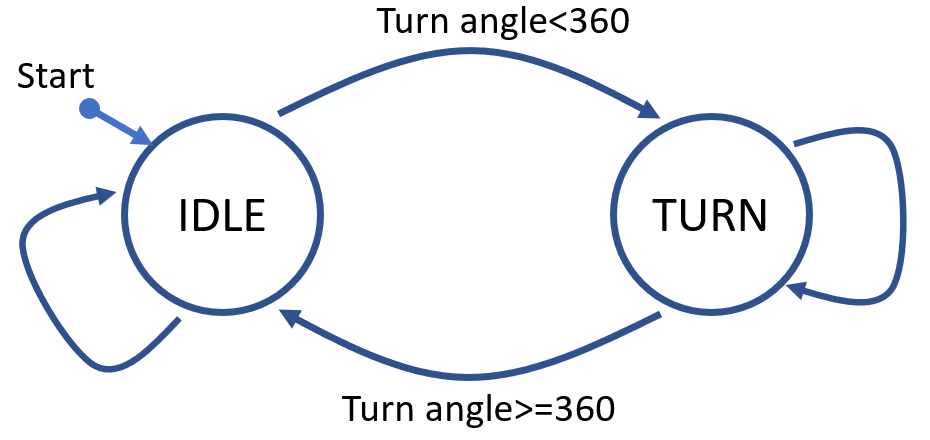
1. Questions (10 + 10 + 30 marks):

Q1. Redo Q4 of Lab 1 (turn the robot to 360, and display time taken on LCD) using a finite state machine-based approach.

* 1. a)  You are provided with an incomplete state diagram (see Figure 1). Complete the state diagram with the conditionals that determine the arrows to self. Include your solutions in the accompanying PDF.
  2. b)  Write the RobotC program to implement the state machine and include it with your submission.

page1image63518784



*Figure 1: Turn 360 degrees clockwise then display the time taken.*

Q2. Similarly, redo Q5 of Lab 1 (move till you have covered 1 m and display the time taken) using finite state machines.

* 1. a)  Complete the state diagram shown in Figure 2. Submit your solutions in the accompanying PDF.
  2. b)  Implement the state machine as a RobotC program and include the code file with your submission.

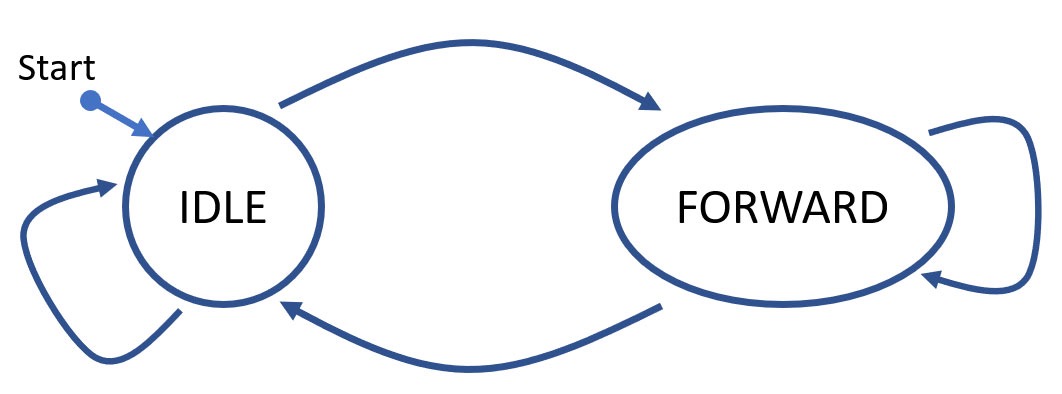
*Figure 2: Move 1 m forward then display the time taken.*

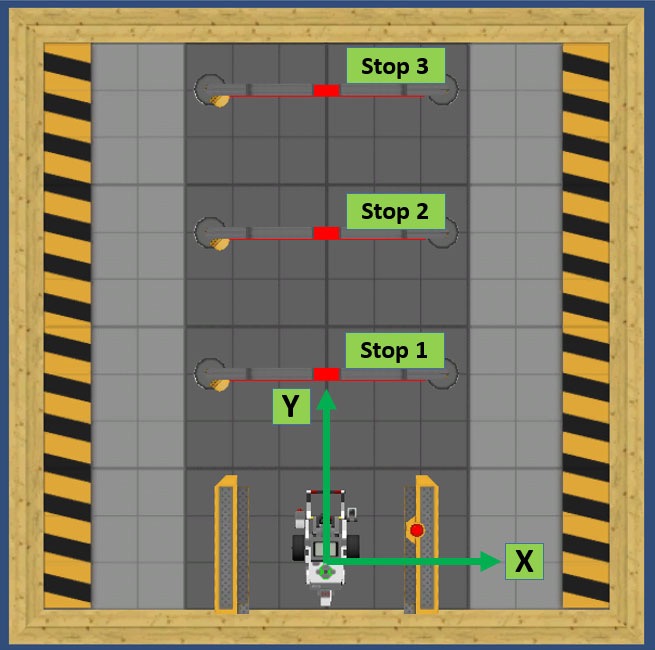
Q3. Use state machines to complete the Sensabot challenge. See the description below.

* 1. a)  Draw a state machine that solves this challenge. Identify the states and the variables. Include the solution in the word file.
  2. b)  Implement the state machine as a RobotC program.

**Overview:** In this challenge, you will program your EV3 robot to move from its starting box to three different lines on a game board, stopping at each one to perform an inspection, represented by raising and lowering the robot’s arm (see Figure 3). When the inspecting all three locations complete, the robot must return to its starting box. The entire robot must be inside the box (no parts over the line). The robot configuration and dimensions are listed in Figure 4.

2





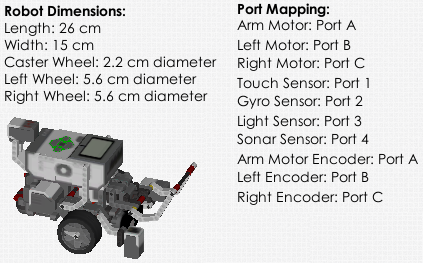
*Figure 3: Sensabot challenge*

**Data Sheet – Virtual Robot – EV3**

*Figure 4: Robot configuration parameters. This is the same as REMBOT configuration.*

* 1. The robot starts inside the starting box and should be with its arm lowered. The robot must move and stop at each line, raising and lowering its arm, representing the inspection process. The arm must be directly over each line when the inspection is performed.
  2. The robot must return to its starting box after completing the inspection process at the third line. The entire robot must inside the box (no parts over the line).
  3. Your program should display on the Remote LCD which state the robot is in during the run. You can determine an appropriate message or labeling structure. These messages should be understandable to a layperson.  
     E.g. Start, End, Stop 1 completed, Arm raised, Arm down, and Moving toward Stop 2, etc.
  4. Also, use debug stream to update which state the robot is in.

3



**Hints:**

* 1. •  Use marks in the virtual world to measure the distances to each line on the board so you know how far you need to move each time.
  2. •  You can use a coordinate system to determine the coordinates of the path. From this coordinates, compute the distance and amount of turn needed to perform to reach to each coordinate in order.
  3. •  Alternatively, you can find the number of centimeters your robot travels in each rotation based in the data from datasheet, and using that to find the number of rotations you need.

Eg. The robot wheel diameter is 5.6 cm, so each turn will cover 𝜋𝑑 length, which is about

17.6 cm.

* 1. •  Similarly, you can also make a test run, then calculate “how many times as far” you need

to move to get to each line, compared to the test run.