Requirements

Extend the req-step routine by measuring the elapsed time from the start of the routine to its end.

Requirement analysis

I assume the time must be printed in those circumstances:

- the step routine ends as expected after the timer fires
- a ***stop*** command is received
- the robot encounters an obstacle and stops

```
System robotchrono

Dispatch cmd     : cmd(X)   // req-cmd
Dispatch step    : step(T)  // req-step
Dispatch stop    : stop(X)  // req-stop
Event  obstacle : obstacle(DISTANCE)

Context ctxMind ip [ host= "localhost"   port= 8023 ]
Context ctxBasicRobot  ip [ host= "localhost"   port= 8020 ]
ExternalQActor basicrobot context ctxBasicRobot

QActor robotmind context stepper {
    ["var StepTime = 0L;"]

    State s0 initial {
        println("init")
    }
    Goto idle

    State idle {
        println("idle")
    }
    Transition tWork
        whenMsg step -> sStep
        whenMsg cmd -> sCmd

    ... // req-cmd

    State sStep {
        println("sStep")
        onMsg (step : step(T)){
            ["StepTime = payloadArg(0).toLong()"]
            forward basicrobot -m cmd : cmd(w)
        }
        // start chronometer
    }
    Transition tStop
        whenTimeVar StepTime -> sEndStep
        whenMsg stop -> sEndStep
        whenEvent obstacle -> sEndStep

    State sEndStep {
        println("sEndStep")
        forward basicrobot -m cmd : cmd(h)
        // measure elapsed time
    }
    Goto idle
}
```

Problem analysis

The robot must be able to ***proactively*** start a chronometer when entering a step routine, as well as stopping it to read the elapsed time value.

```
System robotchrono

```

```
Dispatch cmd     : cmd(X)   // req-cmd
Dispatch step    : step(T)  // req-step
Dispatch stop    : stop(X)  // req-stop
Event  obstacle : obstacle(DISTANCE)

Context ctxMind ip [ host= "localhost"   port= 8023 ]
Context ctxBasicRobot  ip [ host= "localhost"   port= 8020 ]
ExternalQActor basicrobot context ctxBasicRobot

QActor robotmind context stepper {
    ["var StepTime = 0L;"]
    ["var Start = 0L;"]

    State s0 initial {
        println("init")
    }
    Goto idle

    State idle {
        println("idle")
    }
    Transition tWork
        whenMsg step -> sStep
        whenMsg cmd -> sCmd

    ... // req-cmd

    State sStep {
        println("sStep")
        onMsg (step : step(T)){
            ["StepTime = payloadArg(0).toLong();"]
            ["Start = System.currentTimeMillis();"] // start the chronometer
            forward basicrobot -m cmd : cmd(w)
        }
    }
    Transition tStop
        whenTimeVar StepTime -> sEndStep
        whenMsg stop -> sEndStep
        whenEvent obstacle -> sEndStep

    State sEndStep {
        println("sEndStep")
        forward basicrobot -m cmd : cmd(h)
        ["val Elapsed = System.currentTimeMillis() - Start;"] // stop the chronometer
        println("elapsed time: ${Elapsed}")
    }
    Goto idle
}
```

Deployment

Build both BasicRobot and RobotMind into deployable zip files with the commands: gradle -b build_ctxMind.gradle distZip gradle -b build_ctxBasicRobot.gradle distZip Copy any *.pl file into the bin sub directory and execute the executable scripts.

By Michele Proverbio email: michele.proverbio@studio.unibo.it