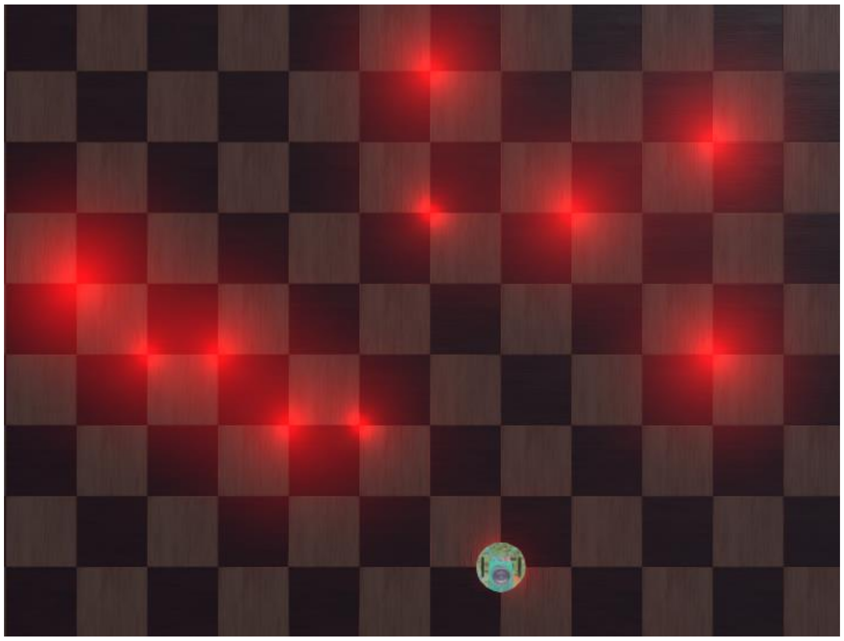
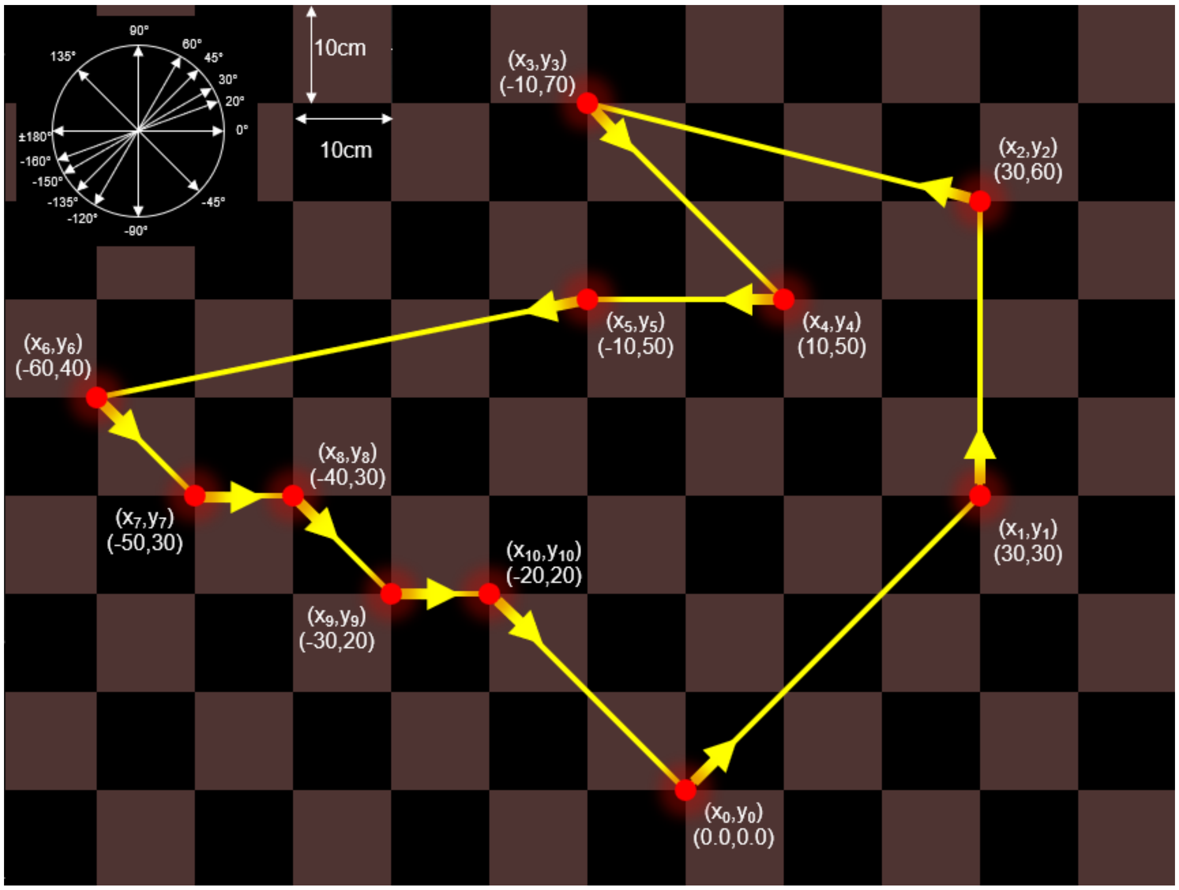
# LAB 5 – Inverse Kinematics

**(1)** Download the **Lab5_InverseKinematics.zip** file and unzip it. Load up the **DarkMeasuredCheckerboard** 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 **Lab5Controller** code has been started for you. The robot starts at position ($\mathbf{x}_0$,$\mathbf{y}_0$, $\mathbf{a}_0$) = (0, 0, 90°). Your goal is to program it to move to a series of points by travelling along a piece-wise linear path as shown below:



Your code MUST make use of the inverse kinematics equations to move from its starting location **($x_0$, $y_0$)** to the remaining points in the order shown and then back to **($x_0$, $y_0$)** again. The robot begins facing upwards but it need not end facing upwards. Your robot should make a series of spins and straight movements. Each spin must turn left or right properly (i.e., choose the shortest of turn … either left or right). Your code MUST make use of the **WHEEL_RADIUS** and **WHEEL_BASE** constants. You MUST NOT have any “magic numbers” in your code (except values **0**, **90**, **180** or **360**). 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.

Code has been written for you in a function called **move()** which makes the robot move until the right wheel has turned **thisManyRadians**. You MUST NOT alter this code. You must call this function either to make the robot go straight or to make it turn. The left and right wheel speed parameters will determine whether the robot is spinning or going straight. You just need to supply the appropriate wheel speeds. When turning, the **thisManyRadians** parameter is the **rightReading** variable that is mentioned on slide 9.

Also, 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.

**Debugging Tip:**

- If your robot does not turn enough (or turns too far), then you have something wrong with your formula. Make sure that you are using radians in your formulas, as opposed to degrees.

- If your robot keeps spinning, you likely forgot to convert from degrees to radians (or vice versa).

Submit your **Lab5Controller.java** code. Make sure that your name and student number is in the first comment line.

More useful tips:

- Get the turning working properly first.  Make sure to use print statements to see if you are calculating the correct number of degrees to turn.
- Make use of the fast forward button to fun fast: