# Exemple activité étudiant MEEF S2I initiation "Modélisation comportementale : Machine à Etats"

## 1. Introduction : les diagrammes d'état

Les diagrammes d'état sont l'outil SysML pour décrire le fonctionnement séquentiel d'un système. Il est habituel de dire que les Grafcets sont un cas particulier des diagrammes d'état. Avec l'outil Simulink, les diagrammes d'état seront conçus et simulés. Il sera possible également de les « transformer » en un programme téléchargeable directement dans une cible, comme un microcontrôleur (ATMEGA 328 de la carte ARDUINO UNO par exemple ou sur du STM32 ou encore ARDUINO MEGA2560).

Sous Simulink, les diagrammes d'état sont nommés *Chart*, ou *State Chart*.

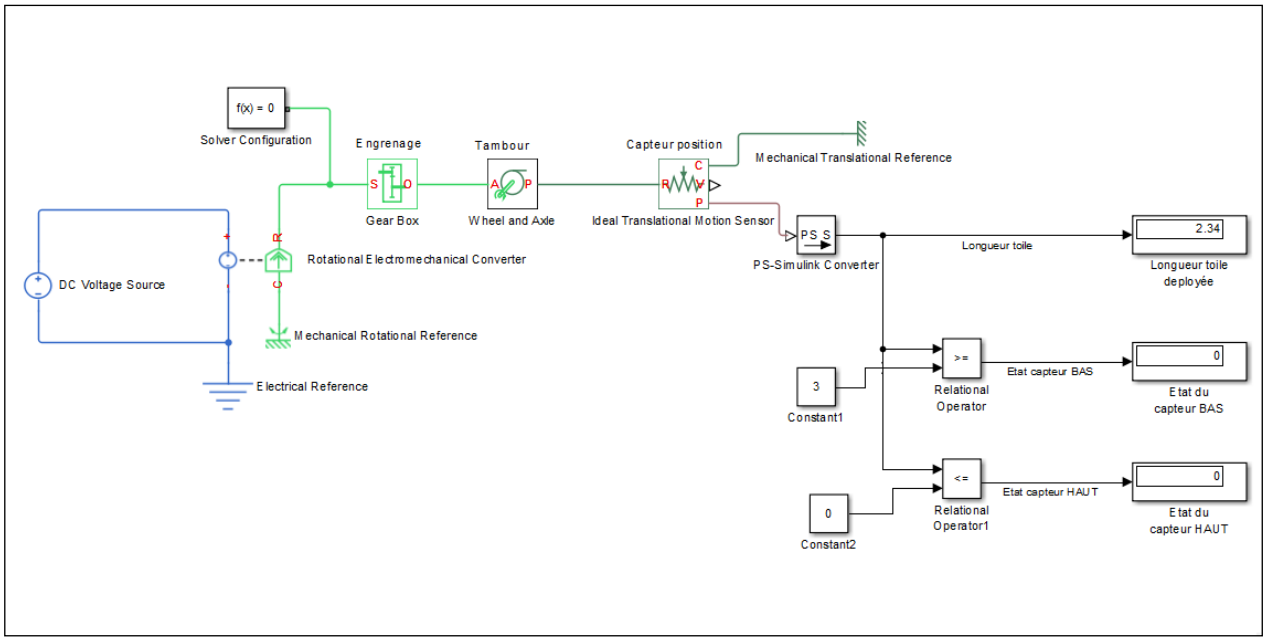
## 2. Objectifs:

- Etre capable de concevoir un modèle de comportement séquentiel à l'aide de diagrammes d'état dans un environnement Matlab/Simulink.
- Compiler puis implanter le programme obtenu dans une carte Arduino UNO.
- Simuler le comportement avec une **Partie Opérative** modélisée.

## 3. Cas d'étude : un store banne motorisé

Afin d'illustrer les propos précédents, nous allons modéliser un store automatisé qui se remonte ou qui se rabaisse en fonction des demandes utilisateur dans un premier temps puis, des conditions climatiques dans un second temps.

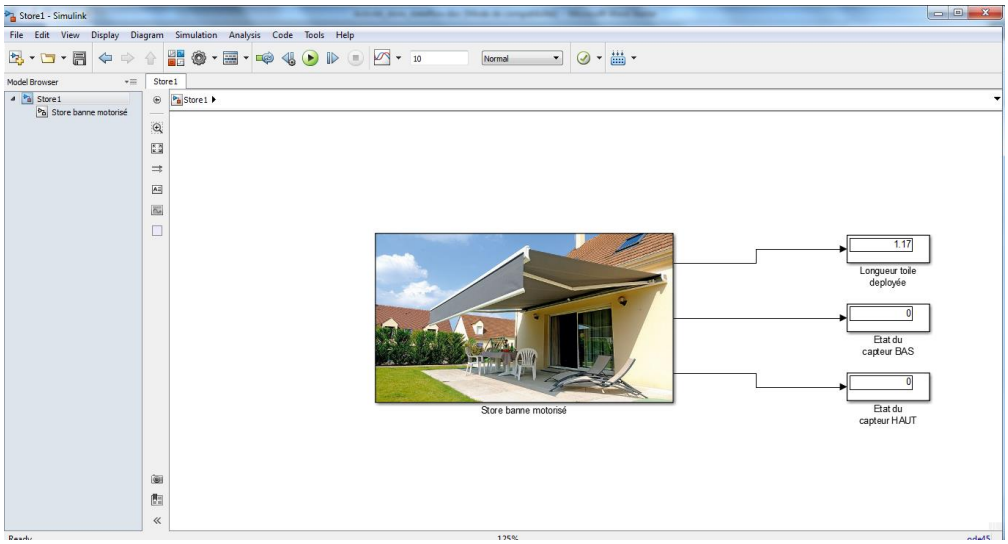
Le modèle multiphysique de la partie opérative est déjà modélisé dans le fichier "**Store1.slx**". Ce modèle, simplifié volontairement et reproduit ci-dessous, est constitué d'une alimentation DC de 12V, un moteur à courant continu, un réducteur mécanique de rapport 100, un système de transformation de mouvement rotation/translation (tambour d'enroulement/déroulement de rayon 70mm), suivi de deux capteurs de position.




On peut calculer le déplacement de la toile du store pour une durée de simulation de 20s. En effet, le moteur alimenté sous 12V tourne à 1600tr/min, soit en 20s : 533,33tr. Après le réducteur, le tambour tourne 100 fois moins vite soit 5,33tr.

**Pour un rayon de 70mm, on obtient donc un déplacement de 2*pi*5;33*0,07=2,34m.**

- **Ouvrir** le fichier "Store1.slx" depuis la fenêtre « Current Folder » de Matlab. Vous devez obtenir sur votre écran une fenêtre Simulink conforme à l'image ci-dessous. Vous pouvez visualiser le contenu du bloc « Store banne motorisé » en le sélectionnant, puis en faisant **Ctrl+U**.




- **Lancer une simulation** sur 20s et "vérifier" la longueur déployée de la toile du store.

Le problème, ici, est que la longueur déroulée dépend uniquement du temps de simulation ! Nous souhaitons maintenant, par programmation, limiter la longueur déroulée du store à 3m par exemple. **Nous allons donc utiliser pour cela une programmation par diagramme d'état.**

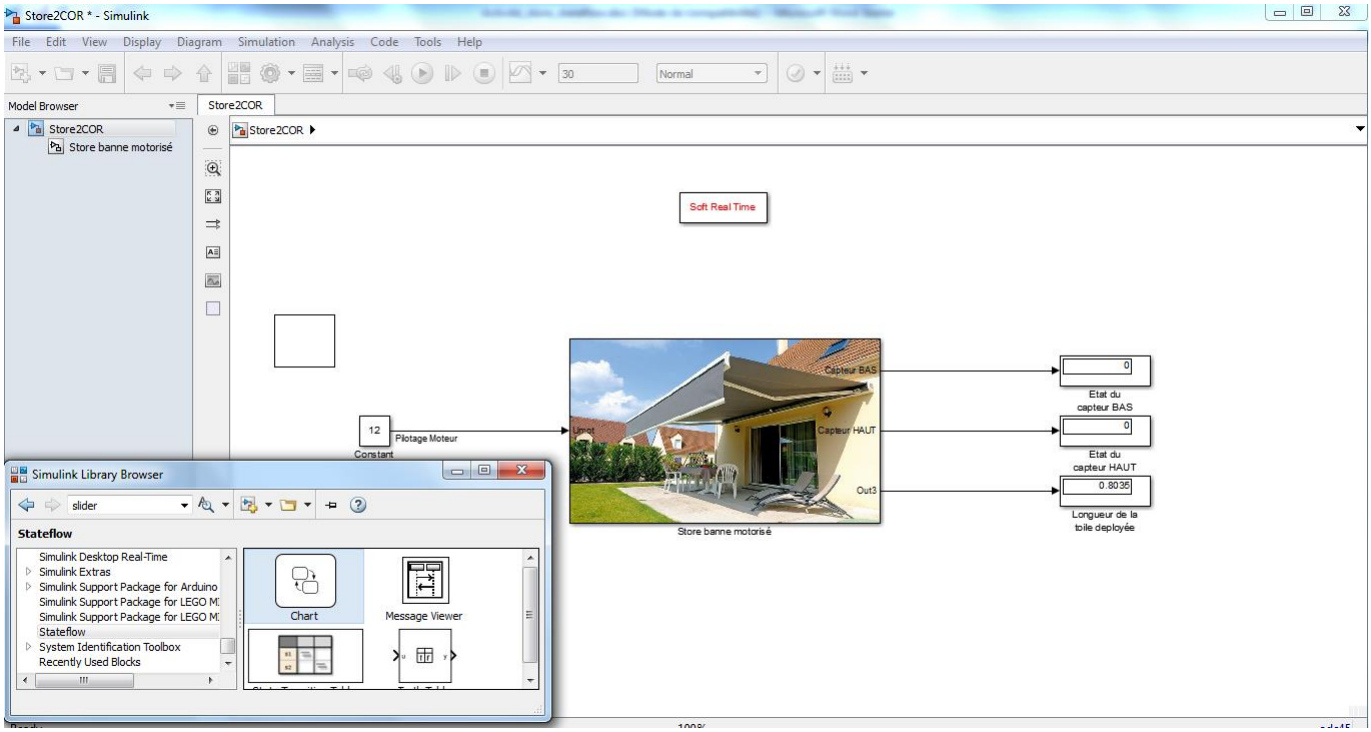
### On souhaite, pour débuter, programmer 2 actions :

1- Faire descendre le store.

2- Arrêter le moteur du store après une course de 3m.

Programmation en machine à états sous Simulink - module STATEFLOW:

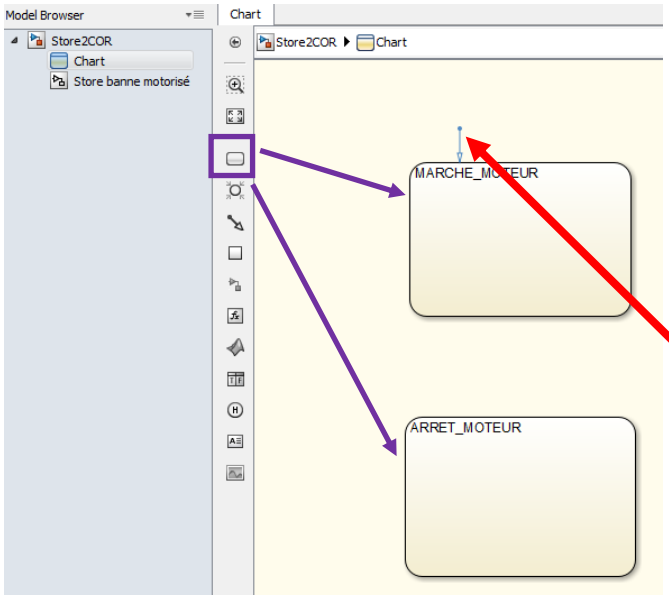
- **Ouvrir** le fichier "Store2.slx", puis à **partir des outils Stateflow** (disponible dans Simulink Library Browser), **ajouter à votre modèle un "Chart",** vous devez obtenir ceci :




Dans le "Chart" nous aurons 2 états : "**MARCHE_MOTEUR**" et "**ARRET_MOTEUR**".

- **En double cliquant** sur le "Chart", vous ouvrez le diagramme d'état. Pour placer les 2 états, **sélectionner** "State" puis "**glisser-déposer**" les états dans le diagramme. **Nommer** les états.

En règle générale, on va éviter dans Matlab les caractères accentués et les espaces…




Les « **MARCHE_MOTEUR** » et « **ARRET_MOTEUR** » sont de simples étiquettes, elles ne donnent aucun renseignement sur le fonctionnement du système.

S'il ne se place pas par défaut, **positionner** sur votre diagramme l'état initial du système et le lier à "MARCHE_MOTEUR".

Le diagramme d'état comporte : **des entrées**, qui proviennent des capteurs du système et **des sorties**, vers les actionneurs du système.

Il faut maintenant définir avec précision les actions associées aux états. Les actions peuvent être effectuées à trois moments, précisés par des mots clefs :

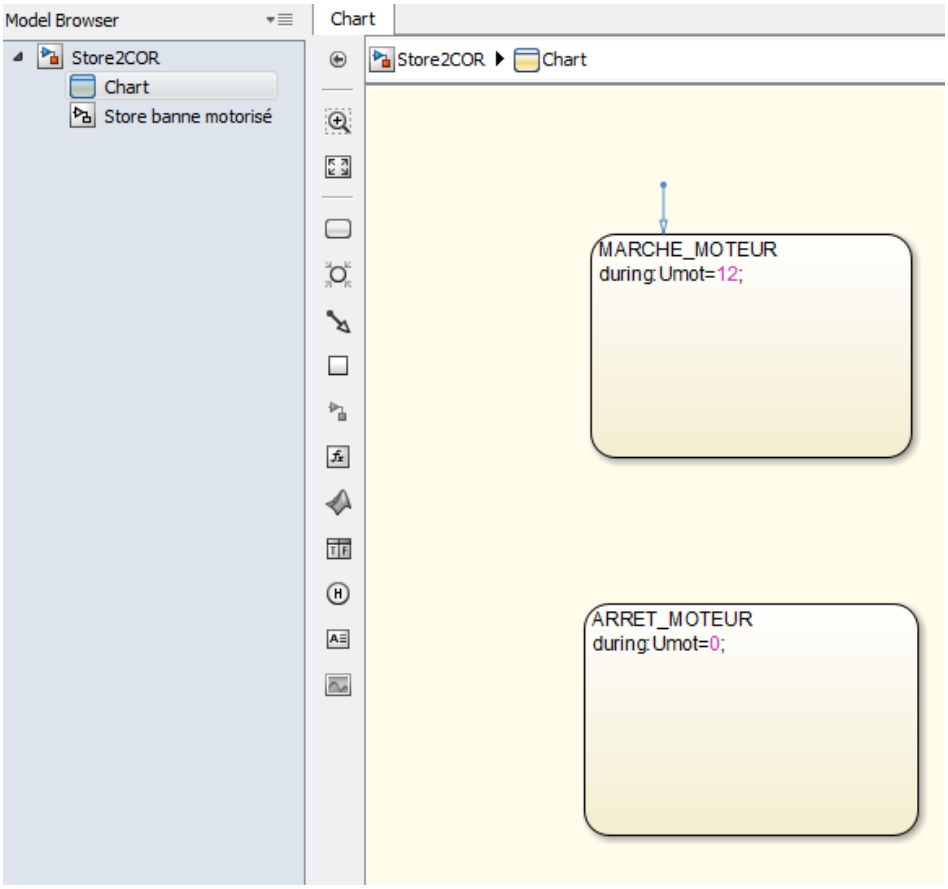
« **entry:** » : action qui sera effectuée à l'entrée dans l'état ;

« **during:** » : action effectuée tant que l'état est actif ;

« **exit:** » : action effectuée à la sortie de l'état.

Dans notre cas, il faut que dans l'état « MARCHE_MOTEUR », la tension d'alimentation du moteur soit à 12V et dans l'état « ARRET_MOTEUR », la tension d'alimentation du moteur soit à 0V.

- ❑ **Renseigner** les actions associées aux états ainsi :



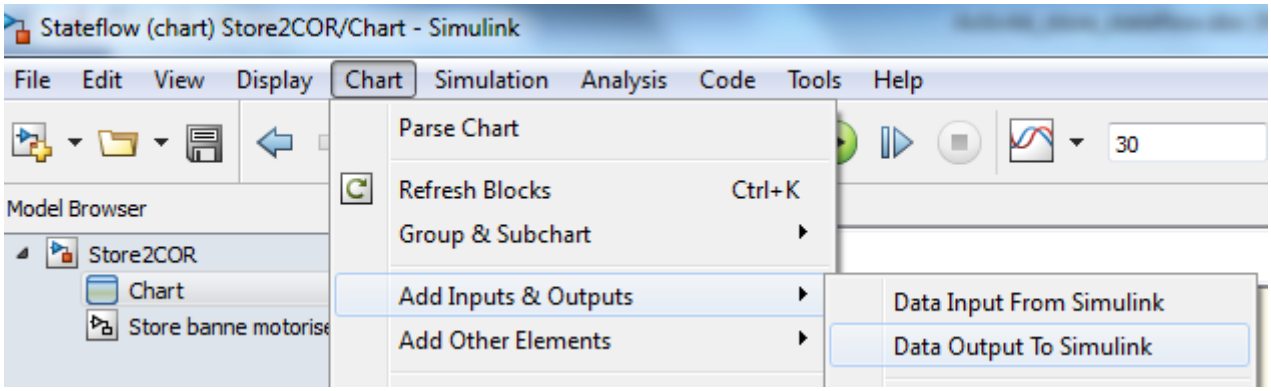

***Attention :***

• les mots clefs 'entry', 'during', 'exit': doivent être écrits sans majuscule, et sans espace ;

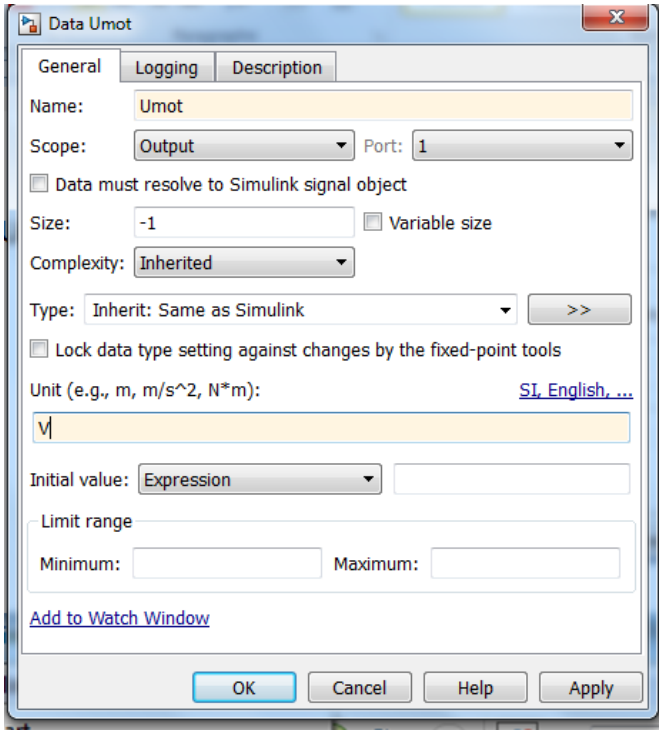
• ne pas oublier le ' ; ' à la fin de la ligne ;

• les chiffres passent en couleur magenta si la syntaxe est correcte.

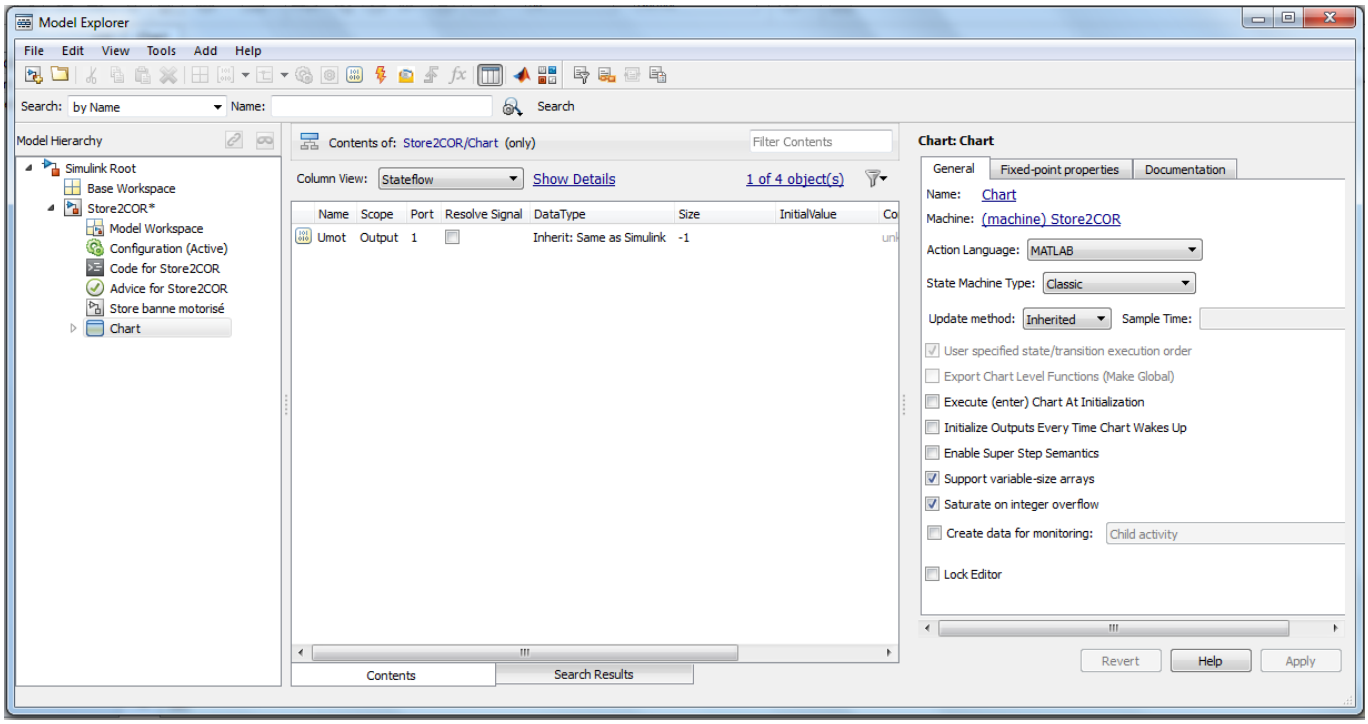
- ❑ **Fermez** la fenêtre Stateflow. Vous remarquez que… rien n'a changé ! Il n'est pas possible de connecter la sortie "Umot" au moteur du modèle.
- ❑ Il faut indiquer à Stateflow que Umot est une sortie vers l'espace Simulink, pour cela :
- ❑ Dans le "Chart", dans le bandeau de commande (en haut) **sélectionner** "chart" puis "add Inputs & Outputs" puis "Data Output To Simulink".



- **Renommer** la variable de sortie en Umot :




- Par un **clic droit** n'importe où dans le diagramme et en **choisissant** "Explore" vous devez avoir la liste des variables actuellement utilisées dans le "Chart".



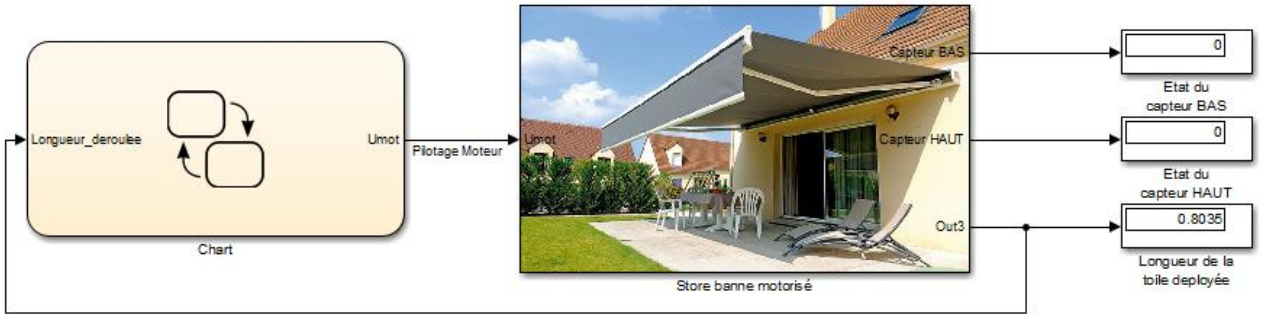

On voit apparaitre sur le bloc "Chart", dans le modèle, une sortie "Umot". Pour la connecter, il faut lier les deux blocs.

- **Relier** la sortie Umot du bloc Stateflow avec l'entrée Umot du bloc Simulink « Store banne motorisé ».

Vous devez obtenir le schéma ci-après :


Pour assurer la prise en compte de la longueur déroulée du store nous allons créer une entrée "**longueur_deroulee**".

- ❑ **Procéder** comme précédemment mais **utiliser une entrée « input »** au lieu d'une sortie « output ». Vous devez arriver au résultat suivant, en ayant **rebouclé l'information de position** du capteur.




Pour tenir compte de cette information de longueur déroulée dans le diagramme, il faut créer des transitions entre les 2 états précédemment définis.

- ❑ Dans le "Chart", **dessiner** la transition en suivant la procédure indiquée dans le cadre; puis **inscrire la condition** que l'on souhaite pour la transition. Dans notre cas : longeur_deroulee>=3, entre crochets.

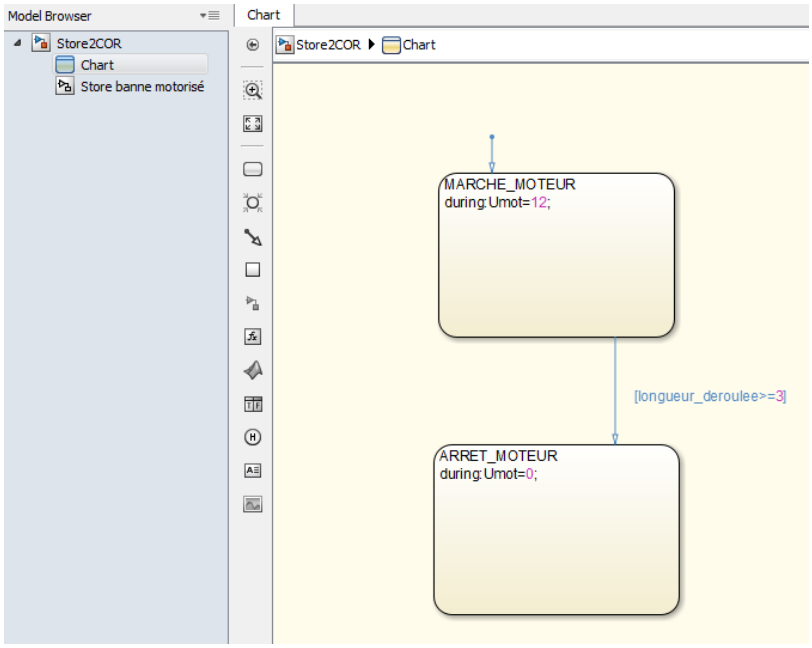
Pour dessiner la transition approcher votre souris du bord du cadre de l'état d'où elle part et lorsque le curseur passe en croix noire, cliquer gauche puis maintenir enfoncer jusqu'au bord du cadre de l'état où on vaut aller.

**En syntaxe Stateflow :**

- les tests sont écrits entre crochets '[ ]' ;

- les comparaisons s'écrivent '>=' ; '>' ; '=='(égal) ; '<'; '>='; '!=' (différent),

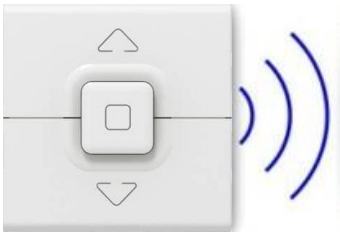
- ET (AND) s'écrit && ; OU (OR) s'écrit || (deux fois Alt Gr – 6).



- ❑ Effectuer une simulation sur 35s. Vérifier si le résultat obtenu correspond au comportement attendu.

Grâce à la bibliothèque "**Soft Real Time**", la simulation se déroule en un temps proche du temps réel qui s'écoule.

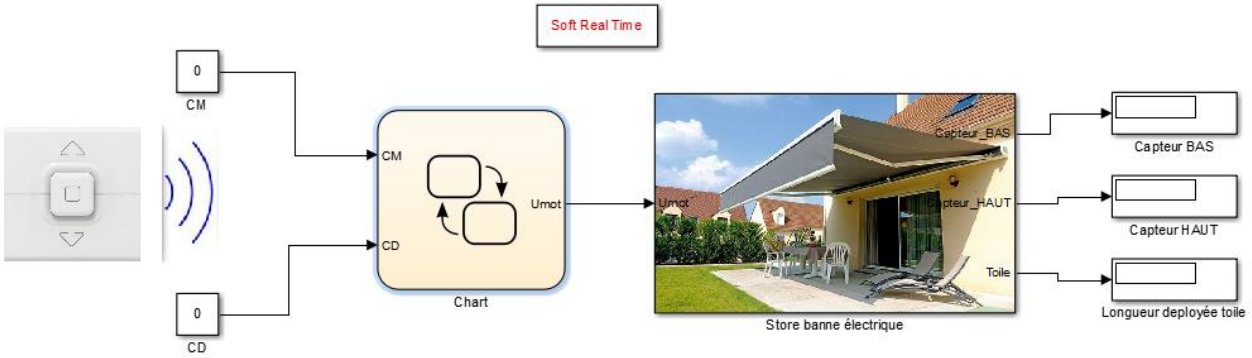
## 4. Modéliser le pilotage manuel du store Banne



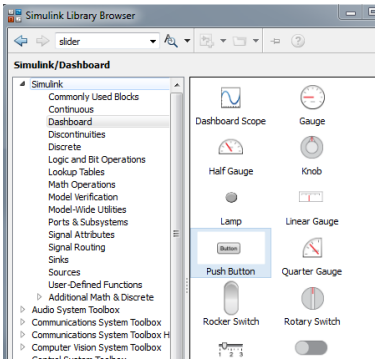
Le store banne électrique est équipé d'un module sans fil - *trois boutons poussoirs* - qui permet le pilotage à distance. On rappelle que les capteurs BAS et HAUT du store banne permettent de détecter les positions limites. Soit respectivement, toile déroulée au maximum et toile totalement enroulée dans le coffre.

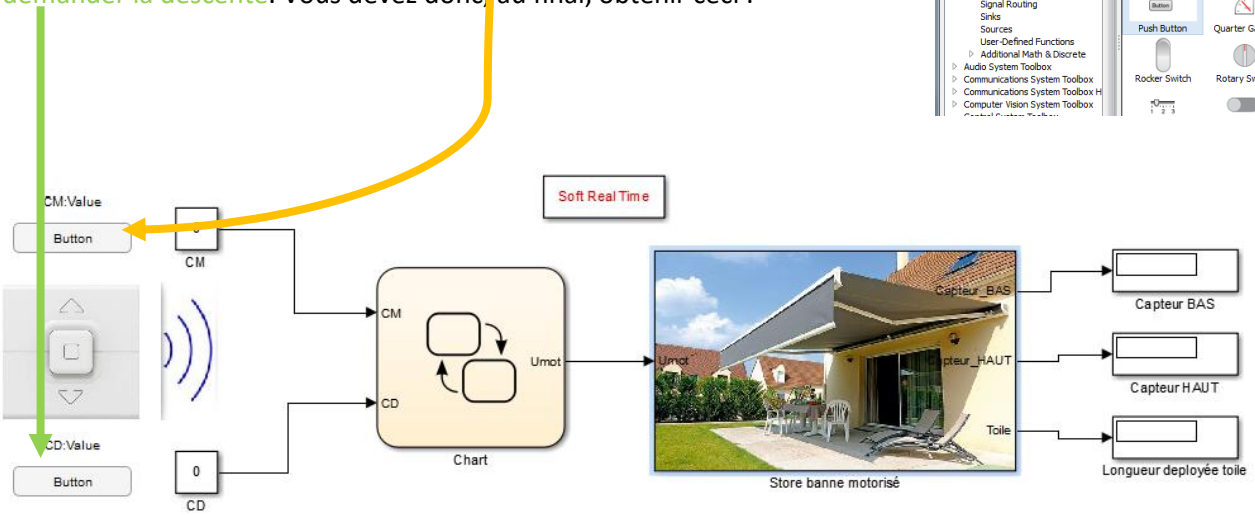
Comportement attendu: si l'utilisateur appui sur le bouton poussoir HAUT, il demande la montée du store et si il appui sur le bouton poussoir BAS, à l'opposé, la toile se déroule. Ces deux actions, enrouler et dérouler la toile, doivent cesser lorsque les capteurs détectent les positions finales.

- ❑ **Ouvrir** le fichier "Store3.slx". Vous devez obtenir sur votre écran une fenêtre Simulink conforme à l'image ci-dessous.




- ❑ A partir des outils **Simulink** puis **Dashboard** (disponible dans Simulink Library Browser), **ajouter** à votre modèle deux **"Push Button"** : un bouton poussoir pour demander une montée de la toile et un deuxième pour demander la descente. Vous devez donc, au final, obtenir ceci :

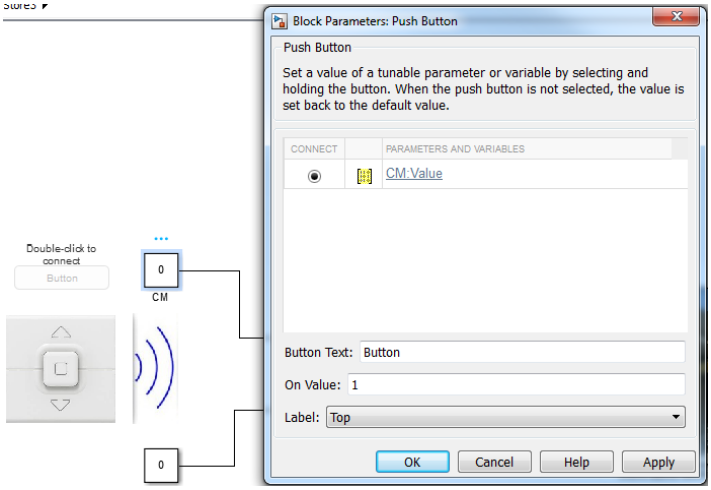



Il reste maintenant à associer les boutons poussoirs avec les entrées concernées. En effet, sur le « Chart », l'entrée CM recevra la demande de montée de la toile par un appui sur le bouton poussoir HAUT alors que l'entrée CD recevra la demande de descente par le bouton poussoir BAS. La démarche sera la suivante :

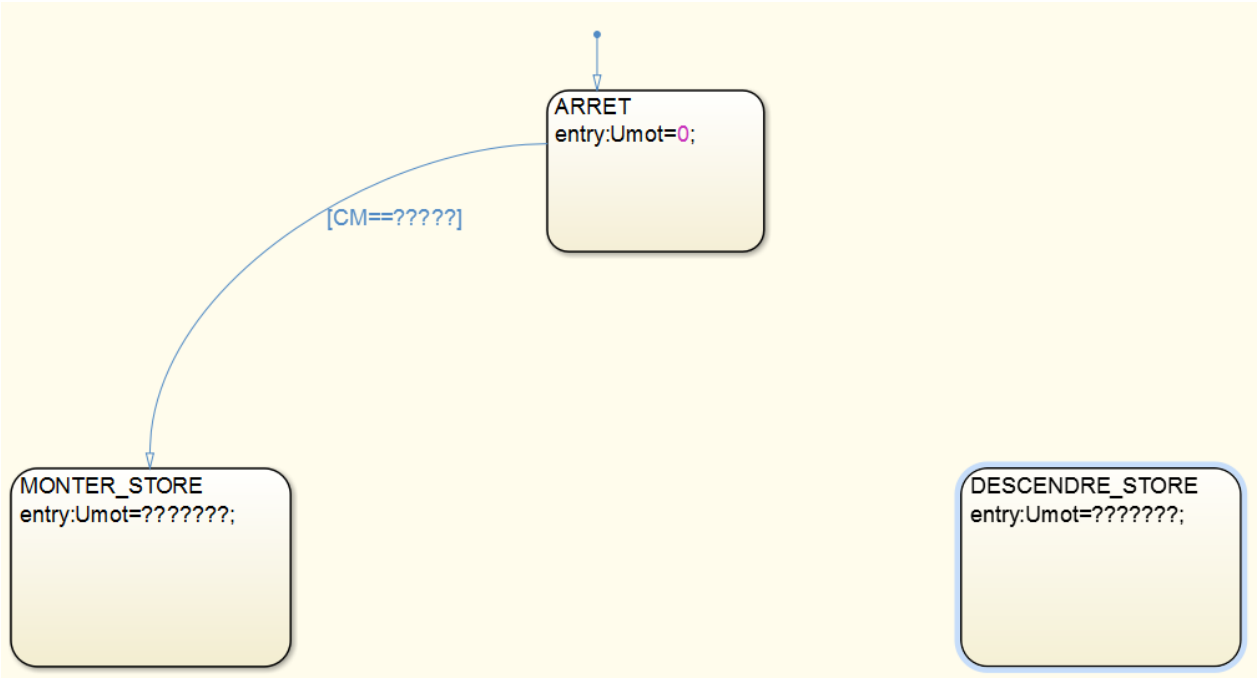
- ❑ En **double cliquant** sur le **bouton poussoir HAUT**, vous ouvrez la fenêtre « Block Parameters :Push Button ». **Sélectionner** avec la souris l'entrée constante **CM** puis **cliquer sur CONNECT** dans la fenêtre Push Button. Puis **valider** avec Apply/OK.

- ❑ **Procéder** comme précédemment mais cette fois pour le **bouton poussoir BAS**, à associer avec l'entrée constante **CD**.




Il ne reste plus qu'a développer notre application. Dans le "Chart" nous aurons 3 états : "**ARRET**", "**MONTER_STORE**" et "**DESCENDRE_STORE**". Umot (en Volt) est la tension de commande du moteur électrique. Une tension de moteur Umot=-12V fermera le store banne alors qu'une tension Umot=12V l'ouvrira.

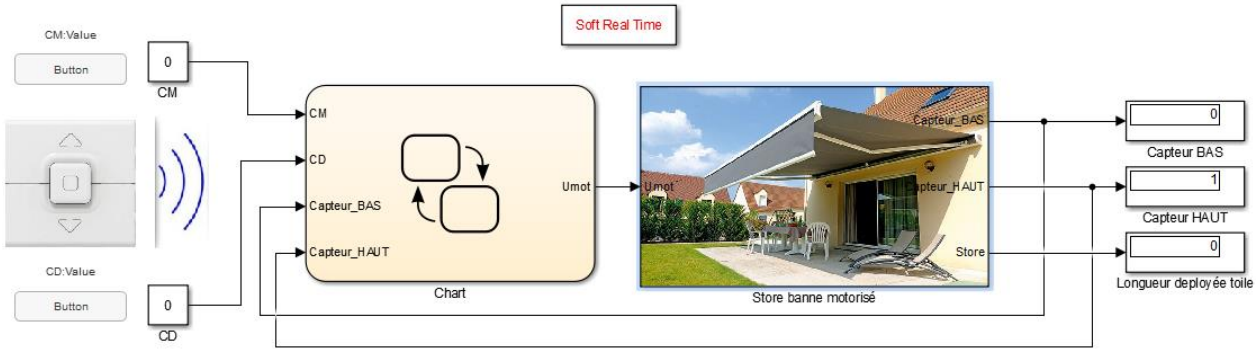
- ❑ **En double cliquant** sur le "Chart", vous ouvrez le diagramme d'état. Vous devez obtenir ceci :




- ❑ **Compléter** les valeurs de Umot (sans indiquer l'unité) dans les états « MONTER_STORE » et « DESCENDRE_STORE ».

- ❑ **Dessiner** les transitions ; puis **inscrire les conditions** que l'on souhaite pour chacune des transitions. *Par exemple : CM==1, entre crochets.*

Avant de tester votre propre solution, vérifier que vous n'avez pas oublié de connecter sur votre « Chart », en entrée, les signaux issus des capteurs de sortie *(Capteur_BAS, Capteur_HAUT)* du bloc « Store banne motorisé ».

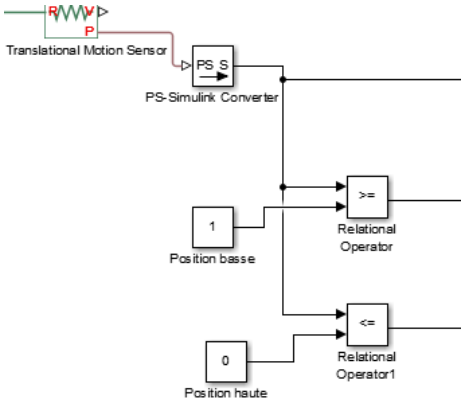



- **Effectuer** une simulation sur 200s. **Vérifier** si le résultat obtenu correspond au comportement attendu.

### Réglage de la longueur de toile déroulée.

Il est possible de définir la course maximale du store, en mètre. Pour cela, il suffit juste de changer la valeur présente dans le bloc **« Position basse »,** accessible depuis le contenu du bloc « Store banne motorisée ».

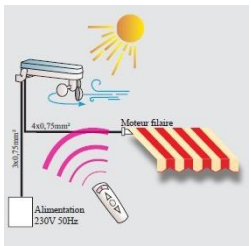
*Dans l'exemple ci-contre, le capteur bas sera actif lorsque la toile se déroulera sur une longueur supérieure ou égale à 1 mètre.*




Une évolution possible consisterait à concevoir une solution qui, lorsqu'on appuie sur un bouton poussoir, permet d'obtenir un arrêt du store dans une position intermédiaire. Dans le « Chart », nous aurions par exemple un état supplémentaire « ARRET_INTER ». A vous de développer votre application à partir du diagramme précédent.

## 5. Modéliser le comportement du store en fonction des conditions climatiques

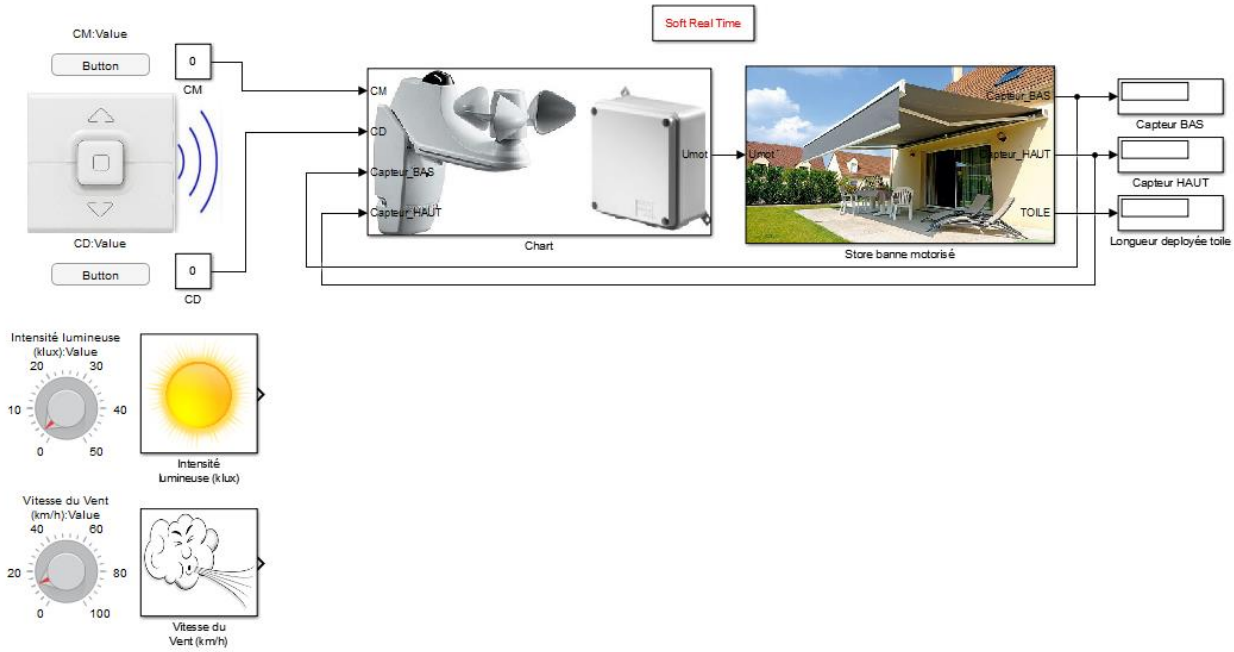
Lorsqu'on choisit une **commande électrique motorisée** pour un **store banne d'extérieur**, il est possible d'ajouter des options de confort. Ces options ne sont pas de simples gadgets, ils permettent d'**automatiser le store d'extérieur** en fonction des conditions météorologiques : soleil, vent, pluie, chaleur, froid. Ils garantissent un confort supplémentaire et une durée de vie plus importante au store.




On peut trouver des équipements qui réalisent plusieurs fonctionnalités en un seul boîtier. Ci-contre, c'est un module qui mesure la vitesse du vent et l'intensité lumineuse.

Comportement attendu: si l'intensité lumineuse est supérieure à un seuil fixé (20 000 lux par exemple) on descend automatiquement le store. Si le vent est trop important, supérieur à un seuil (25km/h par exemple), on remonte automatiquement le store.

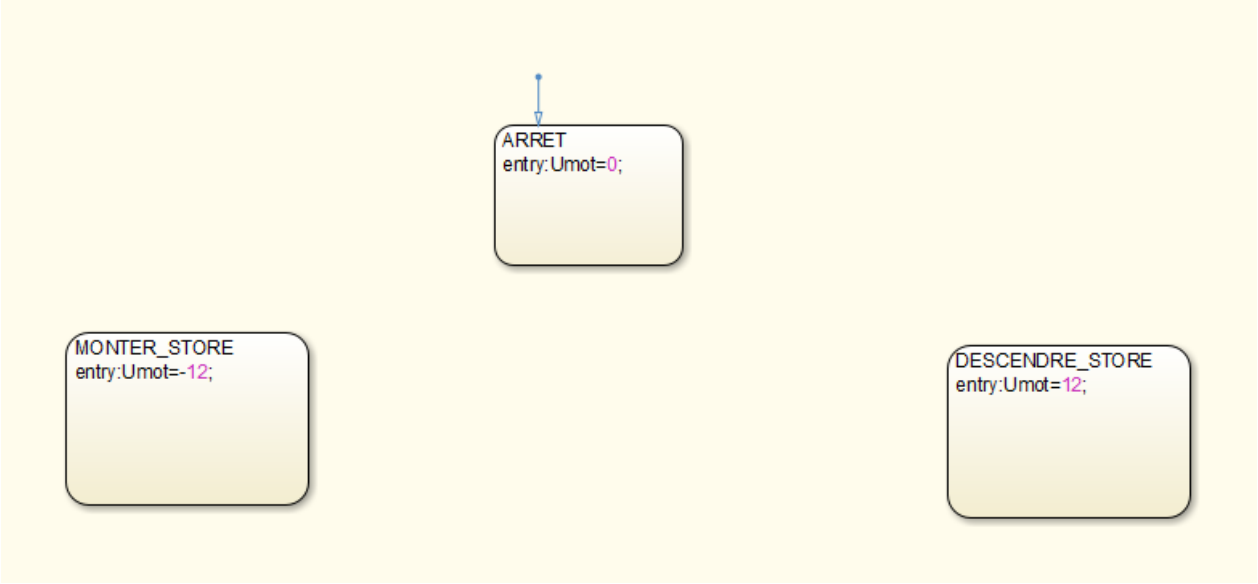
- **Ouvrir** le fichier "Store4meteo.slx". Vous devez obtenir sur votre écran une fenêtre Simulink conforme à l'image ci-dessous.




Vous pouvez remarquer que sur ce modèle Simulink, les deux boutons rotatifs qui permettent de simuler des « conditions météorologiques » ne sont pas connectés au « Chart ». De plus, on notera qu'aucune entrée n'est déclarée afin de récupérer les valeurs de ces grandeurs physiques dans le diagramme d'état.

Il ne reste plus qu'à développer votre application. Le diagramme d'état élaboré lors du chapitre précédent servira de point de départ.

- **En double cliquant** sur le "Chart", vous ouvrez le diagramme d'état. Vous devez obtenir ceci :




- **Dessiner** les transitions; puis **inscrire les conditions** que l'on souhaite pour chacune des transitions. *Vous devrez utiliser deux variables supplémentaires, relatives à l'intensité solaire reçue et à la vitesse du vent.*

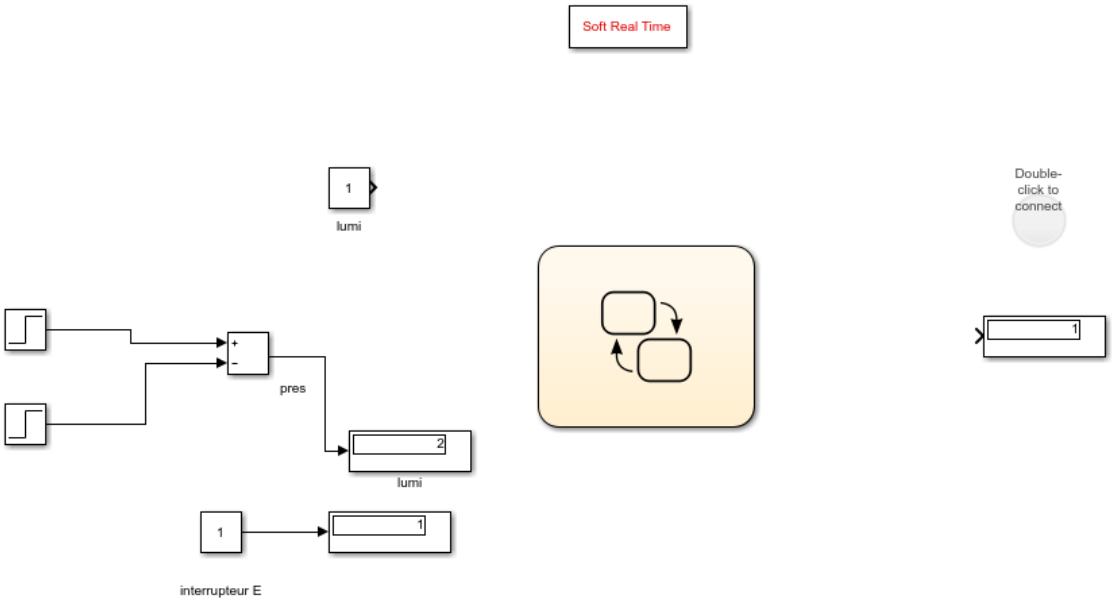
- **Déclarer** dans le « model explorer » les nouvelles variables utilisées comme des **entrées**.

- **Effectuer** une simulation sur 200s. **Vérifier**, en faisant varier les « paramètres météorologiques », si le résultat obtenu correspond au comportement attendu.


## 6 Modéliser la commande d'un éclairage extérieur

- ❑ **Regarder** la vidéo tutorielstateflow.mp4
- ❑ **Compléter** le fichier detecteurlum0 afin qu'il soit conforme au premier schéma (sans l'interrupteur)

La luminosité et la présence sont modélisés par deux entrées, l'éclairage est modélisé par une lampe qu'il faudra connecter et paramétrer ( éteint : rouge , allumé : vert ).




- ❑ **Modifier** ensuite le schéma pour qu'il reproduise le fonctionnement complet (avec interrupteur)

Nommer le fichier detecteurlum1.

Vous savez à présent **concevoir** un diagramme d'état, **l'associer avec un modèle Simscape** afin de **modéliser et simuler** le comportement d'un système. La prochaine étape consiste à découvrir comment **développer une application** que l'on **implantera et exécutera** sur une cible matérielle, plateforme Arduino par exemple.