E-Puck lab #1 (IR)

Overview

In the previous lab, you learned how to compile your code and run it on the robots. You saw how to make the robot move, how to read the values of the proximity sensors, and how to send simple messages back to the computer (useful for debugging). In this lab, you will implement a simple obstacle avoidance algorithm. The goal is to make the robot move forward unless it detects an obstacle too close in front of it. You can decide what “too close” means. If there is an obstacle, it should turn left until the space in front is clear and then move forward. At all times, the robot should indicate the direction to the nearest obstacle by turning on the LED in that direction.

Your algorithm could be structured like this:

```
while true
      read all proximity sensors and store their values and directions
      find the closest proximity reading
      turn off all red LEDs
      turn on the LED facing in the direction of the closest reading
      if proximity sensor #0 or #7 read less than 2 cm
            turn left
      else
            move forward
```

Proximity sensor #0 is directly to the right of the camera (to the left from the camera's point of view). The numbers increase clockwise around the robot, so that #7 is the one just on the other side of the camera.

How to Get Data from the Proximity Sensors

The IR proximity sensors work by sending out a flash of infrared light while simultaneously measuring the intensity of the reflected light. The higher the intensity, the closer the object it's reflecting off. Of course, the measured intensity will also include ambient IR light and other effects which must be calibrated out.

- e_init_prox() - This initializes the proximity sensors. Call it once somewhere at the top of your main function.
- e_get_prox(int sensor_number) – This returns the intensity of the reflection. It is just a raw intensity, so you will need to calibrate it if you want to measure absolute distances. To calibrate the sensors, place objects (of the type you expect to be your obstacles) at known distances and record the raw measurements. Fit a curve and write a function like give_calibrated_proximity(int raw_proximity) to use this curve to give you the calibrated values. In any case, remember that as an object gets closer, the intensity of the reflection will go up, so larger values from e_get_prox() indicate closer objects.

The figure below is an example of measured sensor points and a fitted curve done by a team of students in 2012. Similar trends can be found for all proximity sensors.