NEW YORK CITY COLLEGE OF TECHNOLOGY (City Tech)	Department of Computer Engineering Technology (CET)	CET 4952: Robotics Technology (Elective Course)

Project 1: Waypoint Navigation and Basic Motion Control

This project aims at achieving basic motion control of an antonomous mobile robot on a 2D plane.

Project Description: Build a mobile robot that can move on a 2D plane. Program your robot to go/travel through a given list of waypoints with specified 2D coordinates. The robot should be able to do this autonomously/automatically once the waypoints are given. At the same time, the robot should detect objects/obstacles that are ``seen'' by its onboard range sensor (e.g., sonar). Obtain a plot of robot's own trajectory along with obstacles it detected on the way.

Hardware: VEX Robotic Kit **Software**: RobotC for VEX

Procedures:

1. **Translation (Going to a Distance):** Program your robot to go straight for a specified distance (in meter). The students are required to code a subfunction that takes the specified distance as the input argument and simply travels the specified amount of distance. In the main function, the student can specify the distance, which will be given by the instructor upon evaluation. For example, the instructor may say "let the robot go straight for 2 meters". It is very important that the students code the subfunction, since we are working towards a project.

   When coding this subfuction, use motor encoder readings. Each wheel has one encoder sensor, which provides the degree of rotation (in degree). Assuming that both wheels are synchronized, the distance of travel is a function of the degree of rotation as:

$$\text{distance} = 2 \times \pi \times \text{radius of the wheel} \times \frac{\text{encoder reading}}{360}$$

   - You can name this subfunction as: void go_straight (float dist)
   - You may need to measure the radius of your robot's wheel(s).
   - I need to see the code to make sure motor encoder is used and a subfuction is coded.

2. **Rotation:** Program your robot to be able to rotate in place a specified angle (in radians). Notice that the input angle can be either 0 (no need to rotate at all), positive (rotate counter clockwise), or negative (rotate clockwise). Similarly to "Translation", code a subfunction for this job. Upon evaluation, type in the angle given by the instructor upon evaluation in your main function. For example, the instructor may say "rotate 90 degree counterclockwise or clockwise".

   When the robot needs to rotate in place, one wheel needs to move forward (a positive motor power level) and the other wheel needs to move backward (a negative motor power level). You probably also need to do a little "calibration/tuning" to figure out the relationship between the degree of rotation of the robot and the target encoder value of each wheel.

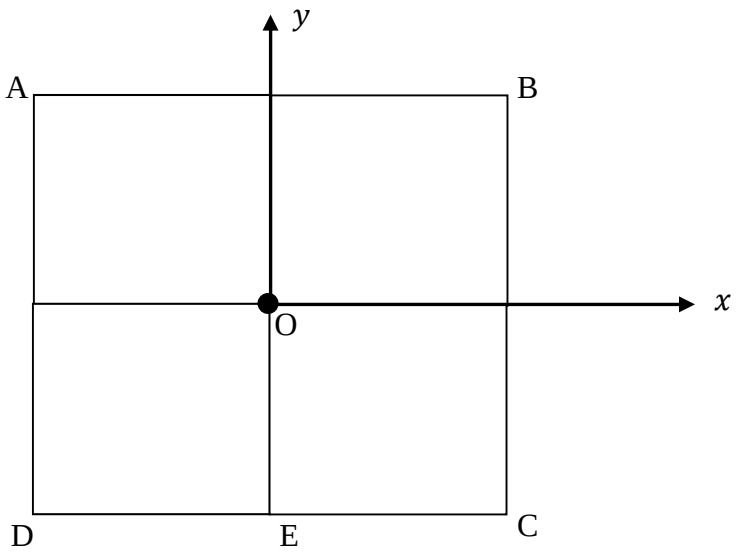
   - You can name this subfuction as: void rotate (float angle_of_rotation)
   - I need to see the code to make sure that the motor encoder is used and a subfuction is coded.

3. **Waypoint Navigation:** Program your robot so that it navigates through a given set of waypoints, where a waypoint is a (X, Y) position relative to the world coordinate system. The default unit of all these waypoints is **meter**.

Coding Help: The instructor should have posted some source code for you to merge with your code. Follow the instructions given in the source code.

One way of going to a waypoint is to turn on the spot through an appropriate angle and then move forward of the right distance. We already know how to let the robot travel a specified distance and rotate certain amount of angle. The challenge of this step is to how to compute this appropriate angle, the right amount of distance (relatively easy), and how to loop through each pair of waypoints by always updating the robot's current orientation and position upon completion of each maneuver.

Suppose the following is the floor of our lab. Upon evaluation, I'll randomly give you a sequence of waypoints to travel, for example, O → B → C → D → E. You type in the coordinates of the given set of waypoints into your program and the robot should navigate through these waypoints automatically one by one. We can assume that the robot always starts from the origin with a 90° orientation w.r.t. the positive x-axis. Your program should also be flexible enough to take different numbers of waypoints.



Evaluation: Make the robot follow the waypoints as closely as possible. This affects your grade. The instructor will measure the error distance between your robot's final position and the goal position (i.e., the last waypoint). Your grade for this step will be given as:

Error Distance (in cm)	**Grade (Out of 10)**
Error Distance $\leq$ 5cm	10
5 cm $<$ Error Distance $\leq$ 15 cm	9
Error Distance $>$ 15 cm but is able to demonstrate the correct navigation pattern	8
Unable to demonstrate the correct navigation pattern	0

4. **Map Building:** On top of all previous steps, add another functionality to your robot by building a map of its surroundings using the data collected from a sonar sensor while performing waypoint navigation. This requires the robot to compute and remember the position of anything detected by the sonar sensor in real time as it translates and rotates. Display the map after the robot comes to the goal location. The students are suggested to mount the sonar facing to the side of the robot, either to the left or to the right.

   Coding Help: The instructor should have provided some sample code help on Blackboard for this lab. Though collection of sensor data can happen during both translation and rotation, to have simple start, let us focus on the translation only. The students need to modify the function where translation is performed (inside the while loop). These is the place to collect sonar data, compute the positions of the detected objects, and then save the computed (x, y) coordinates into an array. The VEX kit does not have a screen big enough to display the trajectory and the sonar points. Instead, we will save all these information in arrays and transfer these information to Desktop computer for post-display using Matlab.

   Evaluation: Your grade for this lab will be given based on the plotted map. The map should represent the environment reasonally well.

5. **Report of the project:** Upon succefully completion of the project, each group needs to write a report including:

   1) **(One Page)**: A cover page showing the course title, project name, and the names of the group members
   2) **(Half Page)** An illstrution figure showing the setup of the project, including the specified trajectory of the robot, how the sonar is installed on the robot (i.e., facing to the left or to the right of the robot), and where the objects/obstacles are located.
   3) **(Half Page)** A picture of your robot
   4) **(Two Pages)** Description of the key components (how each step is achieved/performed)
      a. Translation: describes the relationship between the motor encoder and the distance of traveling
      b. Rotation: describes how the rotation is performed/calibrated
      c. Waypoint Navigation: Using a picture (as the one in homework) to describe how the waypoint navigation is achieved, i.e., rotation first followed by a translation, and how to compute the amount of rotation and translation, respectively.
      d. Map Building: Using a picture (as the one used in the homework) to describe how the position of an obstacle is computed from the robot's current position and orientation, the collected sonar reading, and how the sonar is installed on the robot.
   5) **(One Page)** Experimental Results:
      a. Includ experimental results (the Matlab plot you obtained) showng the trajectory of the robot along with obstacles/objects detected by the onboard sonar sensor. Notice that this should be consistent with your setup described in 2). Describe which cluster of data points is corresponding to which object.
   6) **(One Page)** Your understandings and suggestion(s) for future development:
      a. Describe your learning experience (what you learned)
      b. Describes what to do in order to make the project more challenging, succesful, and interesting.

10 points for each step (Step 1~5)
No Code, No Program in the Report