1. Questions:

Q1. Tasks in RobotC facilitates threadlike parallelization, which enables running separate Tasks simultaneously. For instance, you can have Tasks that focuses exclusively on different peripherals such as motors, sensors, and the display. As we learned in the lectures, Tasks let us accommodate the available resources better.

Implement Lab 1: Q2-b using Tasks. Here you will display a message on the EV3 display. Define two variables, integer variable lineNumber hold the value of the line number and a string variable message to store the display message.

a) Implement a Task that processes the message to be displayed. This Task should run at 1 Hz (add sleep for 1000 ms in the while loop). Within this Task you will update the two variables as specified in Q2-b.

* 1. b)  Implement a display Task which exclusively handles the EV3 display which runs at 3 Hz. This Task will refer to the variables mentioned above and update the EV3 display accordingly.
  2. c)  In the main Task, implement a debug stream that streams these two variables at 5 Hz.

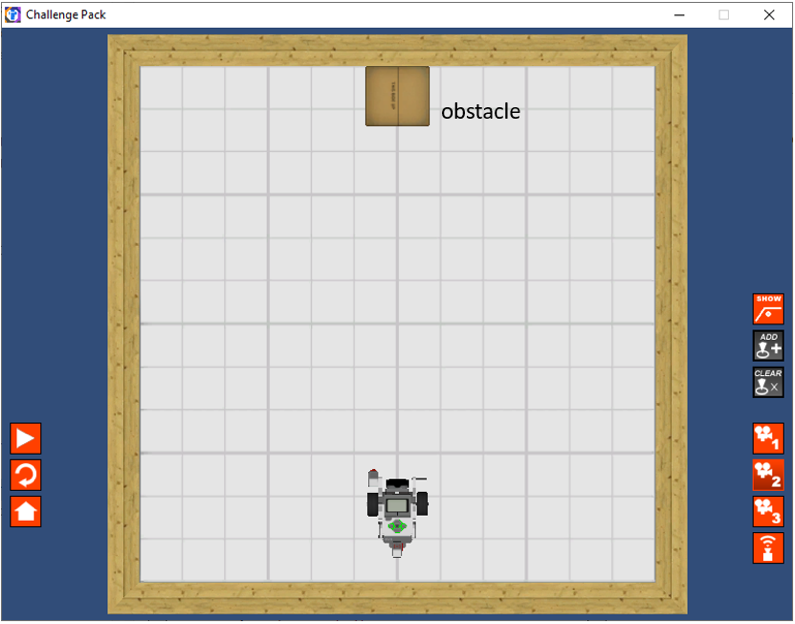
**Note:** The relation between the time delay specified in sleep function and the frequency is: Freq. = 1/tdelay. Recall that display command for the Debug stream is **writeDebugStreamLine(<string>)**

**Deliverables:** Complete code, two screenshots of the LCD showing the updated messages, screenshot of the Debug Stream window.

Q2. Complete the **Forward Until Touch** challenge in the Robot Virtual Worlds under Sensors. In this challenge, you will move the robot forward until the touch sensor is pressed against an obstacle (see Figure 1). Use the getTouchValue() function to know if your robot has contacted the object. You should stop moving the robot forward the moment you detect a collision.

* 1. a)  Develop a pseudocode algorithm to solve this challenge. You should clearly identify what sensors you are reading and how you control the motion of the robot.
  2. b)  Use separate Tasks to interact with the touch sensor (i.e., TouchSensorTask) and the motors (i.e., MotorTask). Use the main Task to implement the algorithm (program flow). Discuss and decide the rate at which these Tasks should run with justification within the code. Note that, you can use higher speeds (less sleep duration, i.e., 10-25 ms) when you are reading values from sensors.
  3. c)  Implement a debug stream that runs at 5 Hz to communicate the state of the program. Decide a helpful messaging structure so that the user can identify the robot’s decision process.

2



*Figure 1: Forward Until Touch*

**Deliverables:** Pseudocode, Complete code, screenshot of the Debug Stream window showing a useful message based on the robot state (different messages when not in touch state and when in touch), screenshot of the Virtual World at the moment when the robot has touched the block.

Q3. *Rethinking Movement:* Gyro sensor measures the angular displacement of the robot and the rate of rotation. The getGyroDegrees() method returns the rotation in degrees since its last count. Therefore, use the method resetGyro() at the beginning of the program so that the rotation is measured from zero from the starting orientation. Note that the return value increases when the robot is turning clockwise and decreases when turning counterclockwise. Motor encoder is another on-robot sensor that records the angular displacement of the wheel motors. See Lab 2 for details on the use of an encoder.

These two on-robot sensors will now be used for accurate robot movements.

* 1. a)  *Task 1 – Turning to a target orientation:* Develop a pseudocode by including the feedback control algorithm discussed in class to provide the motor command when the target orientation is stored in **‘target\_angle’**. In the algorithm, clearly identify what sensors you are reading and how you control the motion of the robot. Similar to the previous questions, implement separate Tasks that exclusively handles the motors and the gyro sensor.
  2. b)  *Task 2 – Moving to a target distance:* Repeat steps a and b for motor encoder sensor while the task is to move a given distance ‘**target\_dist**’.

**Deliverables:** Pseudocode, Complete code, screenshot of the Virtual World at different angle and distance values (keep the ‘red beam’ on in the robot and take the screenshot through Camera 2 view).

3

**Note:**

* + -  An error of ±2 degrees and ±2 mm is acceptable.
  + -  Make sure that you use a reasonable proportional-gain value (k=2 is a fair initial value)

with the controller task running at 50 Hz.

* + -  Use a higher frequency for the main task where debug messages appear and the sensing

tasks where the sensors are being read.

* + -  Use **setMotorSync()** command for operating the motors. Consult the reading from Week

2 to refresh the use of different arguments in this command.