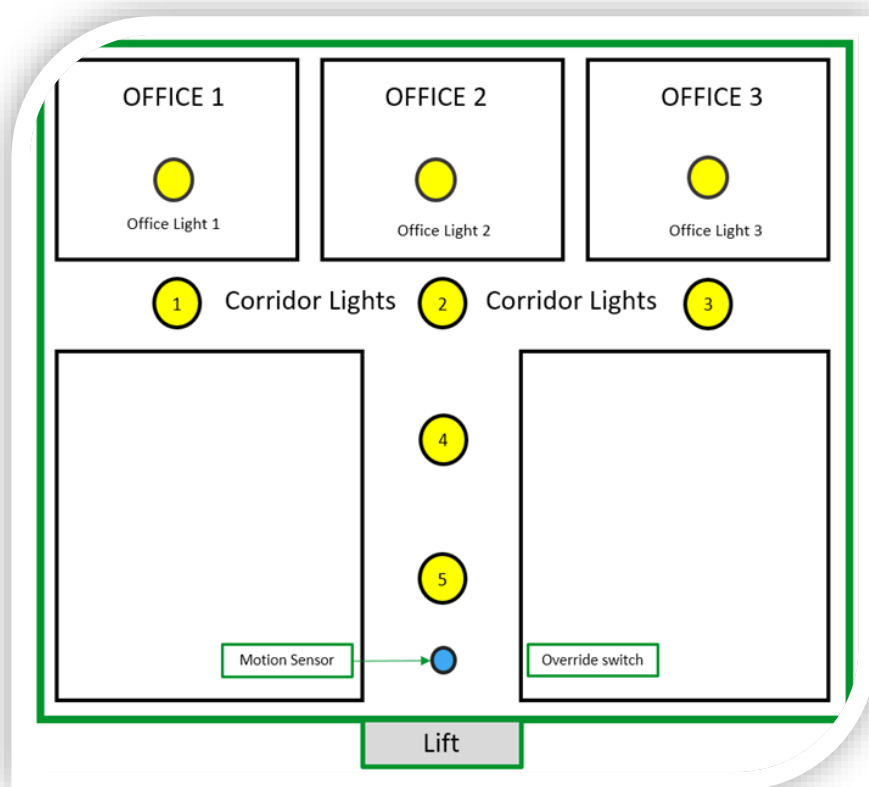


Application note

spaceLYnk Office Building Functions

Configuration Guide - Corridor Linking using LUA scripts



Safety Information

Important Information



Read these instructions carefully before trying to install, configure, or operate this software. The following special messages may appear throughout this bulletin or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of either symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, can result in death or serious injury.

CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury. The safety alert symbol shall not be used with this signal word.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.



Note - You will find additional information here to make your work easier.

Safety Precautions

 WARNING
HAZARD OF INCORRECT INFORMATION • Do not incorrectly configure the software, as this can lead to incorrect reports and/or data results. • Do not base your maintenance or service actions solely on messages and information displayed by the software. • Do not rely solely on software messages and reports to determine if the system is functioning correctly or meeting all applicable standards and requirements. • Consider the implications of unanticipated transmission delays or failures of communications links. Failure to follow these instructions can result in death, serious injury, or equipment damage.

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2014 Schneider Electric. All rights reserved

Table of Contents

- 1 Introduction 6
- 2 Design 6
- 3 Configuration 10
 - 3.1 Assigned objects 10
 - 3.1.1 Monitoring 11
 - 3.1.2 Start of business..... 12
 - 3.1.3 Close of business 13
- 4 Conclusion..... 15
- 5 Appendix 16

1 Introduction

spaceLYnk can be used in the functionality of a typical commercial office building.

A glossary is available in the appendix chapter of this document. Please refer to it whenever necessary.

Competencies

There are a few necessary competencies to follow procedures mentioned in this configuration guide. It is mandatory to have knowledge of KNX commissioning via ETS basic spaceLYnk configuration knowledge. This configuration guide also provides Lua script examples, therefore, familiarity with the programming language Lua script is also recommended.

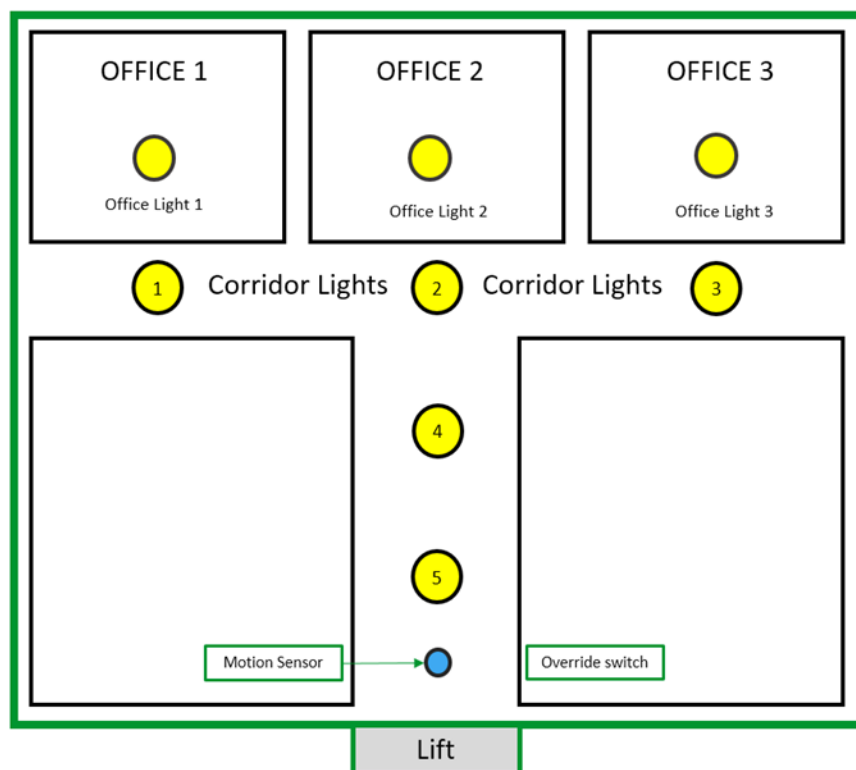
System prerequisites

Software	Version	Download
spaceLYnk	2.1 and newer	http://www.schneider-electric.com

Table 1: software versions of used software

2 Design

The spaceLYnk can provide control over both the corridor lights and office lights for a level of an office building, represented by the diagram below.

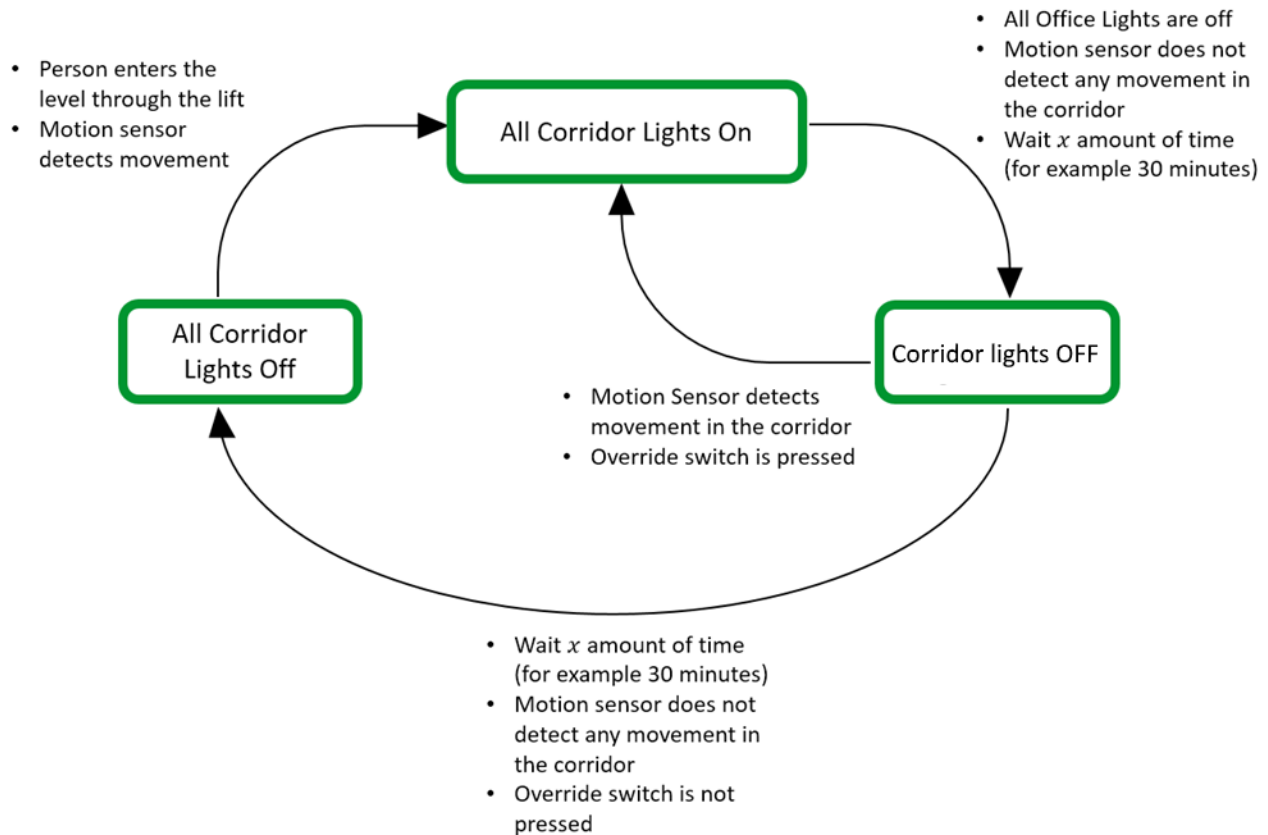


Picture 1: sample floor plan

In a typical office building it is expected that people will enter and exit their office throughout the day, using the corridor to access the lift. At the start of business, the expectation is that all the office lights and corridor lights are off. Consider the first person entering the floor through the lift as they arrive for work. A motion sensor at the entrance to the lift should detect this person's arrival and instantly turn on the corridor lights in response, correspondingly the corridor lights should automatically turn off after everyone has left the building at the end of the day.

This configuration guide shall explore the following office situation:

1. All Corridor and Office lights are off
2. When the motion sensor detects any movement, it shall turn on the corridor lights
3. If all the office lights are off and the motion sensor cannot detect any movement, then it shall switch off lights after a specified time period.
4. Whilst the corridor lights are dimmed, if the motion sensor detects any movement or the override switch is pressed then the corridor lights shall adjust back on.
5. If no movement is detected and the override switch is not pressed, then the corridor lights shall automatically turn off after the same specified time period as before.



Picture 2: Corridor lights logic diagram

Initially, the spaceLYnk shall monitor the state of the following:

1. Office Lights (1, 2 and 3)
2. Corridor Lights (1, 2, 3, 4 and 5)
3. Presence Sensor – presence detected
4. Override switch

To achieve this, create objects to represent each of these items individually.

Don't forget to create status feedback objects for lights.



The following group addresses have been used for this specific configuration, the user can assign any group addresses to these objects as they see fit.

During creation of these KNX objects, assign the keywords as displayed in the table. A residential script shall be used for this example; the repetition of this script simulates the constant monitoring application and updates the KNX objects lighting levels and states as conditions change. The residential script should be written with a sleep interval of 1 second.

3 Configuration

3.1 Assigned objects

Group Address	KNX Object	Tag
2/1/1	Office Light 1	Office
2/1/2	Office Light 2	Office
2/1/3	Office Light 3	Office
2/1/4	Corridor Light 1	Corridor
2/1/5	Corridor Light 2	Corridor
2/1/6	Corridor Light 3	Corridor
2/1/7	Corridor Light 4	Corridor
2/1/8	Corridor Light 5	Corridor
2/2/1	Office Light 2 feedback	Feedback office
2/2/2	Office Light 2 feedback	Feedback office
2/2/3	Office Light 3 feedback	Feedback office
2/2/4	Corridor Light 1 feedback	Feedback corridor
2/2/5	Corridor Light 2 feedback	Feedback corridor
2/2/6	Corridor Light 3 feedback	Feedback corridor
2/2/7	Corridor Light 4 feedback	Feedback corridor
2/2/8	Corridor Light 5 feedback	Feedback corridor
2/3/1	Motion Sensor	No tag for this object
2/3/2	Light Override	No tag for this object



The following group addresses have been used for this configuration, the user can assign any group addresses to these KNX objects as they see fit.

During creation of these KNX objects, assign the tags as displayed in the table. A residential script shall be used for this example; the repetition of this script simulates the constant monitoring application and updates the KNX object lighting levels and states as conditions change. The residential script should be written with a sleep interval of 1 second.

3.1.1 Monitoring

While the corridor lights and office lights should all be switched off, the motion sensor continually monitors the corridor to detect anyone who enters or leaves through the lift, such as employees or cleaners. As previously stated, the spaceLYnk shall need to monitor the states of these KNX objects constantly, using this code shown below.

```
-----Motion sensor-----
sensor_state = grp.getvalue('Motion sensor')
--false if there is no one in the corridor
--true if there are people present

-----Return the state of each of the corridor lights-----
Corridor_1 = grp.getvalue('Corridor Light 1 feedback')
Corridor_2 = grp.getvalue('Corridor Light 2 feedback')
Corridor_3 = grp.getvalue('Corridor Light 3 feedback')
Corridor_4 = grp.getvalue('Corridor Light 4 feedback')
Corridor_5 = grp.getvalue('Corridor Light 5 feedback')

-----Return the state of each of the office lights-----
Office_1 = grp.getvalue('Office Light 1 feedback')
Office_2 = grp.getvalue('Office Light 2 feedback')
Office_3 = grp.getvalue('Office Light 3 feedback')

spaceLYnk can also check to see if any of the lights are on:

-----Perform a bitwise OR and AND operation for each of the office states---
state_o=(Office_1 or Office_2 or Office_3)
state_a=(Office_1 and Office_2 and Office_3)
```

```

-----Perform a bitwise OR and AND operation for each of the corridor states---
state_co=(Corridor_1 or Corridor_2 or Corridor_3 or Corridor_4 or Corridor_5)
state_ca=(Corridor_1 and Corridor_2 and Corridor_3 and Corridor_4 and Corridor_5)

```

The implementation of the bitwise logic operations (OR and AND) is to check the following conditions.

Logic Operation	Section	Condition
OR	Office	Returns 1 if any of the lights are on, 0 if none are on
AND	Office	Returns 1 if all of the lights are on, 0 otherwise
OR	Corridor	Returns 1 if any of the lights are on, 0 if none are on
AND	Corridor	Returns 1 if all of the lights are on, 0 otherwise

These shall be referenced in latter parts of the example.

3.1.2 Start of business

Consider the first person arrives on the floor through the lift, once the motion sensor detects their movement this should trigger all the corridor lights to turn on. The use of keywords is beneficial in this situation, set all the objects linked to the keyword ‘Corridor’ to turn on when the motion sensor detects this first person entering the floor.

```

-----Empty office and corridor-----
---If there is no one present and all the office and corridor lights are off---
---If the motion sensor detects someone then turn all the corridor lights on---
if(sensor_state==true and state_co==false and state_o==false)
    then
        Corridor:write(1)
        --turns on all objects with tag 'Corridor'
    end

```

Before turning on the corridor light spaceLYnk shall check to see if the motion sensor has detected any movement and if all the office and corridor lights are off, if this is the case the corridor lights shall be turned on. This covers the start of business for the commercial building and if cleaners come in after hours, as business commences people will continue to use the corridor as they make their way to their respective offices.

3.1.3 Close of business

As business ends, people will begin to leave their offices (switching their office lights off) and make their way out of the building via the lift. During this time spaceLYnk shall continually monitor these conditions:

1. Are all the corridor lights still on?
2. Are the office lights still on (is anyone still in their office)?
3. Are all the office lights off (has everyone left their office for the day)?
4. Is there any movement in the corridor?

At the close of business, consider that all people have left their offices (all office lights have been switched off) and are either making their way toward the lift or have already gone. Once the spaceLYnk detects that all of the office lights have been turned off and that the motion sensor can no longer detect any movement, it shall switch off corridor lights. This time delay can be declared like so:

```
-----Timer variable, used to set the length of time after an event has occurred--  
Timer=5 --(in seconds)
```

This time delay can be changed to any value the user sees fit, however for this example 5 seconds shall be used. The following code shall take place when all of the conditions discussed above are met, with the time delay being implemented from this point.

```
-----People leaving the office-----  
  
-----Checks to see if all the corridor lights are on-----  
-----Checks to see if any of the office lights are off-----  
-----Checks to see if all of the office lights are off-----  
-----Checks to see if the motion sensor can detect anyone-----  
  
----If the corridor lights are on and all the office lights have gone off-----  
----and the sensor cannot detect any motion then save this time stamp-----  
if(state_ca==true and state_a==false and state_o==false and sensor_state==false)  
    then  
        initial_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))  
        storage.set('time_3',initial_time) --store this time in a storage key  
  
--After 5 seconds have elapsed from that point, switch OFF corridor lights--  
expected_time=(storage.get('time_3',0))+Timer
```

```

current_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
while(current_time~=expected_time)
do
current_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
expected_time=(storage.get('time_3',0))+Timer
end

```

```
Corridor:write(0)  --Switch off corridor lights
```

As seen above, the bitwise operations, mentioned in chapter (2.1) have been used to monitor each of the conditions discussed at the start of this chapter. The code shall save the time stamp when all of the conditions are met, following that time stamp the code shall then delay execution of the rest of the script for 5 seconds (as specified above), after 5 seconds have elapsed the code shall switch off Corridor lights.

If somebody want to continue chatting in corridor, there is a manual override button they can press to turn the lights back.

There is a KNX object designated for a lighting override switch, this can be a simple switch on the wall outside the lift. If two colleagues are talking and they notice the lights switch off one can simply press the switch to turn the lights back up.

```

-Turn the corridor lights off after another 5 seconds-----
    initial_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
    storage.set('time_4',initial_time2)

    expected_time2=(storage.get('time_4',0))+Timer
    current_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
--Monitors the current state of the motion sensor and the override switch-----
    while(current_time2~=expected_time2)
    do
        current_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
        expected_time2=(storage.get('time_4',0))+Timer
ss2=grp.getvalue('Local','Lighting','Motion_Sensor')
state_override=grp.getvalue('Local','Lighting','Light_Override')
        storage.set('sensor',ss2)
        storage.set('override',state_override)
        if(ss2==true or state_override==true) then

```

```

        storage.set('sensor',ss2)
        storage.set('override',state_override)
        current_time2=expected_time2
    end
end

--If during this time someone comes back into the corridor or if someone presses the
override switch---
--Turn the corridor lights back to 100%-----
-----
    if((storage.get('sensor',0)==false) and (storage.get('override',0)==false)) then
        grp.update('Corridor', mode, 0)
    elseif((storage.get('sensor',0)==true) or (storage.get('override',0)==true)) then
        grp.update('Corridor', mode, 1)
    end
end
end

--If during this time someone comes back into the corridor or if someone presses the
override switch---
--Turn the corridor lights back to 100%-----
-----
    if((storage.get('sensor',0)==false) and (storage.get('override',0)==false)) then
        grp.update('Corridor', mode, 0)
    elseif((storage.get('sensor',0)==true) or (storage.get('override',0)==true)) then
        grp.update('Corridor', mode, 1)
    end
end
end

```

This completes the corridor linking implemented by spaceLYnk.



The codes shown in this section are to be used as a collective, for the purposes of this configuration guide, the different code segments have been shown at each of their relevant sections, the full code can be found in the appendix and in an attached Lua script file.

4 Conclusion

spaceLYnk provides corridor linking functionality for a typical office building.

5 Appendix

Full Commercial Example code

```
-----Motion sensor-----
sensor_state = grp.getvalue('Motion sensor')
--false if there is no one in the corridor
--true if there are people present

-----Return the state of each of the corridor lights-----
Corridor_1 = grp.getvalue('Corridor Light 1 feedback')
Corridor_2 = grp.getvalue('Corridor Light 2 feedback')
Corridor_3 = grp.getvalue('Corridor Light 3 feedback')
Corridor_4 = grp.getvalue('Corridor Light 4 feedback')
Corridor_5 = grp.getvalue('Corridor Light 5 feedback')

-----Return the state of each of the office lights-----
Office_1 = grp.getvalue('Office Light 1 feedback')
Office_2 = grp.getvalue('Office Light 2 feedback')
Office_3 = grp.getvalue('Office Light 3 feedback')

-----Perform a bitwise OR and AND operation for each of the office states---
state_o = (Office_1 or Office_2 or Office_3)
state_a = (Office_1 and Office_2 and Office_3)

-----Perform a bitwise OR and AND operation for each of the corridor states---
state_co = (Corridor_1 or Corridor_2 or Corridor_3 or Corridor_4 or Corridor_5)
state_ca = (Corridor_1 and Corridor_2 and Corridor_3 and Corridor_4 and Corridor_5)

-----Empty office and corridor-----
---If there is no one present and all the office and corridor lights are off---
---If the motion sensor detects someone then turn all the corridor lights on---
if(sensor_state==true and state_co==false and state_o==false)
    then
        Corridor:write(1)
        --turns on all objects with tag 'Corridor'
end
```

```

-----Timer variable, used to set the length of time after an event has occurred--
Timer=5 --(in seconds)

-----People leaving the office-----

-----Checks to see if all the corridor lights are on-----
-----Checks to see if any of the office lights are off-----
-----Checks to see if all of the office lights are off-----
-----Checks to see if the motion sensor can detect anyone-----

----If the corridor lights are on and all the office lights have gone off-----
----and the sensor cannot detect any motion then save this time stamp-----
if(state_ca==true and state_a==false and state_o==false and sensor_state==false)
    then
        initial_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
        storage.set('time_3',initial_time) --store this time in a storage key

--After 5 seconds have elapsed from that point, switch OFF corridor lights--
expected_time=(storage.get('time_3',0))+Timer
current_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
while(current_time~=expected_time)
do
    current_time=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
    expected_time=(storage.get('time_3',0))+Timer
end

Corridor:write(0) --Switch off corridor lights

--Turn the corridor lights off after another 5 seconds-----
    initial_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
    storage.set('time_4',initial_time2)

        expected_time2=(storage.get('time_4',0))+Timer
        current_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))
--Monitors the current state of the motion sensor and the override switch-----
    while(current_time2~=expected_time2)
    do
        current_time2=time_convert(os.date('%H'), os.date('%M'), os.date('%S'))

```

```

        expected_time2=(storage.get('time_4',0))+Timer
ss2=grp.getvalue('Local','Lighting','Motion_Sensor')
state_override=grpg.getvalue('Local','Lighting','Light_Override')
        storage.set('sensor',ss2)
        storage.set('override',state_override)
        if(ss2==true or state_override==true) then
            storage.set('sensor',ss2)
            storage.set('override',state_override)
            current_time2=expected_time2
        end
    end

--If during this time someone comes back into the corridor or if someone presses the
override switch---
--Turn the corridor lights back to 100%-----
-----
    if((storage.get('sensor',0)==false) and (storage.get('override',0)==false)) then
        grp.update('Corridor', mode, 0)
    elseif((storage.get('sensor',0)==true) or (storage.get('override',0)==true)) then
        grp.update('Corridor', mode, 1)
    end
end
end

```

Schneider Electric Industries SAS
Head Office 35, rue Joseph Monier 92506 Rueil-Malmaison Cedex France http://www.schneider-electric.com