E-puck Lab 1 – Get Moving!

The e-puck robot was developed at the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland, and is equipped with a low-resolution color camera, IR/light sensors, microphones, a speaker, an accelerometer, and various LEDs.

Setting Up

Go to our class web page under *Links* and click on *E-puck robot*. Follow the instructions to set up a connection to your robot, and name your Python file **robot.py**.

Simple Commands

Try out each of the following movement commands with your robot. IMPORTANT: Have your lab partner make sure that the robot doesn't accidentally drive off the table!

```
e.rotate(speed)
e.stop()
e.rotate(speed, seconds)
e.forward(speed)
e.forward(speed, seconds)
e.backward(speed)
e.backward(speed, seconds)
e.turnLeft(speed)
e.turnLeft(speed, seconds)
e.turnRight(speed)
e.turnRight(speed, seconds)
```

where ***speed*** is a number between 0 and 1 and ***seconds*** is an optional parameter that specifies how long the robot should continue to move at the given speed.

You can also produce a curved trajectory by combining movement with rotation using these commands:

```
e.move(forwardSpeed, rotationSpeed)
e.move(forwardSpeed, rotationSpeed, seconds)
```

The speed values for the `move` and `rotate` commands can range from –1 to +1.

More Complex Behaviors

1. Create a sequence of commands that will cause the robot to trace a square. Hint: use the commands `forward` and `turnLeft` with appropriate speed and time values. You'll have to experiment with your robot to find values that work well.

2. Next, create a sequence of commands that will make the robot trace a *larger* square than in exercise 1.

3. Create a sequence of commands that will make the robot trace a *smaller* square than in exercise 1.

4. Create a sequence of commands that will make the robot trace a figure-eight.

Creating New Commands

5. We can create a new compound command from one or more individual commands by putting them together into a single Python function. For example, we could define a new function called `spin()` that rotates the robot to the right at full speed for three seconds, and then prints out a message, as shown below:

```
def spin():
    e.turnRight(1, 3)
    print "Woohoo! I'm getting dizzy"
```

Add this code to the file **robot.py**, save it, and then type `run robot` in IPython to reload the file. Then type `spin()` to try it out.

6. Define a function called `square()` that executes your sequence of commands from exercise 1.

7. Define a function called `bigSquare()` that executes your commands from exercise 2.

8. Define a function called `smallSquare()` that executes your sequence of commands from exercise 3.

9. Define a function called `yoyo()` that makes the robot move back and forth a few times like a yo-yo.

10. Define a function called `wiggle()` that makes the robot wiggle left and right.

Parameterizing Commands

11. Modify the definition of `square` so that it takes an input parameter called ***length***, which specifies the number of seconds that the robot should move forward when tracing each side of the square. By calling `square` with different values for ***length***, the robot will draw squares of different sizes.

12. Modify your `yoyo` definition so that you can tell it how fast to move the robot back and forth.

13. Create a new function called `yoyo2` that takes two input parameters: the speed of movement, and the number of seconds to move in each direction.

Building Blocks

14. Create a new function called `dance()` that uses your parameterized functions from above as building blocks to make the robot do some type of dance. You can also make the robot temporarily pause for a specified number of seconds using the **`wait(`*`seconds`*`)`** command. This can be useful when designing the "choreography".

Lights, Camera, Sound, Action!

Experiment with the following commands available for using the robot's camera, lights, and sound effects. This command plays one of the robot's five built-in sounds:

`e.playSound(`*`number`*`)` *use a number from 1 to 5*

The commands below turn on the robot's LEDs, which are numbered from 0 to 7 going clockwise around the robot from the front. The parameter value **`"all"`** turns on all of the red LEDs along the edge of the robot, **`"front"`** turns on the front red spotlight LED, and **`"body"`** turns on the robot's green body LED.

`e.onLED(`*`number`*`)`	*use a number from 0 to 7*
`e.onLED("all")`	*quote marks required*
`e.onLED("front")`	*quote marks required*
`e.onLED("body")`	*quote marks required*

These commands turn off the LEDs:

e.offLED(*number*)	*use a number from 0 to 7*
e.offLED("all")	*quote marks required*
e.offLED("front")	*quote marks required*
e.offLED("body")	*quote marks required*

These commands cause the LEDs to flash on and off:

e.flashLED(*number*)	*use a number from 0 to 7*
e.flashLED("all")	*quote marks required*
e.flashLED("front")	*quote marks required*
e.flashLED("body")	*quote marks required*

These commands turn all of the LEDs on or off in a cycle:

```
e.onCycleLED()
e.offCycleLED()
e.flashCycleLED()
```

This command causes the robot to take a picture using its camera and then display it in a pop-up window:

```
e.takePicture().show()
```

After experimenting with the above commands, use them to enhance your robot's `dance()` behavior with light and sound effects.

Repeating Commands

The following looping constructs can be used to execute a block of commands over and over. The first form repeats the block of commands a specified number of times. The second form executes the block for the given amount of time in seconds. The indentation of the commands in the block must be exactly the same for all commands.

```
for count in range(number):
    command1
    command2
   ...etc....


while timeRemaining(seconds):
    command1
    command2
   ...etc....
```