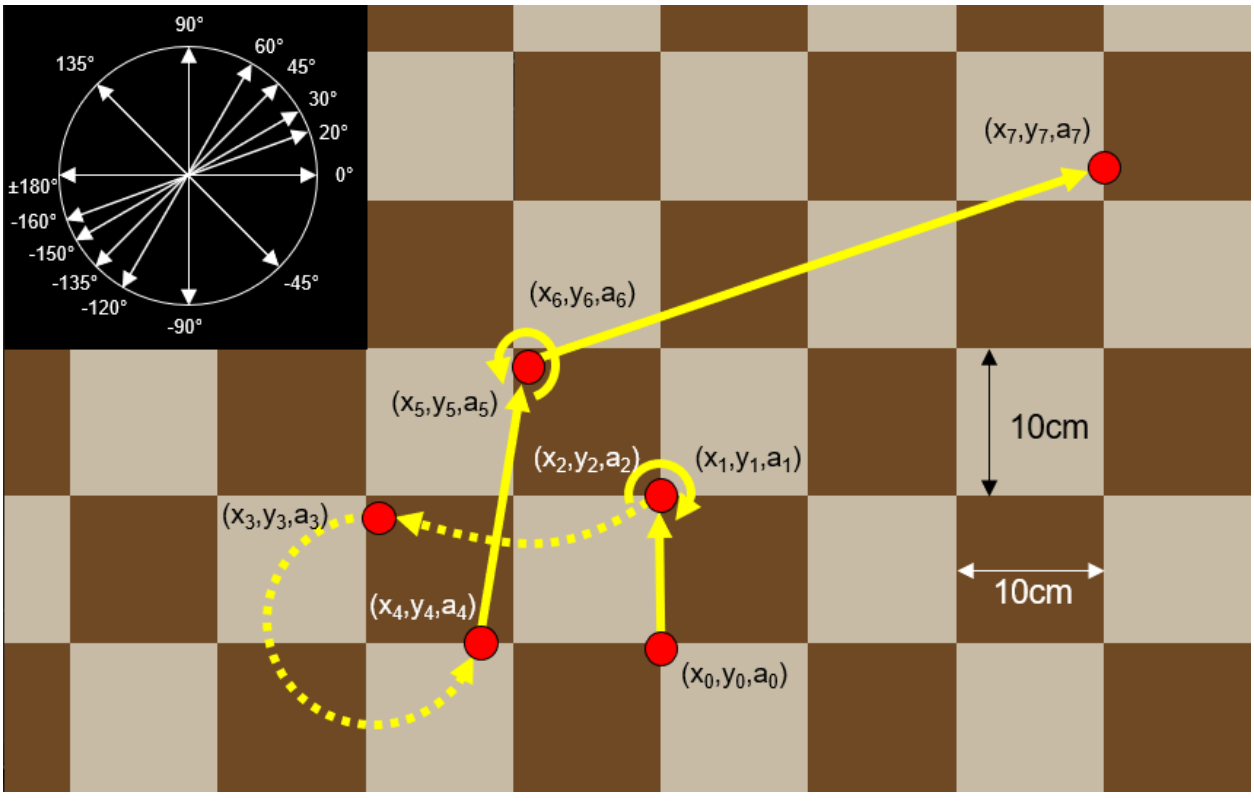
LAB 4 – Position Estimation

(1) Download the **Lab4_PositionEstimation.zip** file and unzip it. Load up the **MeasuredCheckerboard** world. The goal of this lab is to compute the forward kinematics of the e-puck robot as it makes straight movements, spins and curves in the environment.

(2) The **Lab4Controller** code has been started for you. The robot starts at position ($\mathbf{x}_0$,$\mathbf{y}_0$, $\mathbf{a}_0$) = (0, 0, 90°) and is programmed already to move as shown by the yellow path below. It begins with a straight-line movement, followed by a right spin, a left curve, a sharper right curve, a straight path, a left spin and finally a straight line. As it moves, it touches positions ($\mathbf{x}_i$,$\mathbf{y}_i$, $\mathbf{a}_i$) in sequence, where $0 \le i \le 7$. Note that $(x_1,y_1) = (x_2,y_2)$ and $(x_5,y_5) = (x_6,y_6)$ since these are spins … but $a_1 \neq a_2$ and $a_5 \neq a_6$. A compass chart (showing examples of angles in degrees) is shown on the top left for your reference. Each square of the grid has a dimension of 10cm x 10cm.



(3) It is your job to provide the code so that all the values (x_i, y_i, a_i) are displayed for all $0 \le i \le 7$. Each should be printed on a separate line with all values being doubles with one decimal place as follows: (x0, y0, a0) = (0.0, 0.0, 90.0)

For curved movements, you should show the ICC **radius** as well as the θ^{Δ} values (each with one decimal place as well).

> Tip: In JAVA, to print values to 1 decimal place, you can do this:
>
> ```
> System.out.printf("x = %2.1f\n", x);
> ```

In between each move, you must calculate and display the values of (x_i, y_i, a_i) before making the next move. Do NOT change the given code. It has been carefully constructed to move the robot the specified amount of time and updating the position sensors accordingly. You will notice that there is no WHILE loop with SWITCH statements because the robot will not run indefinitely. Instead, it is just a one-time run of the code as an experiment to test the robot positioning formulas.

You should use the **LeftReading** and **RightReading** variables … these represent the number of radians that the left and right wheels have turned, respectively.

Make use of the lecture notes for the necessary formulas. You should also make use of the **WHEEL_RADIUS** and **WHEEL_BASE** constants.

> Tips: Here are the math functions in JAVA. Note that all trigonometry functions require angles in **radians** … NOT in degrees! If you use the wrong unit, your results will be wrong:
>
> ```
> Math.sin(radiansValue)
> Math.cos(radiansValue)
> Math.toRadians(degreeValue)
> Math.toDegrees(radiansValue)
> ```
>
> Also, angles like **-394.2°** look strange. It is better to always display angles within a **-180°** to **+180°** range. You can use IF statements to adjust value for printing:
>
> ```
> IF (angle > 180)  THEN angle = angle - 360°
> IF (angle < -180) THEN angle = angle + 360°
> ```

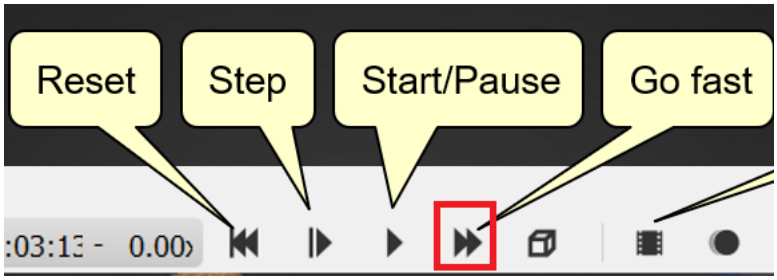
You MUST NOT have any "magic numbers" in your code (except values **180** or **360**) … so please define constants for hard-coded values that you will use.

DO NOT change the MAX_SPEED constant. It appears that the Webots simulator does not calculate the proper robot movements if you change this value. The robot will run slow.

Submit your **Lab4Controller.java** code. Make sure that your name and student number is in the first comment line.

Other Tips:

- Work on each path portion one at a time. Look at your results and make sure that the number make sense according to the 10cm scale and compass chart.
- Make use of the fast forward button to fun fast:



- Don’t forget that if you accidentally mess up the environment or view, it will be reset when you restart. You can zoom in closer and use ALT-1 to adjust for a top-down view if you mess up the viewpoint … but when you restart, it will go back to the default view. You can change the default view so that it goes to that view on a restart, but just make sure that the robot is at its original start position BEFORE you save the world. To do this … make sure that the simulation is not running, press restart to get the robot back to the start, then change your view to how you want it (WITHOUT running the world). Then save the world. Then you can start the simulation with that new starting view.