Report lab activity 01

Phototaxis

First thing first, to implement a random walk, I decided to implement a behavior that, when the robot spawn in a random position in the arena, it starts to move from its standing direction at the maximum speed valor (15). Then, during each steps, I find the index, in the light sensors list, with the highest value using a function that search across the list and return the index of the highest value light sensor. After this, every five steps I check if the index found is on the right ($1 < index < 13$) or on the left ($12 < index < 24$) hemisphere of the light sensors circle and I rotate the robot in that direction (using the other wheel as the center of rotation), until the highest sensor index is 1 or 24: in that case, it starts random walking again. I decided to exclude from the range the sensors 24 and 1 due to the fact that, for the strategy that I have chosen, keeping that sensors lead the robot to start zigzagging too much towards the light. By the end, after the robot has reached the light, it keeps moving under the bulb.

Collision Avoidance

In collision avoidance task, the robot needs to random walking while avoid obstacles randomly placed in the arena. The random walk strategy I implemented is the same as for phototaxis. Then, to avoid obstacles, I initially decided to implement a similar behavior as the one I used for the first task but using the proximity sensors. So I find the index of the proximity sensor with the maximum value and, based on the position of the sensor, I rotate the robot, but this time in the opposite direction (because the robot needs to avoid obstacles). However, this strategy does not worked due to the fact that, when the index is 1 or 24 the robot got straight to the obstacle without avoiding it. Also, including those indexes leads the robot to start spinning around. The strategy I chosen by the end was not to compute the maximum index, but to compute the sum of all sensors in a range: index from 1 to 8 in right hemisphere, and index from 16 to 24 in the left hemisphere. After computing those values, if both (left and right) are equals to zero, the robot keeps walking randomly. Otherwise, I check which value is greater and then I rotate the robot in the opposite direction. Also, while computing those values, I use a threshold for the sensors to exclude from the sum all the proximity sensor's values that are lower than the threshold. I do this due to the fact that during some tests, I found out a scenario where the robot gets stuck between two very close obstacles, even if the space between them was enough to pass through.