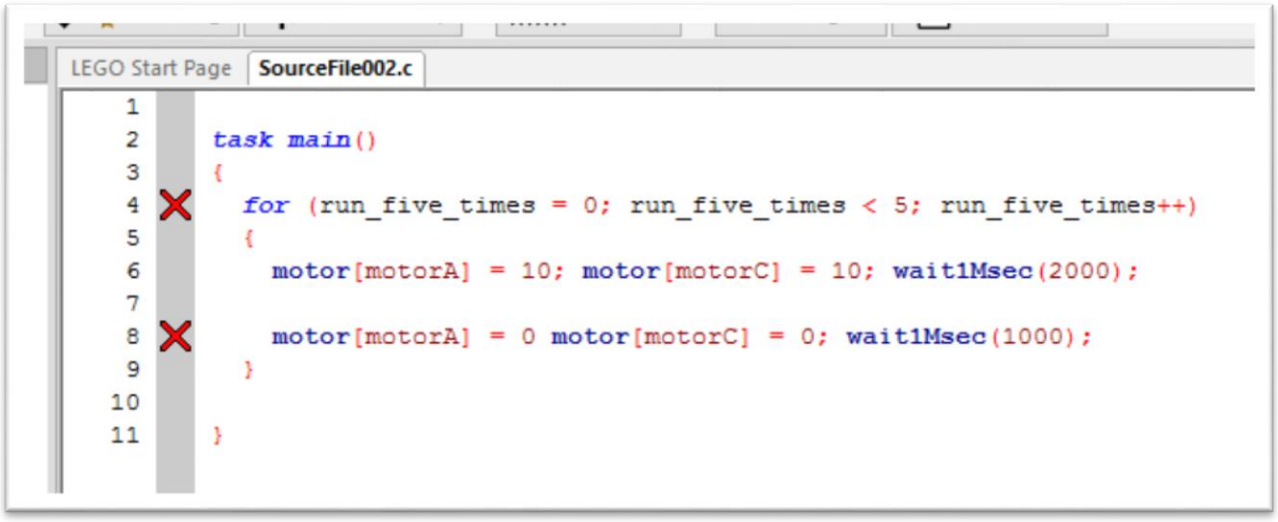

Figure 5. Error locations marked with red X's

Fix the errors and compile again.

Once you are able to compile the code successfully, download the program to the robot. The "Program Debug" window will appear. You can start the program by clicking the "Start" button from this window, or by running the program from the robot directly. The advantage of running the program from the desktop computer is that you can trace variable values on the computer for debugging purposes. The advantage of running the program from the robot directly is that you do not need to be tethered to the computer with the USB cable.

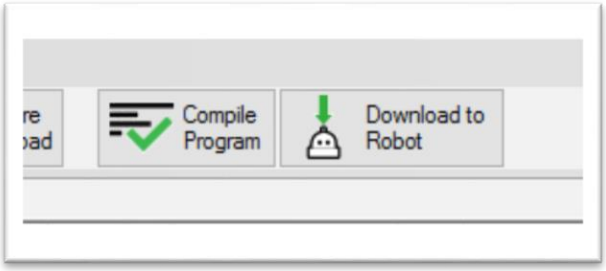

Figure 6. Compile and Download buttons

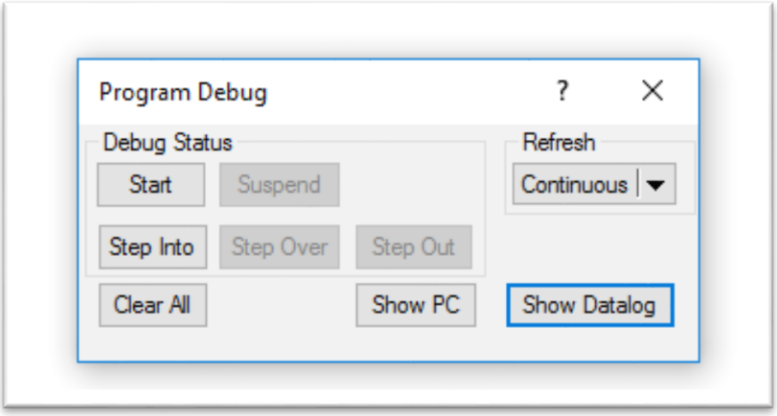

Figure 7. Program Debug window

QUESTION 1 – FLOWCHART (ROBOTC)

Flowcharts can be a very helpful tool to help design real-time program flow.

Draw a flowchart and then write a program that:

1. Displays the lab session and your group (robot) number on line 0 of the display. For example,

```
Tues early 07
```

2. Repeat for three button presses:
   - If the right button (on the brick) is pushed and released, the robot should rotate in place 90 degrees clockwise
   - If the left button (on the brick) is pushed and released, the robot should rotate in place 90 degrees counter-clockwise
   - If the center button (on the brick) is pushed and released, the display should say "Vamonos" and drive straight at top speed for 2 seconds.
3. Drive the robot in a circle or arc until it runs into something with the front bumper.

Submit your flowchart with the assignment package.

QUESTION 2 – BASIC CONTROLS (ROBOTC)

Write the program for the specification in Question 1. Download the code to your robot and test its behavior.

Once you are satisfied with your robot's behavior, bring your robot to a TA (at the front of the room) to demonstrate your working program. The TA will test your robot's behavior to see if it fully meets the specification. Once the TA is satisfied with your robot's behavior, they will record that you have successfully submitted "output" for your program.

As the demonstration to the TA serves as output for your submission, you are only required to hand in your code for RobotC questions. No "output" section is required for RobotC questions in the assignment package.

QUESTION 3 – NAVIGATION (ROBOTC)

In this program for safety reasons, when the robot bumper hits anything (touch sensor pressed) the robot should immediately stop moving and end the program.

STARTING SEQUENCE

The robot should start by displaying your tutorial section and group number on line 0 of the display, then wait for the center button (Enter) to be pressed and released.

NAVIGATION

The robot should navigate an obstacle course by repeatedly executing the following actions:

- Rotate to the left of its original position (counter-clockwise) by about 45 degrees. Take an ultrasonic measurement.
- Rotate back to its original position. Take an ultrasonic measurement.
- Rotate to the right of its original position (clock-wise) by about 45 degrees. Take an ultrasonic measurement.
- Rotate to the position where the highest ultrasonic measurement was taken. Move forward 40cm in that direction.

ENDING SEQUENCE

The robot should stop moving immediately any time the front bumper is pressed. After it is stopped, the robot should display the time in seconds from when the center button was released and when the robot stopped.

QUESTION 4 – DUCK SEASON (ROBOTC)

In this question, you will be writing a RobotC program. However, you will NOT be running it on an actual EV3 robot. This should serve as good practice in preparation for midterm and final exams, where you will be required to write RobotC programs without access to a computer or robot. You may choose to handwrite your code or to type it into the RobotC editor.

Prof. Hulls has taken up ornithology. She would like your assistance acquiring and tagging ducks on the UW campus. She has provided you with a modified EV3 robot. This robot has the same design as the standard GENE 121 configuration that you have used in the tutorials, with the following changes:

- motorD drives a duck acquisition and tagging system on the front of the robot. Drive this motor in the positive direction at power 25 for 3 seconds to acquire and tag a duck. Drive this motor at power -15 for 1 second to release the duck.

- Spinning the robot at power 75 (one motor forward, one motor backwards) for 400 milliseconds will result in a 90 degree turn.
- The touch sensor bumper has been extended so that it is now 1 meter wide.

Prof. Hulls would like you to use the UW football field to tag the ducks. The dimensions of the field are 101m long by 59m wide. Start your robot on the southwest corner and facing north. Begin the program by waiting for the user to press and release the center button. Systematically drive the robot across the field according to the drawing below.

Since ducks are short and have small heads, when the robot runs into a duck with the bumper, the ultrasonic sensor will give a reading of 255. If the robot bumps into an obstacle (not a duck), the ultrasonic sensor will give a reading less than 255.

When the robot encounters a duck, it should acquire it, tag it, turn 180 degrees, release the duck, turn 180 degrees, and continue traversing the field. When the robot encounters an obstacle, it should stop and wait 3 seconds for the obstacle to move out of the way, and then continue traversing the field. The program should end when it reaches the eastern edge of the field. End the program by displaying the total number of ducks tagged.

State at least one assumption you need to make in order to have your program work correctly. State one thing that could go wrong that would cause your program to fail. (Not related to your assumption.)