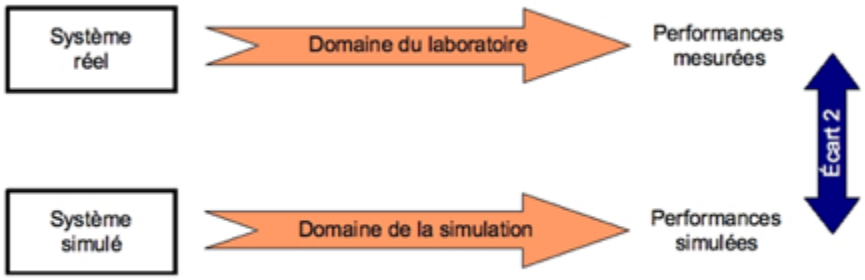
Lycée Jules Ferry Versailles	**Diagrammes d'états/transitions** ***Programmation sur Cible ARDUINO*** **Portail DRAGON**	S-SI
S si		***TP***

Noms : _____
Prénoms : _____
Classe : _____
Date : _____

Note : /20

1. Objectifs :

Nous nous intéresserons plus particulièrement à l'écart entre le système réel et le système simulé.



2. Critères d'évaluation et barème :

Autonomie, quantité et qualité du travail, soin...	/3

Etude du modèle physique	/5
Modélisation du mode manuel	/5
Programmation du portail réel	/5
Modélisation du mode automatique	/2

3. Matériel nécessaire :

- Système Portail Dragon du laboratoire
- Logiciel MATLAB + Simulink

4. Problème technique :

Le portail dragon est un équipement domestique permettant d'ouvrir et de fermer un passage.

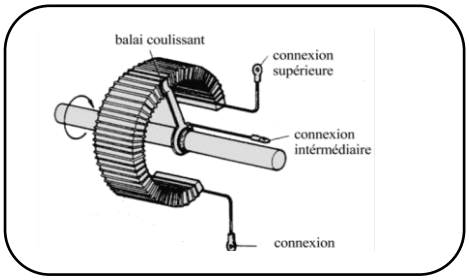
Afin de réduire les coûts, le constructeur désire changer le microcontrôleur existant par un ATMEL Atmega2560.
En phase de prototypage, le portail a été équipé de la carte de prototypage Arduino MEGA 2560 qui dispose du microcontrôleur Atmega2560.

Les objectifs de ce TP sont :

- Réaliser l'algorithme de programmation en graphe d'états / transition
- Simuler l'algorithme avec Simulink et Stateflow (Matlab)
- Implanter le graphe d'état compilé dans la carte de prototypage Arduino MEGA 2560 afin de tester réellement l'algorithme.

5. Entrées/Sorties du diagramme :

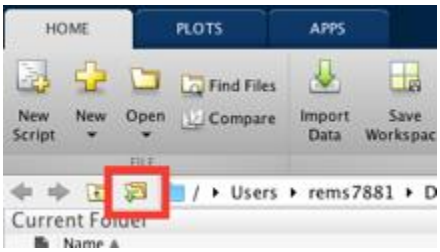
Remarque : Un capteur analogique (potentiomètre vis) de position angulaire a été placé sur la vis du réducteur. Ce capteur permet de déterminer la position du portail



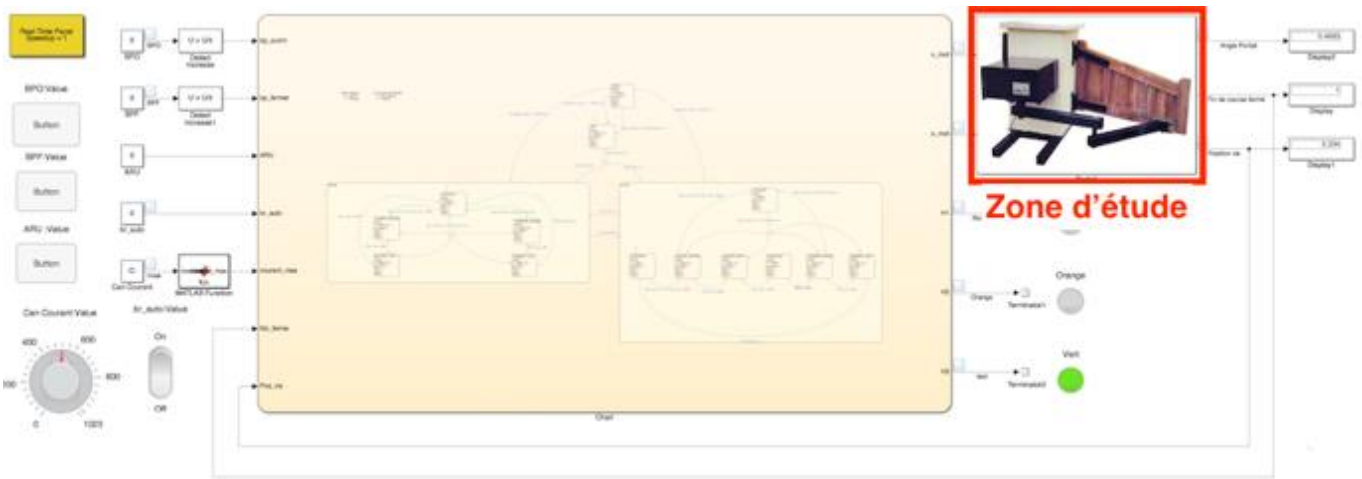
Variable	Description	Etat 0	Etat 1	Analogique
ARU	Arrêt d'urgence	Relâché	Enfoncé	
bp_ouvrir	Bouton poussoir ouvrir	Relâché	Enfoncé	
bp_fermer	Bouton poussoir fermer	Relâché	Enfoncé	
br_auto	Bouton rotatif manu/auto	Manuel	Auto	
fdc_ferme	Fin de course fermé	Non fermé	Fermé	
pos_vis	Potentiomètre vis			0 : portail fermé 1000 : portail ouvert
courant_max	Détection courant maximum	Courant normal	Courant maxi dépassé	
v_mot	Vitesse moteur			0 : arrêt 100 : vitesse lente 255 : vitesse max
s_mot	Sens moteur	Ouvrir	Fermer	
H1	Voyant rouge	Voyant allumé	Voyant éteint	
H2	Voyant orange	Voyant allumé	Voyant éteint	
H3	Voyant vert	Voyant allumé	Voyant éteint	

6. Etude du modèle physique :

- Télécharger le fichier « Modele_portail.zip » et le décompresser dans le dossier téléchargement.
- Démarrer MATLAB , et se placer dans le dossier « **Modele_portail/1_SIMU** » :
- Double-cliquer le fichier « **PORTAIL_DRAGON_ELEVE_VXX.slx** »



6.1. Modélisation de la carte de puissance :



La carte de puissance MD03 permet de distribuer l'énergie électrique en fonction des ordres de la chaîne d'information. Elle permet de faire varier la vitesse du moteur et son sens de rotation.

Elle est pilotée par la sortie MLI D8 de l'Arduino proportionnellement à un mot numérique **v_mot** de 8 bits :

- Lorsque les 8 bits sont à 0, la tension de sortie de la carte est de 0V.
- Lorsque les 8 bits sont à 1, la tension de sortie de la carte est de 12V.

On considère que la vitesse de rotation du moteur est proportionnelle à la tension de sortie de la carte, et que pour une tension de 12V, ω_{mot} = 200 rd/s.

Enfin, on considère que si le moteur tourne dans le sens positif, le portail s'ouvre.

Q1. *Donner la valeur décimale du mot numérique v_mot correspondant à une tension de sortie de 0V et de 12V.*

Q2. *Exprimer alors ω_{mot} en fonction de v_mot.*

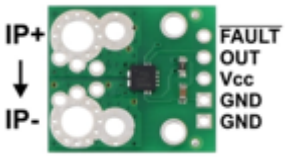
Q3. *Reprendre sur feuille de copie le code ci-dessous afin de modéliser la carte de puissance en tenant compte de l'ordre du sens de rotation du moteur **s_mot** (voir tableau page 2) :*

```
if s_mot == 0
    w_mot =.....;
else
    w_mot =.....;
end
```

- Double-cliquer sur le bloc « **Portail** » puis sur le bloc « **Commande moteur** ».
- Modifier le code d'après la question 3 et sauvegarder le bloc.

6.2. Modélisation capteur de courant :

Afin de détecter le blocage du portail par un obstacle, un capteur de courant a été installé sur l'alimentation de la carte de puissance du moteur.
Il renvoie une tension analogique de sortie proportionnelle au courant mesuré.



*On considère qu'un blocage correspond à un courant **dépassant 4A**.*

On donne l'équation caractéristique du capteur : $V_{courant}$ = 0.136*I+2.5, avec :

- $V_{courant}$: tension analogique de sortie du capteur en V
- I : courant mesuré en A (gamme entre -18A et 18A)

Q4. *Sur feuille de copie, tracer le graphique $V_{courant}$ = f(I).*

Q5. Donner la valeur de $V_{courant}$ pour les valeurs extrêmes de I, ainsi que pour I = 0A.

Le capteur de courant est branché sur un convertisseur analogique / numérique de la carte Arduino :

- 10 bits
- Pleine échelle : 5V

Q6. Montrer que le mot numérique décimal de sortie du CAN $N_{courant} = 27.82*I+511.5$

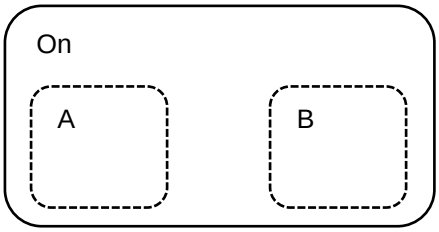
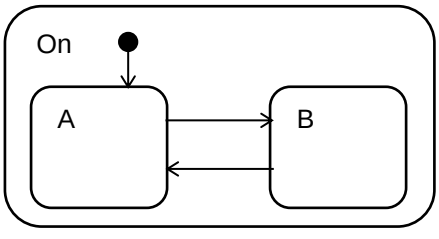
Q7. Calculer alors $N_{courant}$ pour un courant correspondant à un blocage du portail.

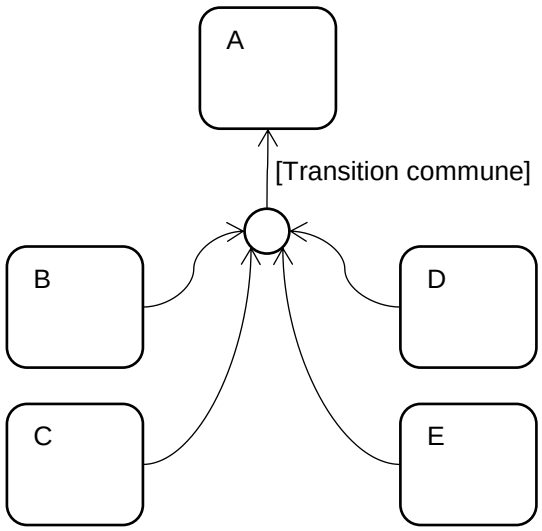
Q8. Reprendre sur feuille de copie le code ci-dessous afin de modéliser le seuil de courant maximum (voir tableau page 2).

```
if N_courant >.....
    courant_max = ..... ;
else
    courant_max = ..... ;
end
```

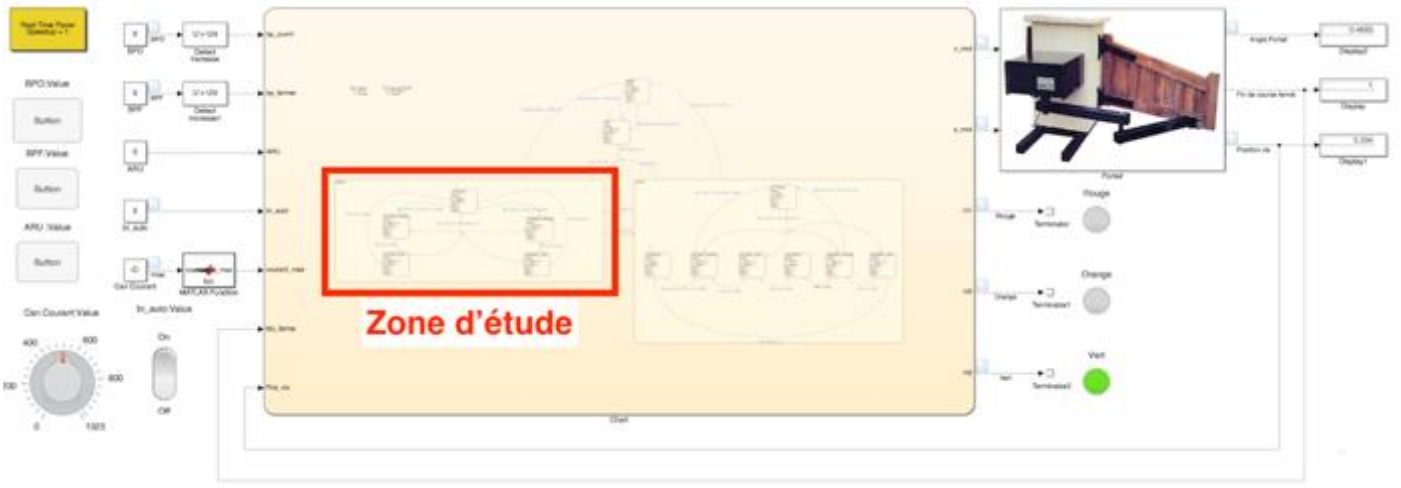
- Double-cliquer sur le bloc « **Courant max** ».
- Modifier le code d'après la question 8 et sauvegarder le bloc.

7. Rappels STATEFLOW :

Algorithme	Syntaxe STATEFLOW
Vérifier que la variable v est **égale** à 1	[v==1]
Vérifier que la variable v est **supérieure ou égale** à 1	[v>=1]
Vérifier que la variable v est **supérieure** à 1	[v>1]
Vérifier que la variable v est **inférieure ou égale** à 1	[v<=1]
Vérifier que la variable v est **inférieure** à 1	[v<1]
Vérifier que l'état A est **actif**	[in(A)]
La transition est validée si la condition1 **ET** la condition2 sont vérifiées	[Condition1 & Condition2] →
La transition est validée si la condition1 **OU** la condition2 sont vérifiées	[Condition1 | Condition2] →
Dans l'état On, Les états A et B s'activent en **parallèle (A ET B)**	
Dans l'état On, Les états A et B s'activent en **de façon exclusive (A OU B)** en fonction des transitions. Par défaut le premier état à s'activer est l'état A.	

Algorithme	Syntaxe STATEFLOW
Plusieurs états connectés à une **transition commune** :	

8. Modélisation du mode « Manuel » :



8.1. Cahier des charges :

En mode manuel, le portail :

- Attend l'ordre d'ouverture ou de fermeture du portail (bouton poussoir ouvrir ou fermer)
- S'ouvre si la position de la vis est inférieure à 1000
- Se ferme si le contact fin de course fermé n'est pas activé
- Lors de l'ouverture : passe en vitesse lente lorsque la position de la vis est supérieure à 800
- Lors de la fermeture : passe en vitesse lente lorsque la position de la vis est inférieure à 100

En phase d'ouverture ou de fermeture, un appui sur le bouton poussoir ouvrir ou fermer provoque l'arrêt du portail
Seul le voyant vert est allumé.

8.2. Diagramme d'état :

Variables de transition :	bp_ouvrir bp_fermer fdc_ferme pos_vis
Etats :	ATTENTE entry: v_mot=…..; s_mot=…..; H1=…..; H2=…..; H3=…..;
	OUVRIR_RAPIDE entry: v_mot=…..; s_mot=…..; H1=…..; H2=…..; H3=…..;
	OUVRIR_LENT entry: v_mot=…..; s_mot=…..; H1=…..; H2=…..; H3=…..;
	FERMER_RAPIDE entry: v_mot=…..; s_mot=…..; H1=…..; H2=…..; H3=…..;
	FERMER_LENT entry: v_mot=…..; s_mot=…..; H1=…..; H2=…..; H3=…..;
Transition par défaut :	● ↓

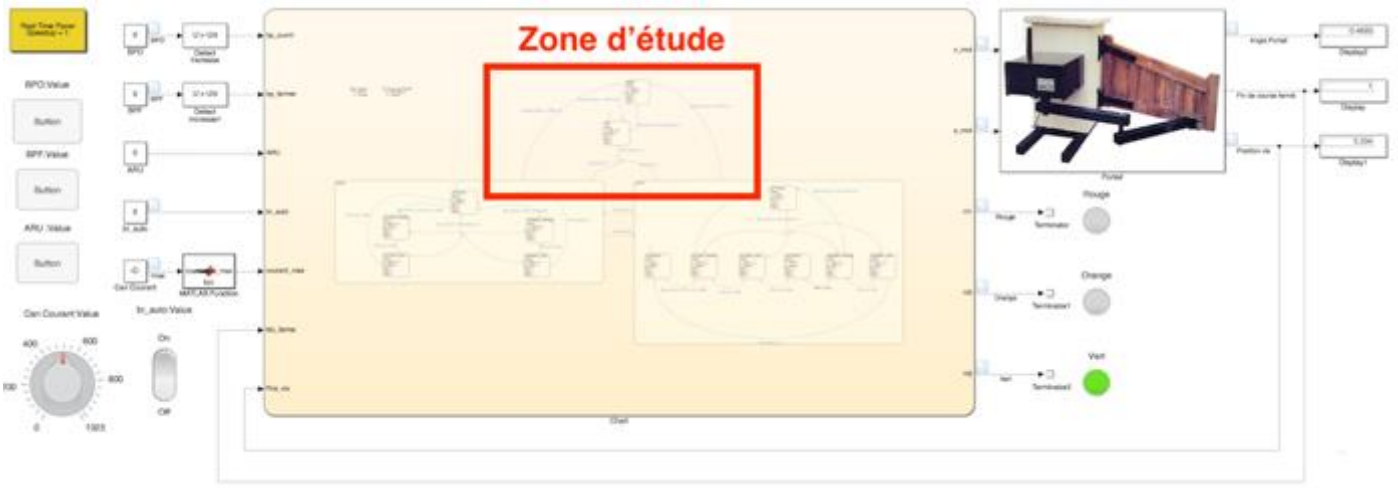
Le comportement des variables est décrit sur le **tableau de la page 2**.

Q9. *Proposer sur feuille de copie un diagramme répondant au cahier des charges du mode manuel.*

Q10. *Implémenter votre solution, effectuer une simulation (temps infini) et compléter sur feuille de copie la check-list ci-dessous :*

Critères		Validé	Non validé
Phase d'ouverture	Lors de l'appui sur le bouton ouvrir, le portail s'ouvre en vitesse rapide		
	Le portail passe en vitesse lente lorsque la position de la vis dépasse 800		
Phase de fermeture	Lors de l'appui sur le bouton fermer, le portail se ferme en vitesse rapide		
	Le portail passe en vitesse lente lorsque la position de la vis passe en dessous de 100		
En phase de mouvement, un appui sur l'un des boutons poussoir provoque l'arrêt du portail			
Le voyant vert est allumé			

9. Modélisation des modes « Sécurité » et « Initialisation » :



9.1. Cahier des charges :

9.1.1. Mode initialisation :

A l'allumage du portail, le portail entre dans le mode initialisation, qui ferme le portail en vitesse lente jusqu'à la détection du contact fin de course fermé.
Le voyant orange est allumé.
Le système entre ensuite dans le mode sélectionné (auto ou manuel).

9.1.2. Mode sécurité :

Afin de garantir la sécurité des utilisateurs, le constructeur décide de réaliser une détection de surintensité, qui permet de détecter le blocage du portail par un obstacle.
Le système est par ailleurs équipé d'un bouton « coup de poing » d'arrêt d'urgence, dont l'activation a le même effet que la détection de surintensité.

Dans ces deux cas, le portail se met en mode « sécurité », le moteur est arrêté et le voyant rouge est allumé.
Un appui sur le bouton ouvrir ou fermer déclenchera le mode initialisation.

9.2. Diagramme d'état :

Q11. Proposer sur feuille de copie un diagramme répondant au cahier des charges des modes initialisation et sécurité.

Q12. Implémenter votre solution, effectuer une simulation (temps infini) et compléter sur feuille de copie la check-list ci-dessous :

Critères		Validé	Non validé
Mode sécurité	Le bouton arrêt d'urgence déclenche le mode sécurité		
	Une surintensité déclenche le mode sécurité		
	Un appui sur l'un des boutons poussoir provoque le passage en mode initialisation		
	Le voyant rouge est allumé		
Mode initialisation	Le portail se ferme en vitesse lente		
	Le voyant orange est allumé		

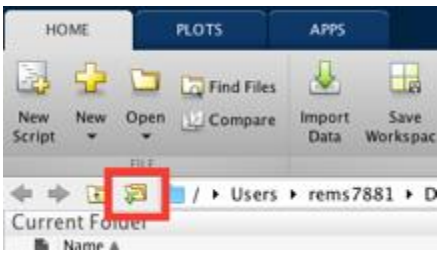
10. Programmation du portail réel :

10.1. Préliminaires :

On se propose à présent de tester le diagramme d'état de commande sur le système réel.
Pour cela, nous allons compiler le diagramme et l'exécuter sur la cible Arduino du portail (fonction de Simulink).

- Sous Stateflow, presser les touches « **CTRL + A** » afin de sélectionner tous les éléments du diagramme.
- Puis presser « **CTRL + C** » pour copier :

- Se placer dans le dossier « **Modele_portail/2_ARDUINO_EXTERNAL** » :

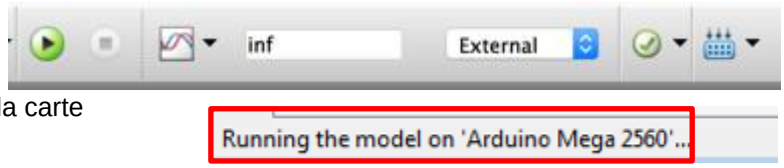


- Renommer le fichier le fichier « **PORTAIL_DRAGON_ARDUINO_ELEVE_VXX.slx** » en y rajoutant « **NOM1_NOM2** » en entête
- Double-cliquer le fichier « **NOM1_NOM2_PORTAIL_DRAGON_ARDUINO_ELEVE_VXX.slx** »
- Ouvrir le diagramme Stateflow et coller (**CTRL + V**) les éléments précédemment copiés.
- **Sauvegarder et copier le fichier dans l'emplacement indiqué par votre professeur.**

10.2. Exécution du programme sur cible (sur le PC connecté au Portail) :

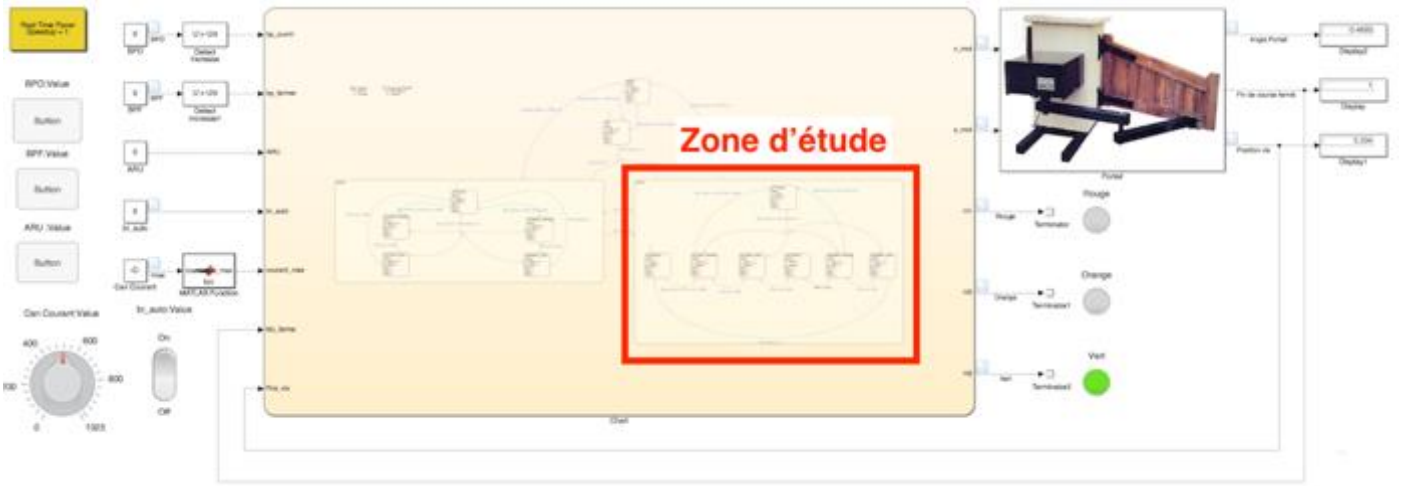
Rappel : le modèle à exécuter en mode « External » doit être stocké sur un dossier local (pas de dossier réseau).

- Ouvrir le fichier « **NOM1_NOM2_PORTAIL_DRAGON_ARDUINO_ELEVE_VXX.slx** »
- Vérifier les paramètres : temps : **inf**, mode : **External**
- Lancer le modèle et *patienter 1 min* afin que le programme se compile et tourne sur la carte Arduino :



Q13. Vérifier le bon fonctionnement du portail réel en reprenant les check-lists précédemment réalisées (ajouter une colonne « Réel »).

11. Modélisation du mode « Automatique » :



11.1. Cahier des charges :

Le mode automatique reprend le **fonctionnement de base du mode manuel**.
Lorsque la phase d'ouverture est terminée, le système patiente 15 secondes avant de lancer le cycle de fermeture.

11.2. Diagramme d'état :

Q14. Proposer sur feuille de copie un diagramme répondant au cahier des charges du mode automatique.