Project Report

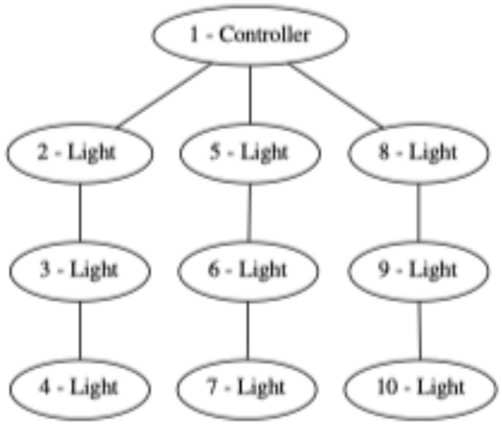
Daniele Mammone (10625264) - Gianmarco Naro (10610374)

Introduction

The aim of the project is to implement a Smart Network of Light Bulbs. This network is composed by a controller, and by smart bulbs. Even if both controller and bulbs have the same hardware running tinyOS (Sky Mote type), the two categories of devices run different firmware. This allows to make the binary lighter for the bulbs and to leave only to the controller the light controlling logic.

Addressing

The addressing used is the same proposed by the requirements document. Here the scheme just for reference.



Routing

In our routing strategy, two cases must be taken in consideration:

1) *Command messages*: when N receives a packet for X, with N $\neq$ X, if X $>$ N the packet is redirected to X + 1, otherwise to X – 1. We didn't take care of the case in which a leaf node, such as 4, receives a message for a node with higher address, such as 5, since this situation is impossible: the controller knows that every message for 2, 3, 4 must be routed through 2, every for 5, 6, 7 through 5 and ones for 8, 9, 10 through 8.
2) *Confirmation messages*: they are sent by bulbs to the controller. If the node is not a child of the controller, routes the message though the father node; else, it is routed directly to 1.

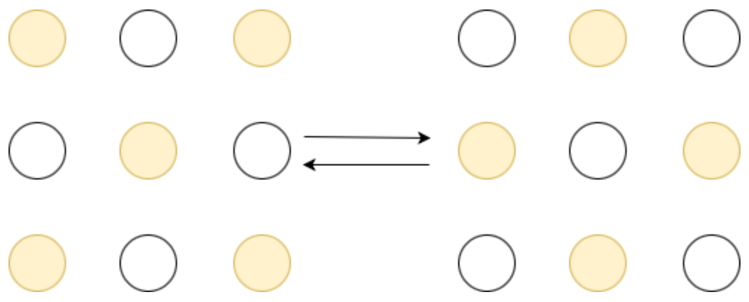
Bulbs Working

Bulbs cannot do anything without receiving a message. When a message is received, it checks the type of message:

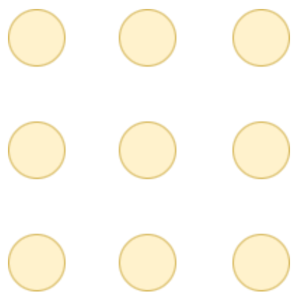
- *Command message*: If the message is for the node that received the packet, the led is set to the value contained in the message and a confirmation is sent to the controller, according to the previously described routing strategy. Otherwise, if the message is for another bulb, it is redirected to the destination node, also with the routing strategy described before.
- *Confirmation message*: it is surely a confirmation from another node. It must be redirected to the controller, using the routing mechanism described in the previous paragraph.

Controller Working

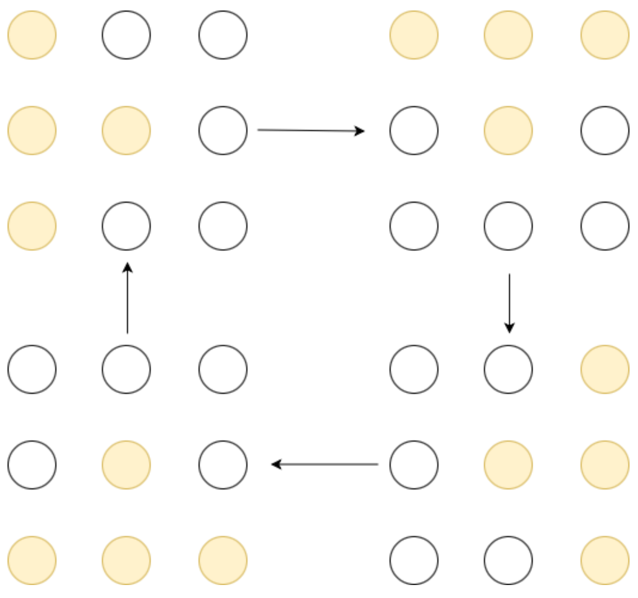
Controller implements three different patterns:



Cross Switch Pattern



All LEDS On Pattern



Triangolar Switch Pattern

For brevity, situations with LEDS off are omitted and the sequence can be changed due to the number of maximum iterations. Indeed, the variable of the previous pattern is shared between triangle and cross switch, to make the code lighter.

The controller operates as follow:

1) A timer is started when radio is initialized. Each time the timer is fired, the nextPattern routine is called, and two things happen, since each pattern run for a fixed number of iteration:
    - If the maximum number of iterations of the same pattern is reached, the pattern is changed, and the routine nextPattern is recalled.
    - Else:
        - If all the LEDS are on, the controller sends a message to all the nodes to turn off the LEDS.
        - Else, calls a routine to start the next iteration of the current pattern.
2) When a routine to start the iteration of the next pattern is called, some operations are done
    a. The current node to be processed is reinitialized to 2, and the routine to process next bulb is called.
    b. If all the bulbs have been processed, the routine ends. The controller waits for the next timer fire to process the next iteration.
    c. Otherwise, the single node is processed, and
        i. If the LEDS must be turn on, a command message is sent, and the controller waits for a confirmation of led processing from the bulb. Then, once the latter is received, the routine for processing next led is called.
        ii. Otherwise, no messages are sent (since all lights are turn off before turning on other LEDS) and the routine is recalled to process the next led.

Each pattern has its own routines to set up the next iteration of the specific pattern, and to process the next bulb.
In all the message sending case, an ACK is requested, and the message is resent if the ACK is not received.
The mechanism of confirmation messages is necessary due how tinyOS manages messages. In fact if a node is processing a message, and another one arrived, it’s discarded. This provoked in our simulation a random behavior: in fact, with some random seeds, certain messages were not processed, even if acked.