E-puck Lab 2 – Making Sense of It All

In this lab we will experiment with the sensory capabilities of the e-puck robot. Our control programs will take sensory readings as input and produce motor actions in response. This type of control is called *reactive control*.

E-Puck Sensory Interface Commands

***robot.*getLight()**
Returns a list of sensor readings from the light sensors. Sensor readings range from 0 (maximum light) to 4000 (no light).

***robot.*getLight(*sensor_number*)**
Returns a single sensor reading from the specified light sensor. Sensor readings range from 0 (maximum light) to 4000 (no light). Sensors are numbered clockwise from 0 to 7, starting from the front of the robot.

***robot.*getProximity()**
Returns a list of sensor readings from the proximity sensors. Sensor readings range from 0 (no object in range) to 4000 (object at closest range).

***robot.*getProximity(*sensor_number*)**
Returns a single sensor reading from the specified proximity sensor. Sensor readings range from 0 (no object in range) to 4000 (object at closest range). Sensors are numbered clockwise from 0 to 7, starting from the front of the robot. **getDistance** and **getIR** are synonyms for **getProximity**.

***robot.*getIRAngle(*sensor_number*)**
Returns the angle offset in degrees of the specified IR (light/proximity) sensor on the robot's body, measured clockwise from the front of the robot. Sensors are numbered clockwise from 0 to 7, starting from the front.

***robot.*getAmbientLight()**
Returns the average of all light sensor readings. Sensor readings range from 0 (maximum light) to 4000 (no light).

***robot.*getAccel()**
Returns a list of the the accelerometer readings as [*x, y, z*].

***robot.*getSound()**
Returns a list of the microphone readings as [*front_right*, *front_left*, *rear*].

***robot.*getWheels()**
Returns a list of the left and right wheel encoder values as [*left_wheel*, *right_wheel*]. Encoder values go from 0 to 65535 then wrap around to 0 again.

***robot.*resetWheels()**
Resets the wheel encoders to 0.

***robot.*getSelector()**
Returns the current position of the selector switch on the robot (0-15).

***robot.*getLEDnumber(*sensor_number*)**
Returns the LED number of the LED closest to the specified IR sensor on the robot's body. Sensors and LEDs are numbered clockwise from 0 to 7, starting from the front.

Programming Exercises

1. Here is a program that displays the values of the robot's front-left and front-right light sensors:

   ```
   def seeklight():
       try:
           while True:
               left = e.getLight(6)
               right = e.getLight(1)
               print "left = %.2f, right = %.2f" % (left, right)
               wait(0.1)
       except KeyboardInterrupt:
           pass
   ```

   Modify this code to make the robot rotate toward a light source, without moving forwards or backwards. The robot should stop rotating when the light source is directly in front of it. You can do this by comparing the left and right sensor readings. If they differ by an amount greater than some threshold, the robot should respond appropriately, otherwise it should not move. You will need to experiment with different thresholds to fine-tune the robot's response. It may also help to normalize the light readings to compensate for the surrounding ambient light in the environment. Put the following lines at the beginning of your file to set the ambient light level automatically each time the code is loaded:

   ```
   ambient = e.getAmbientLight()
   print "Ambient light level is", ambient
   ```

2. Generalize your light-seeking program to respond to all of the robot's sensors, not just sensors 1 and 6. Your robot should turn smoothly toward the light no matter which direction it is coming from. The command getLEDnumber(*sensor_number*) may be useful for debugging. For example, you could use it to turn on the LED corresponding to the most active light sensor at any given moment, in order to better understand what is happening.

3. Now add forward motion to your program, so that the robot will actively follow a light source. If it does not detect any light, it should stop.

4. Implement an obstacle-avoiding program that repeatedly reads the robot's distance sensors and moves forward (or in a random direction) unless there is something in the way. If the robot detects an obstacle, it should turn away from it in an appropriate manner.