Gürtler Fabienne 12-935-508

Series 2

Exercise 2a)

Step 1

```
# forward
emit forward SPEED_NORM
for i in 0:DELAY do end # wait before stopping movement
emit stop
```

The moving e-pucks moves forward with both wheels set to SPEED_NORM = 400. After emitting command forward it waits for DELAY = 30000 until stopping movement by emitting command to stop. I've decided to keep the duration of every movement the same.

Step 2

```
# step 2
emit curve [SPEED_NORM, SPEED_NORM/2 -cal1]
```

Both e-pucks drive a curve, one to the left the other to the right. They should finish with a 90° resp. -90° angle to their starting position. Since the time available for the movement is constant, to get the desired angle is achieved by adjusting speed and ratio of the e-pucks. In this case the settings are as following: the faster wheel turns with SPEED_NORM while the slower turns with SPEED_NORM/2. A ratio of ½ does not yet quite give the right angle so a variable *cal1* = 05 was subtracted from slower wheel. An effort was made to keep the code as generally applicable as possible. Thus I tried avoiding hard-coding the speeds. If SPEED_NORM changes it suffices to adjust *cal1* to get the correct angle.

Step 3

```
#e-puck giving command      |   #e-puck receiving signal
# step 3                    |   onevent backward
emit backward SPEED_NORM    |   speed.left = -args[0]
                            |   speed.right = -args[0]
```

Driving backwards is almost like driving forward except the 2 e-puck's receiving the signals react to this command by setting speed.left and speed.right to -SPEED_NORM.

Step 4

```
# 180-degree turn
emit curve [SPEED_NORM/2 +cal2, -SPEED_NORM/2 -cal2]
```

After driving the same curve as in step 2, the e-puck's perform a 180° turn so they end up in the same position as when starting the triangle. The principle of finding the correct speed to get a U-turn is similar to what is described in step 2. By setting one wheel to the negative value of the other wheel the e-puck's turn on the spot. Again, by experimenting, the variable *cal2* has to be calibrated correctly to make a complete turn.

Step 5

```
for j in 0:2 do
    emit curve [SPEED_NORM, SPEED_NORM/2 -cal1][...]
end
```

To repeat the triangle 3 times a loop is put around steps 2 to 4.

It is very difficult to get exactly the required angles, my values are only an approximation. In both simulation and reality I was only able the get the robots to return to roughly their starting position.
It is however possibly to use the U-turn to compensate these imperfect 90° curves a little. If, for example, the e-puck makes a little less than a 90° curve and the U-turn is a little over 180° it evens things out a little. Also by repeating steps 2-4 three times, this impreciseness gets multiplied each time a triangle is completed.
If α is error in 90° curve and β error of U-turn, error for each cycle is : $2\alpha + \beta$, for every $\alpha, \beta \in \mathbb{Z}$.
If triangle is completed three times the e-puck ends up with an total impreciseness of $6\alpha + 3\beta$.

Exercise 2b)
Like in Series 1 the angle of the real e-puck's turn is smaller than in the simulation. The cause of this difference in behavior could be either the time or speed. Maybe time passes differently in the simulation as opposed to reality. Or the actual robot just generally moves slower, possibly due to roll friction and other factors not incorporated in the simualtion. As such it was necessary to adjust *cal1* and *cal2* quite a bit to get the same pattern. It can also be noted that this recalibration is not linear, where *cal1* is multiplied with 10, *cal2* is multiplied with 7. So one can not simply multiply all values in the simulation with one constant to get the correct behavior in reality.

var	Simulation	Reality
cal1	05	50
cal2	10	70

Table 1: Comparison simulation <-> reality