

---

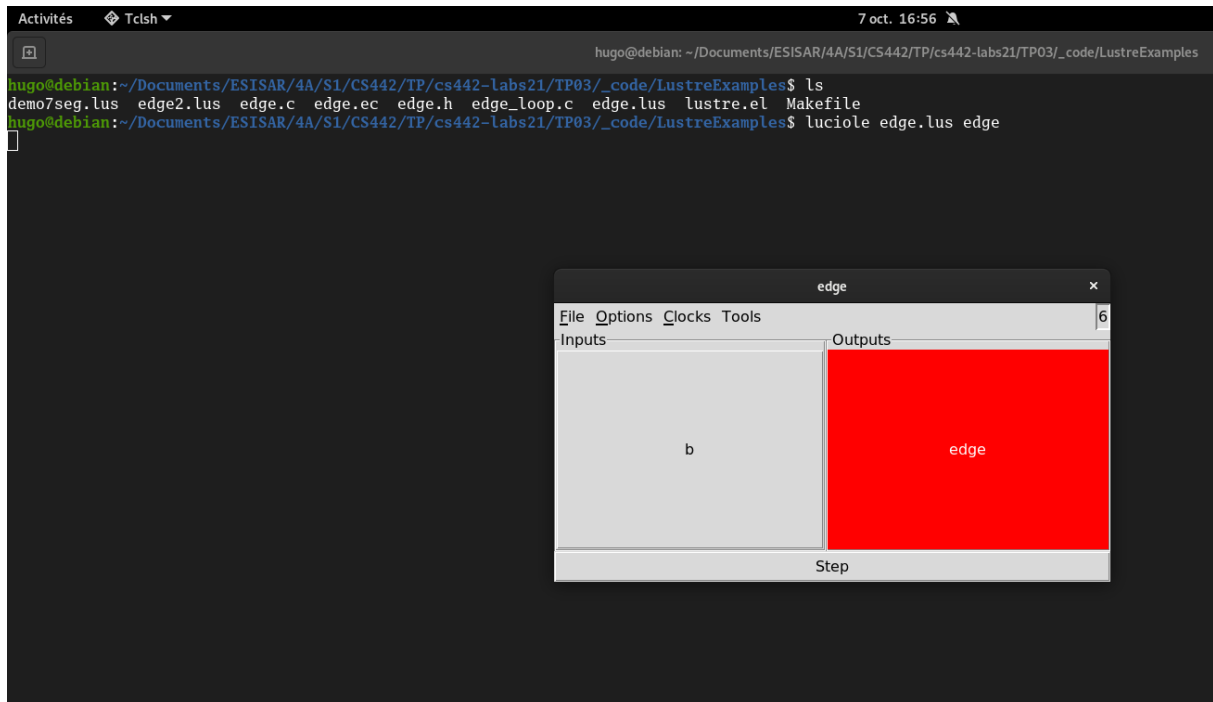
# **Compte-rendu TP3 Lustre**

DOUGY Hugo & LEVERN Albéric

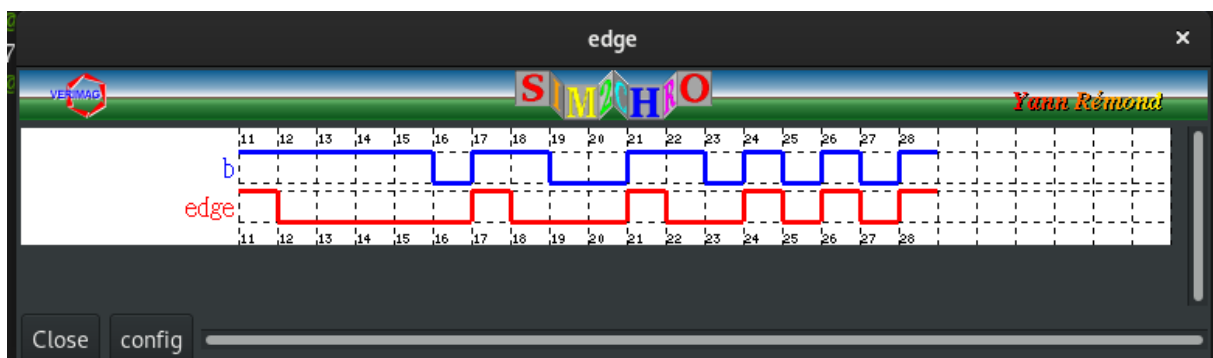
## Step 1:

### A simple compilation and simulation

On a exécuté la commande `lus2c edge.lus edge -loop` afin d'exécuter le programme `edge.lus` avec `luciole` On obtient le résultat suivant:



**Figure 1:** Résultat du programme `edge.lus` avec `luciole`



**Figure 2:** Chronogramme du programme `edge.lus`

On compile le programme avec la commande `gcc -o edge edge*.c`

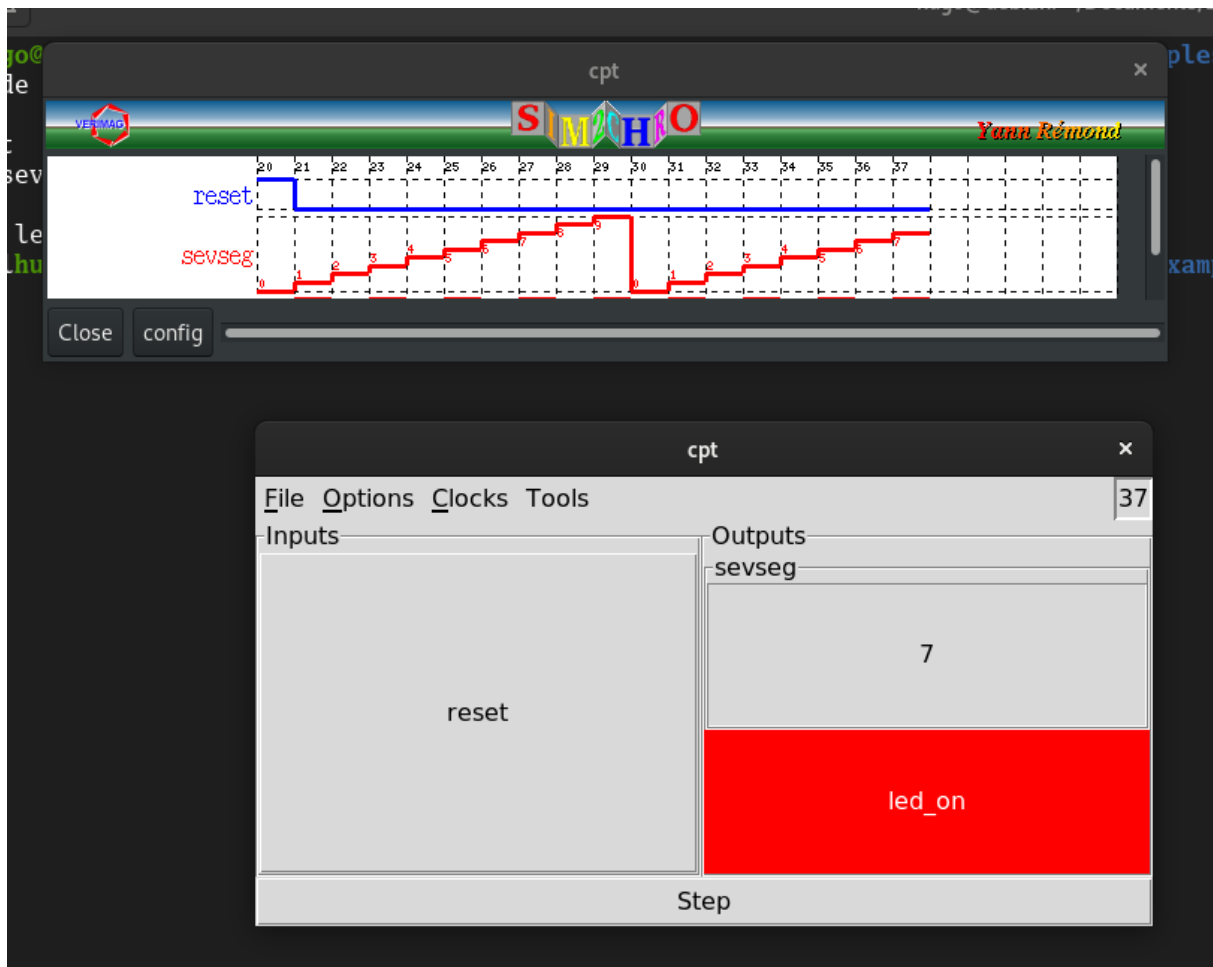
On teste le programme executable edge. On obtient le résultat suivant:

```
hugo@debian:~/Documents/ESISAR/4A/S1/CS442/TP/cs442-labs21/TP03/_code/LustreExamples$ ls
demo7seg.lus  edge2.lus  edge.c  edge.ec  edge.h  edge_loop.c  edge.lus  lustre.el  Makefile
hugo@debian:~/Documents/ESISAR/4A/S1/CS442/TP/cs442-labs21/TP03/_code/LustreExamples$ gcc
demo7seg.lus  edge.c          edge.h          edge.lus      Makefile
edge2.lus     edge.ec        edge_loop.c    lustre.el
hugo@debian:~/Documents/ESISAR/4A/S1/CS442/TP/cs442-labs21/TP03/_code/LustreExamples$ gcc -o e
dge edge*.c
hugo@debian:~/Documents/ESISAR/4A/S1/CS442/TP/cs442-labs21/TP03/_code/LustreExamples$ ls
demo7seg.lus  edge2.lus  edge.ec  edge_loop.c  lustre.el
edge          edge.c     edge.h   edge.lus     Makefile
hugo@debian:~/Documents/ESISAR/4A/S1/CS442/TP/cs442-labs21/TP03/_code/LustreExamples$ ./edge
## STEP 1 #####
b (1,t,T/0,f,F) ? 0
edge = false
## STEP 2 #####
b (1,t,T/0,f,F) ? 1
edge = true
## STEP 3 #####
b (1,t,T/0,f,F) ? 0
edge = false
## STEP 4 #####
b (1,t,T/0,f,F) ? 1
edge = true
## STEP 5 #####
b (1,t,T/0,f,F) ? 0
edge = false
## STEP 6 #####
b (1,t,T/0,f,F) ? 0
edge = false
## STEP 7 #####
b (1,t,T/0,f,F) ? 0
edge = false
## STEP 8 #####
b (1,t,T/0,f,F) ? 1
edge = true
## STEP 9 #####
b (1,t,T/0,f,F) ? 1
edge = false
## STEP 10 #####
b (1,t,T/0,f,F) ? 1
edge = false
## STEP 11 #####
b (1,t,T/0,f,F) ? ☐
```

**Figure 3:** Résultat de l'executable edge

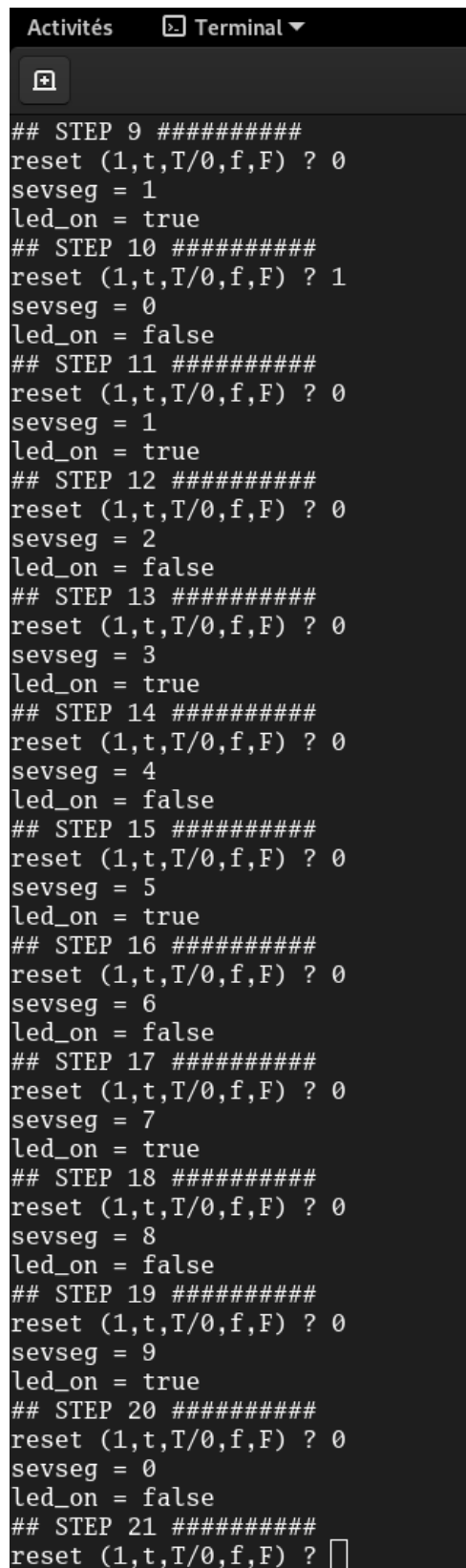
On test avec le programme demo7seg.lus

On recommence les opérations de compilation et avec luciole. On obtient le résultat suivant:



**Figure 4:** Résultat du programme demo7seg.lus avec luciole

On test également l'exécutable:



```
Activités Terminal
## STEP 9 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 1
led_on = true
## STEP 10 #####
reset (1,t,T/0,f,F) ? 1
sevseg = 0
led_on = false
## STEP 11 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 1
led_on = true
## STEP 12 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 2
led_on = false
## STEP 13 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 3
led_on = true
## STEP 14 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 4
led_on = false
## STEP 15 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 5
led_on = true
## STEP 16 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 6
led_on = false
## STEP 17 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 7
led_on = true
## STEP 18 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 8
led_on = false
## STEP 19 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 9
led_on = true
## STEP 20 #####
reset (1,t,T/0,f,F) ? 0
sevseg = 0
led_on = false
## STEP 21 #####
reset (1,t,T/0,f,F) ?
```

**Figure 5:** Résultat de l'exécutable demo7seg

Pour compter, on met le signal reset à 0.

## Step 2:

### Exercice 1: Air-conditioner controller

Nous avons choisis de concevoir uniquement une climatisation car le sujet ne spécifie pas la conception d'un chauffage.

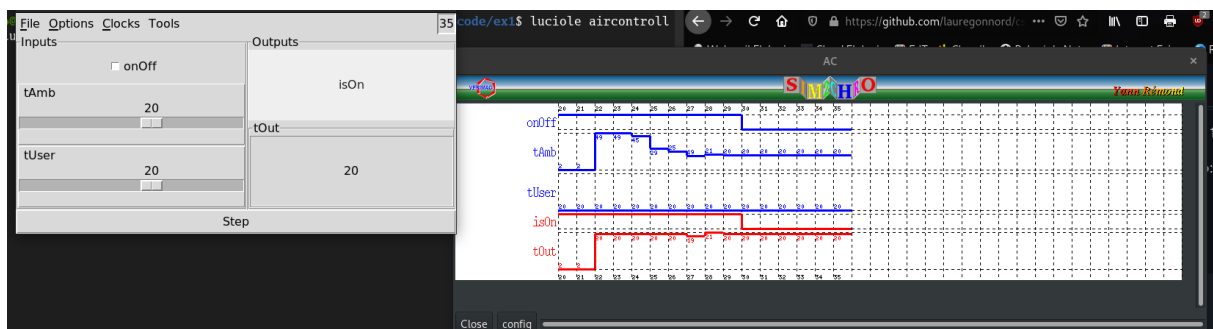
Notre système ne fonctionne que lorsque l'on décide d'activer l'interrupteur onOff. L'air conditionné est donc activé lorsque `isOn = true`.

Si jamais le système ne fonctionne pas (`isOn = false`) alors il sauvegarde la valeur de consigne donnée par l'utilisateur pour pouvoir se réactiver quand `isOn = true`.

La climatisation ne se déclenche que lorsque la température ambiante est supérieure à la température de commande. Dans la situation inverse l'air conditionné soufflera un air à la température ambiante.

Nous avons fait le choix d'activer la climatisation avec un hystérésis de 1°.

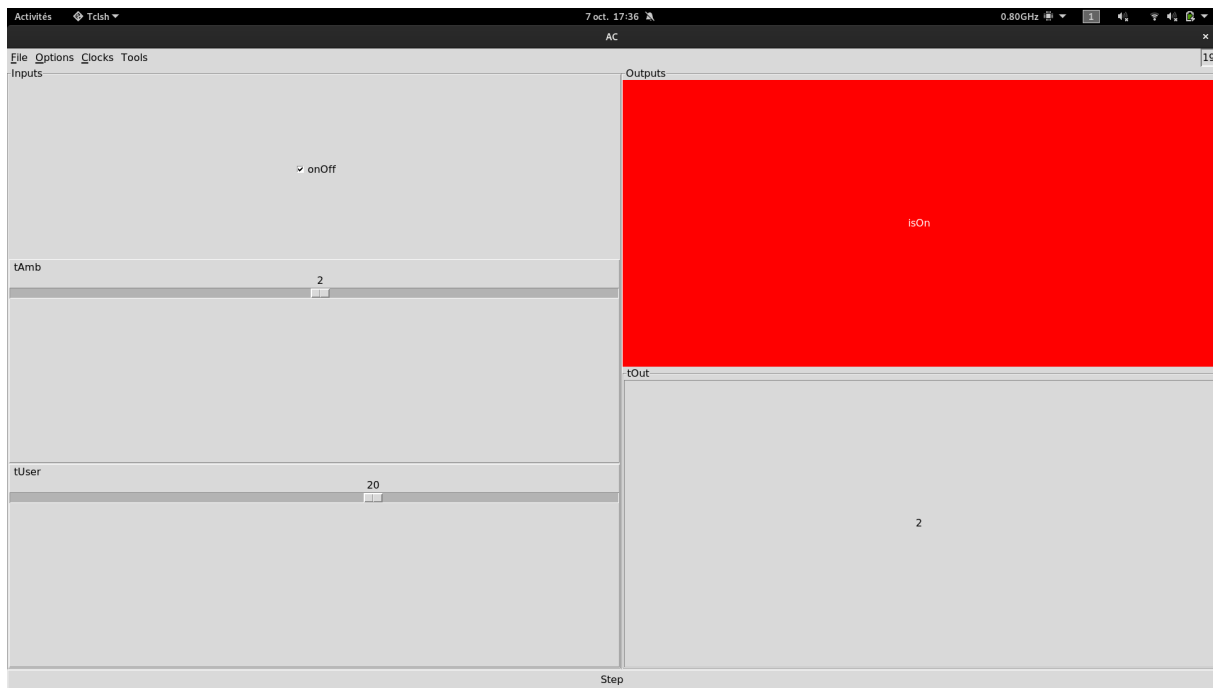
On a le résultat suivant en simulant le programme avec Luciole:



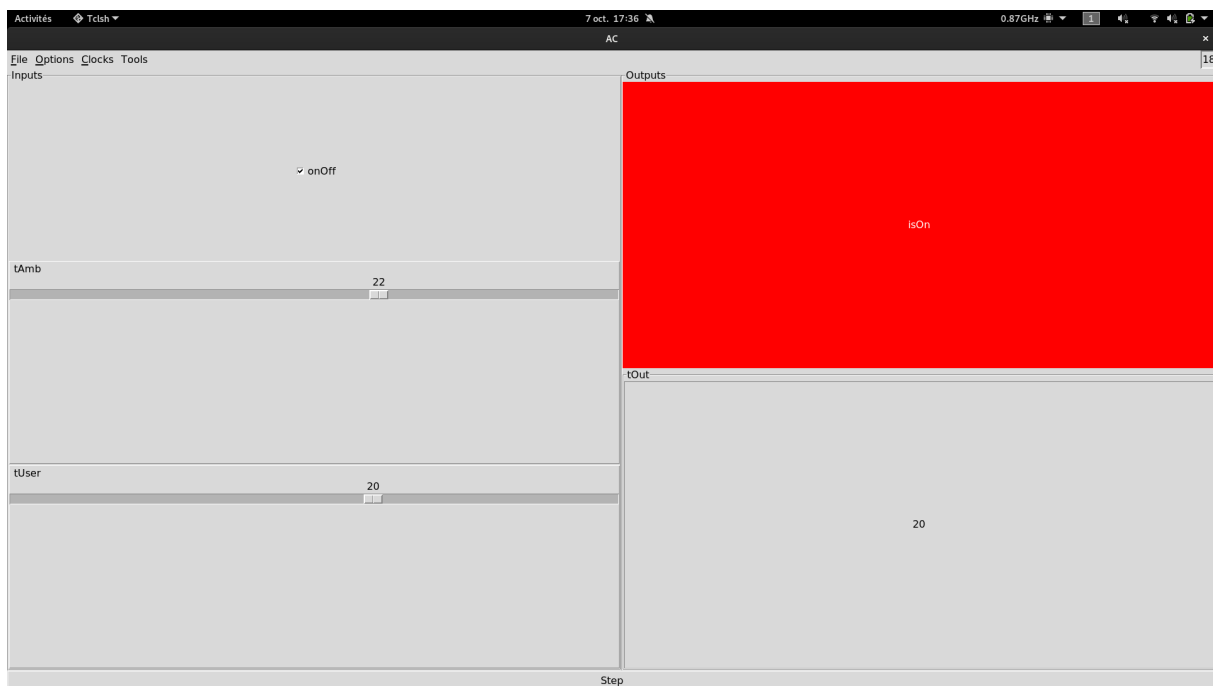
**Figure 6:** Résultat du programme aircontroller.lus avec luciole

On peut voir avec le chronogramme que l'on tend bien vers notre valeur tUser en fonction de la température tAmb. De plus on sauvegarde la valeur de tAmb lorsque `onOff = false`.

On a également les 2 cas suivants où la  $t_{Amb} > t_{User}$  et  $t_{Amb} < t_{User}$



**Figure 7:**  $t_{Amb} < t_{User}$



**Figure 8:**  $t_{Amb} > t_{User}$

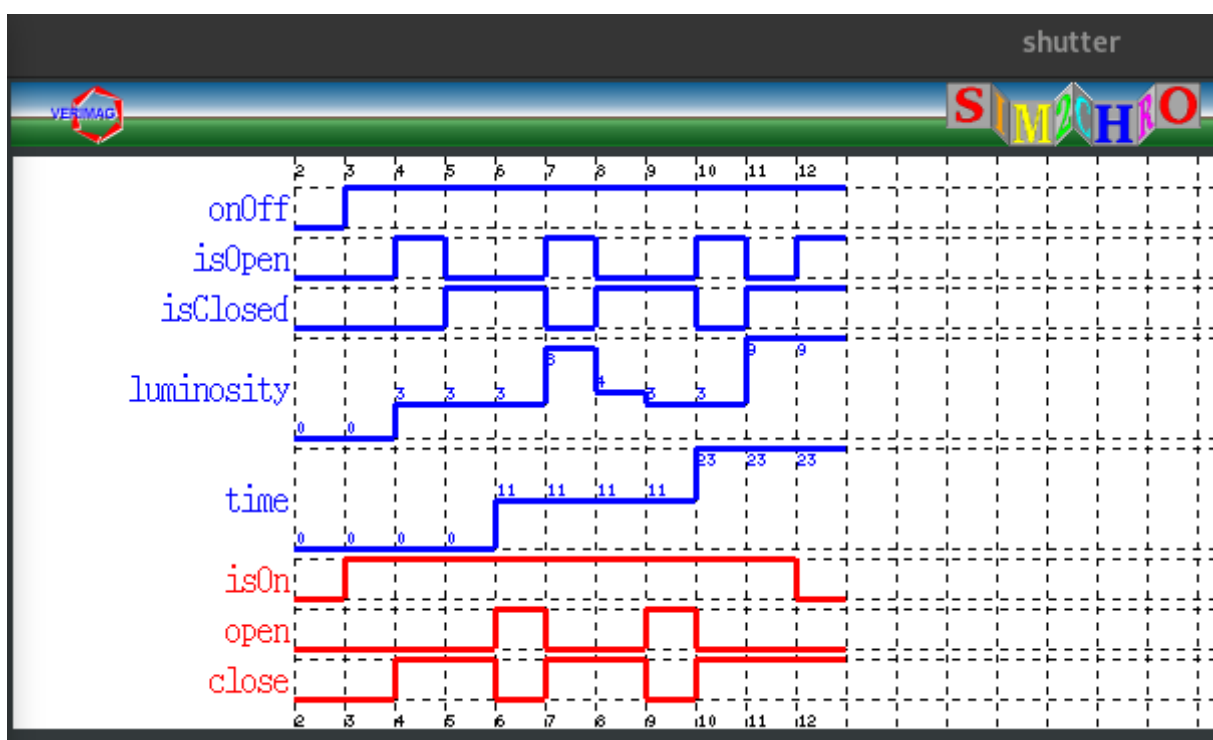
## Exercice 2: Roller shutter controller

Pour cet exercice, nous avons choisi d'implémenter un second noeud: `hour` afin de savoir si l'heure est comprise entre 22h et 6h. Si oui, le noeud renvoie `true`. Le but est d'alléger le code et de le rendre plus lisible.

Pour l'implémentation du noeud principal, on procède par priorité. C'est à dire qu'on regarde premièrement les états des volets. Si ils ne sont pas dans le bon état on ne change pas leur état.

On regarde ensuite l'heure puis la luminosité.

En simulant notre programme avec `luciole`, on obtient le résultat suivant:



**Figure 9:** Chronogramme du programme `shutter_controller.lus`

On voit bien que toutes les conditions sont respectées et que l'heure est prioritaire par rapport à la luminosité pour la fermeture.

De plus, les sorties `open` et `close` ne sont jamais vraies en même temps. Et si `isOpen` et `isClosed` sont vraies en même temps, les volets s'éteignent.

**Dans la simulation, il faut faire attention à bien changer l'état manuellement de `isOpen` et `isClosed`**

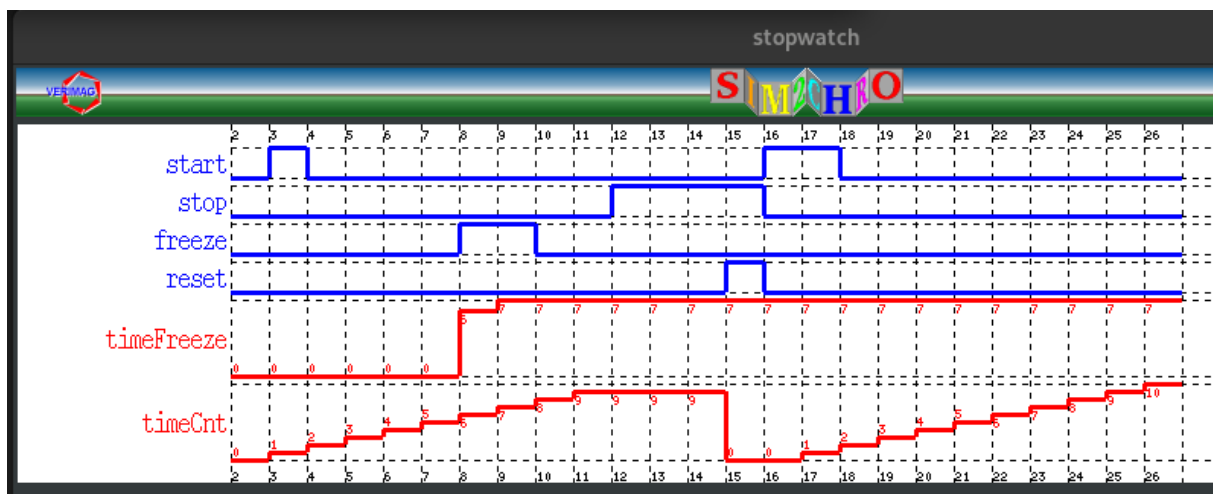


### Exercice 3: Stopwatch

Pour cet exercice, on crée un nouveau noeud `swtich` afin de savoir quand à lieu un changement sur la variable `start`.

Si `start` repasse à `true` alors on remet le chronomètre à 0 et on attend un cycle d'horloge pour recommencer à recompter.

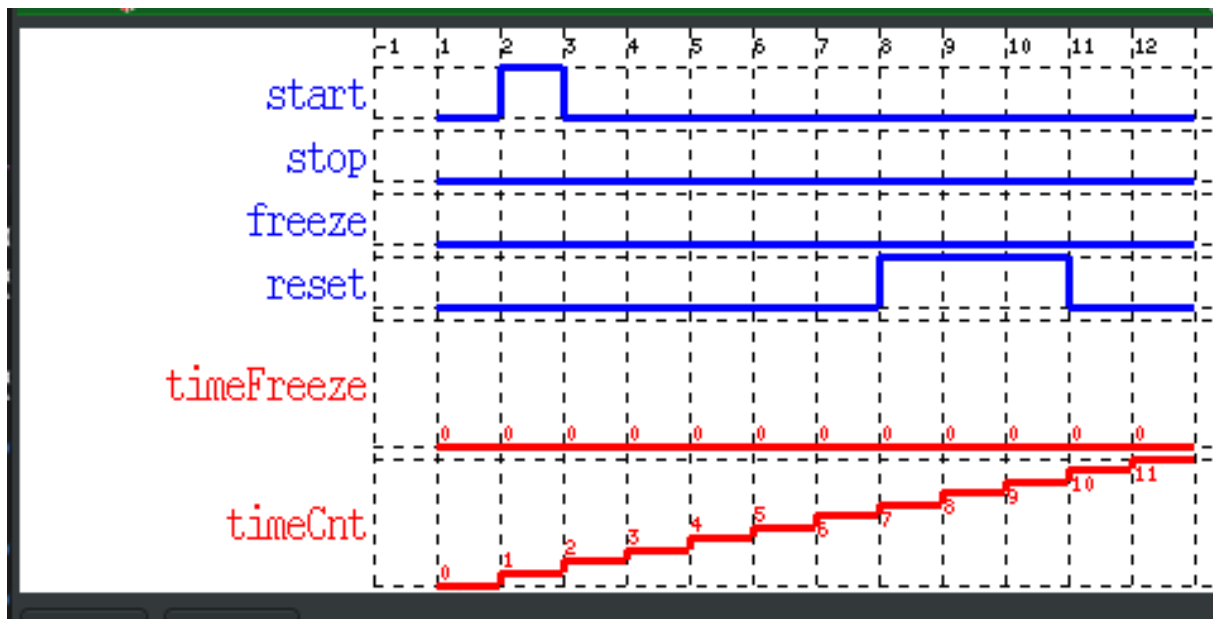
En simulation, on obtient le chronogramme suivant:



**Figure 10:** Chronogramme du programme stopwatch.lus

Quand `freeze` est à `true`, on a bien la sauvegarde du compteur. Quand `stop` est à `true`, on arrête de compter et quand `reset` est à `true` quand `stop` est à `true` on réinitialise le compteur.

De plus, si `reset` est à `true` sans `stop` cela n'affecte pas comptage.



**Figure 11:** Chronogramme du programme stopwatch.lus avec reset sans stop