**Project Report**

Daniele Mammone (10625264) - Gianmarco Naro (10610374)

Introduction

The aim of the project is to implement a network of Smart Lights, composed by Light Bulbs and a Controller. Both the type of devices run the real-time OS TinyOs. In a real context, the Controller and the Bulbs should be constructed on top of devices with different HW, that should be more performant on the controller, and less powerful on bulbs. For simplicity, we implements controller and bulbs on the same Sky Mote device. In doing this, controller never turns on/off his legs, while bulbs never sends command messages to other bulbs, but for redirecting.

Controller Implementation

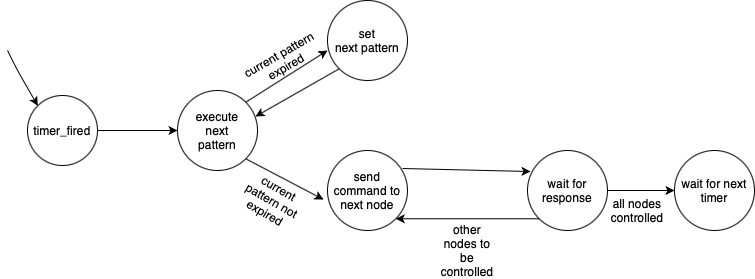
The controller is implemented with three different luminous patterns, that cyclically changes after a certain number of iterations (defined in the code).

The controller have three variables, controlling the logic of the luminous patterns:

* the pattern currently running
* the iteration of the current pattern
* if the lights are currently on or off

At each iteration,

* if the lights are on, the controller turns off all the lights
* otherwise, if the lights are off,
  + if the maximum number of iterations per pattern is not expired, starts the next iteration of the current pattern.
  + otherwise, sets the next pattern and starts the next iteration of the current pattern.



For each iteration of a certain pattern, the controller sends a message to each smart bulb, with the command to turn off, or turn on the lights. tinyOS has some limitations in message sending and receiving, such as

* loss of dynamic memory allocation
* multiple message sending
* multiple message receiving

there isn’t the possibility to send multiple messages at once, and we must take care that a node doesn’t receive a message while processing another one (possible situation since bulbs must take care of messages directed to themselves and messages directed to other bulbs), we implemented message sending as follows:

* a message is sent to a certain bulb
* the controller waits for the bulb to send a confirmation message of command processing
* when the confirmation message is sent, the controller calls a routine to send the next command message of the current pattern
* if other nodes must be processed, a message is sent, otherwise the next time expiration is waited.

Since by requirements lights must be turn off between iterations of the same pattern, there is a boolean variable that takes trace if lights are currently on or off. If they are currently on, the code of the “toggle” pattern (later explained) is used in order to turn off all the lights.

Nodes implementation

The only tasks of the nodes are to turn off/on lights and send a confirmation message to the controller, and to redirect messages to other nodes, when receiving ones not directed to them.

In the receive procedure, there are two cases:

* a confirmation message is received: through the sendConfirmation routine, is redirected to the controller
* a command message is received: if the message is for the current node, it’s processed, otherwise redirected to the correct node.

We used the same addressing and routing suggestions of the requirements: the controller is the 1 node, while the others are in the same tree strutted proposed in the requirements.