|  |  |  |
| --- | --- | --- |
| **Sciences de**  **l’Ingénieur** | **Compétences travaillées :**  - Évaluer une solution  - Matérialiser une solution virtuelle | **Connaissances associées :**   * Mesures et tests des performances de tout ou partie de la solution innovante   - Mise en œuvre d’outils de prototypage rapide  - Prototypage de la commande |
| **Durée :** 2 h |
| Référence :  Hydrao\_Solution\_3 |
| **Matériel :**  Système didactique HYDRAO DIDACT |
| **Logiciel :**  Matlab Stateflow |
| **Documents :**  Dossier technique du système |
| **Pré-requis :**  - Eléments sur les diagrammes SysML  - Eléments sur les machines à états SysML  - Premiers éléments sur le prototypage rapide sur carte cible | |

|  |  |  |
| --- | --- | --- |
| *Conseil professeur sur les durées* | Question | Durée indicative (min) |
| Q1 | 1 | 5 |
| Q2 | 2.1 | 10 |
| 2.2 | 15 |
| 2.3 | 10 |
|  | Sous total : 35 |
| Q3 | 3.1 | 5 |
| 3.2 | 10 |
|  | Sous total : 15 |
| Q4 | 4.1 | 15 |
| 4.2 | 10 |
| 4.3 | 15 |
|  | Sous total : 40 |
| Q5 | 5.1 | 5 |
| 5.2 | 10 |
|  |  | Sous-total : 15 |
| Consultations annexes |  | 10 |

# Problème scientifique et technologique :

Le système de douche intelligent imaginé par Hydrao pour réduire le volume d’eau consommé lors d’une douche, et conséquemment l’énergie nécessaire pour chauffer cette eau, repose à la fois sur une limitation du débit d’eau, et l’implication de l’utilisateur en l’informant de ses consommations.

A un moment donné du développement, la validation du concept nécessite la réalisation d’une maquette afin de vérifier tout ou partie du fonctionnement final attendu. Pour cela, des éléments standards (déjà existants) associés à des éléments spécifiques au produit (le moins possible à cette étape) sont assemblés pour prototyper le plus rapidement possible la maquette.

On se propose dans cette troisième activité (Hydrao\_Solution\_3) de prototyper rapidement une partie du traitement dans une carte cible à microcontrôleur (qui n’est pas la carte définitive du produit) en implémentant directement le diagramme états-transitions de la machine d’état « technologique » retenue pour l’écriture du programme (cf. annexe 1). Ce diagramme correspond sensiblement au diagramme technologique SysML (page 6 du fichier Hydrao\_SysML.pdf regroupant l’ensemble des diagrammes) avec quelques modifications pour simplifier la programmation.

Cette activité s'appuie sur une démarche innovante appelée ***Ingénierie des systèmes basée sur le modèle*** (en anglais ***Model-Based Systems Engineering*** ou MBSE). Pour cela vous exploiterez des fonctionnalités avancées du logiciel Matlab-simulink-stateflow.

Documents disponibles :

* Le dossier technique du système incluant notamment les diagrammes SysML et le schéma synoptique du pommeau Aloé (fichier ***Hydrao\_SysML.pdf*** ). Si besoin, des informations sur le SysML figurent dans le fichier ***Ressource\_Elements\_sur\_Diagrammes\_SysML***.
* Le document ***Ressource\_Ingenierie\_des\_systemes\_basee\_sur\_le\_modele*** vous fournit des informations sur la démarche et vous pourrez l'exploiter dans les différentes phases du travail à effectuer.
* Le document ***Ressource\_implementation\_Simulink\_vers\_Arduino*** explicitant la mise en œuvre du prototypage rapide que vous aurez à effectuer ainsi que les matériels nécessaires.

|  |  |
| --- | --- |
| Mise en œuvre matérielle :   * Effectuez l'assemblage des cartes, le raccordement au générateur basse fréquence, le raccordement des broches A0 et D2.   Cf. doc Ressource\_implementation\_Simulink\_vers\_Arduino   * Raccordez la carte arduino au PC via le cordon USB |  |

|  |  |
| --- | --- |
| Travail à effectuer : Q1  Nota : Il conviendra de réaliser un dossier réponse documenté à l’aide de copies d’écran pertinentes si besoin. | Q3  Q5  Q4  Q2 |

## Exigences/spécifications à satisfaire en suivant la démarche MBSE :

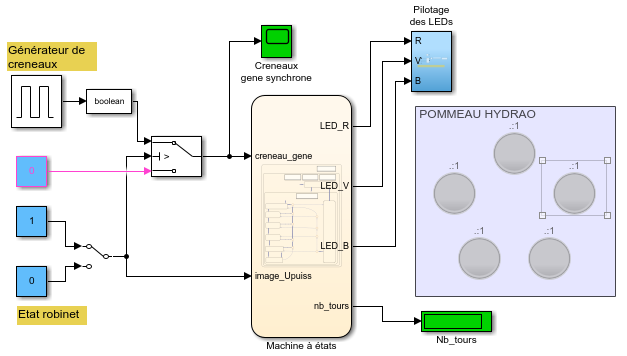
* A partir de la tension alternative fournie par la génératrice synchrone nous utilisons :
  + Les créneaux obtenus par mise en forme de la tension alternative. On a montré que la fréquence de ces créneaux est proportionnelle au débit d'eau dans la micro-turbine. Le système doit pouvoir mesurer le débit jusqu'à 20 litres par minute.
  + La tension image de la tension du générateur synchrone redressée et filtrée. Le système doit lancer l'éclairage des LEDs à partir d'une tension-image de Upuissance égale à 230 mV et passer en Pause ou en Stop si la tension-image tombe sous 210 mV.
* Le fonctionnement de la machine d'état intégrant la gestion de l'éclairage des LEDs doit correspondre au diagramme présent dans la description SysML.

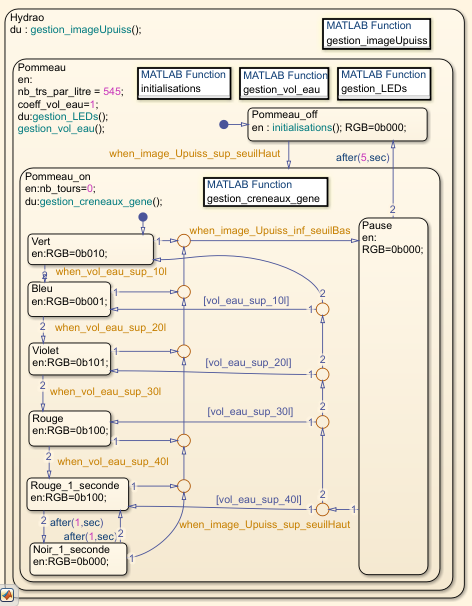
## 1ème partie : modèle multiphysique vérifié par simulation MIL (model in the loop) et PIL (processor in the loop)

Ouvrez dans Matlab-simulink le modèle ***STM\_MIL\_PIL\_base\_rxxx.slx*** où xxx désigne la version matlab.

Ce modèle fonctionne en simulation intégrale dans l'environnement Matlab du PC. Vous pouvez noter que la machine d'état "stateflow" a une structure identique à la machine d'état retenue pour la programmation, elle-même dérivée de la machine d'état SysML. Les différences sont dues au fait que certaines instructions ne sont pas implémentables directement dans Stateflow, d'où l'utilisation de fonctions Matlab. Ceci concerne en particulier la génération des évènements liés au volume d'eau mesuré et à la valeur de la tension-image de Upuissance.

Nota : Le commutateur "Etat robinet" permet de simuler l'ouverture (état 1) ou la fermeture (état 0) du robinet.





**Question 1 :**

A partir du débit maxi, calculez la fréquence maxi des créneaux de la génératrice que doit pouvoir mesurer le microcontrôleur.

Caractéristique turbine : 545 tours par litre

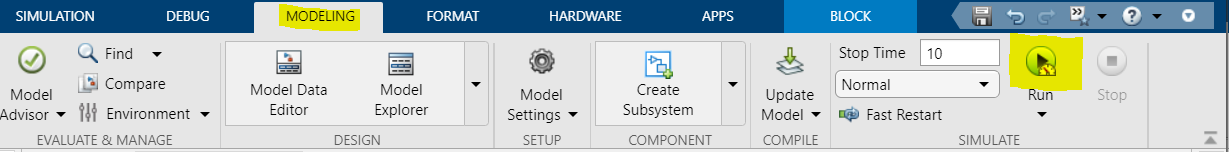
Caractéristique génératrice : 1 impulsion par tour

Débit maxi : 20 litres par minute soit 0,33 l/s

Fréquence maxi = 545 x 0,33 = 182 Hz

**Question 2 : simulation MIL (Model In the Loop)**

-2.1- Lancez l'exécution du modèle :



Vérifiez que le fonctionnement est conforme au fonctionnement attendu lors des différentes phases : séquence Vert, Bleu, Violet, etc., séquence fermeture robinet pour savonnage, séquence fermeture robinet en fin de douche et réouverture. Explicitez votre réponse.

* Séquence "normale" : le robinet étant fermé (commutateur en position 0) le diagramme est à l'état "Pommeau\_off" (état "Pommeau\_off" actif) et les LED sont éteintes. Après ouverture du robinet (commutateur en position 1) le diagramme passe à l'état "Pommeau\_on" (état "Pommeau\_off" inactif et état "Pommeau\_on" actif), ce qui entraine l'activation successive des états : Vert, Bleu, Violet, etc., et l’éclairage des LEDs correspondantes.
* Séquence "savonnage" : on ferme le robinet => l'état Pause est activé et les LEDs s'éteignent. Si on rouvre le robinet avant la fin de la temporisation de 5s (2 minutes en réalité) la séquence "normale" reprend à partir de l'état quitté à la fermeture du robinet.
* Séquence "fin de douche" : idem séquence "savonnage" mais, si le robinet n'est pas rouvert avant la fin de la temporisation, le diagramme passe à l'état "Pommeau-off". Si on rouvre le robinet on repasse à "Pommeau\_on" en partant de l'état Vert (situation intiale).
* Conclusion : on a bien le fonctionnement attendu.

-2.2- Vérifiez que la mesure du nombre de tours de la turbine est exacte. Justifiez votre réponse en précisant votre méthode. Expliquez comment est obtenue la valeur du nombre de tours.

Exemple de méthode :

* La fréquence des créneaux géné est de 100 Hz (période = 0.01s à noter dans le bloc "Pulse generator")
* Placer le commutateur "Etat robinet" sur 1.
* Régler la durée de simulation sur 10s (valeur "Stop Time" = 10 dans la fenêtre "SIMULATION".
* Exécuter la simulation (Run)
* A la fin de l'exécution on observe que le nombre de tours (afficheur "Nb\_tours") vaut 1000 donc conforme à la valeur attendue = fréquence x durée = 100 x 10.
* Obtention de la valeur du nombre de tours :
* L'entrée "creneau\_gene" de la machine d'état reçoit des créneaux binaires issus du générateur de créneaux.
* Dans l'état "Pommeau\_on" la variable "signal-gene" prend la valeur de "creneau\_gene" (cf. fonction Matlab "gestion\_creneaux\_gene")

function gestion\_creneaux\_gene

signal\_gene = creneau\_gene;

* à chaque passage de l'état 0 à 1 de la variable "signal\_gene", on incrémente la variable "nb\_tours"

function gestion\_vol\_eau

%Mesure nombre de tours turbine

if (~signal\_gene\_prec)&&(signal\_gene)

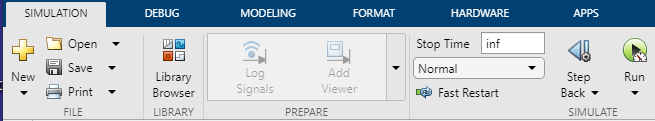
nb\_tours = nb\_tours + 1;

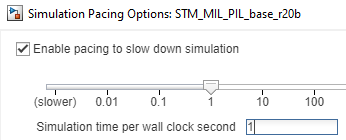
end

signal\_gene\_prec = signal\_gene;

-2.3- Limites de fonctionnement du modèle :

Le modèle est configuré pour exécuter la simulation en temps réel grâce au paramétrage du "Run" en mode "pacing" et en appliquant une valeur 1 au coefficient de déroulement temporel de la simulation.





Lorsque la limite de la puissance de calcul du PC est atteinte, la simulation en langage "matlab" ne s'effectue plus en temps réel et le temps de simulation devient supérieur au temps de simulation affiché.

Réglez le temps de simulation à 10s et, en utilisant un chronomètre, vérifiez à partir de quelle fréquence de créneaux géné la simulation ne s'exécute plus en temps réel. Utilisez de préférence une approche par dichotomie en partant de 10 kHz puis 100 kHz etc.

La limite dépend de la puissance du PC. La limite de la fréquence des créneaux se situe en principe entre 10 kHz et 100 kHz (par exemple 50 kHz avec un "core i5")

**Question 3 : simulation PIL (Processor In the Loop)**

Vous allez tester le fonctionnement du modèle codé par Matlab en langage C reconnu par le langage arduino et exécuté par le microcontrôleur de la carte : il s'agit d'une simulation PIL. Pour réaliser cette simulation, il faut effectuer l'exécution du modèle en mode "RUN ON HARWARE" en cliquant sur "Monitor & Tune" dans l’onglet hardware.

Nota : ce modèle n'utilise pas les entrées-sorties de la carte arduino donc on pourrait utiliser une carte seule simplement raccordée au PC par le cordon USB.

-3.1- Réglez la fréquence des créneaux à la valeur maxi déterminée à la question 1 et effectuez le même travail qu'à la question 2.1. Donnez votre conclusion.

A 182 Hz, c’est-à-dire avec une période de 5,5 ms, le fonctionnement est correct avec respect du temps réel.

-3.2- Augmentez la fréquence des créneaux géné de façon à déterminer le seuil de fonctionnement "temps réel" de la carte arduino. Commentez le résultat.

La limite de la fréquence des créneaux pour un fonctionnement temps réel se situe à environ 2 kHz. Cette valeur est beaucoup plus faible qu'en simulation MIL car le processeur de la carte arduino est beaucoup moins puissant que celui du PC.

## 2ème partie : déploiement complet sur carte arduino

Dans la 1ère partie vous avez testé la partie logicielle du projet, c'est-à-dire le code généré par Matlab-Simulink, essentiellement pour vérifier que la machine d'état était opérationnelle. Dans cette partie vous allez compléter la mise au point en rajoutant la partie matérielle, c'est-à-dire les entrées-sorties de la carte arduino, et en testant l'ensemble.

**Question 4 : Run On Hardware** mode **Monitor & Tune**

Dans le modèle fourni en 1ère partie, ***STM\_MIL\_PIL\_base\_rxxx.slx***, il faut remplacer les entrées-sorties simulées (générateur de créneaux, commutateur "robinet", LEDs) par les entrées-sorties physiques de la carte arduino correspondant au schéma.

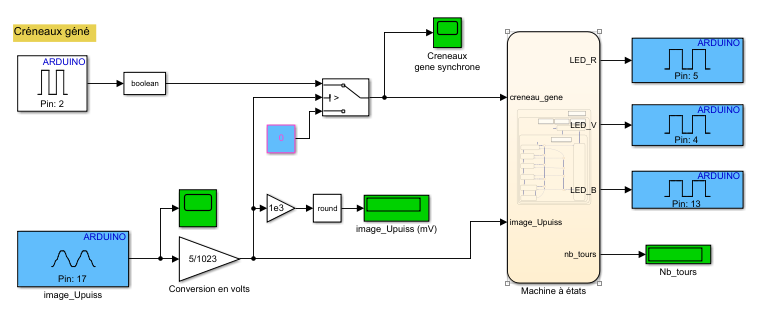
-4.1- Sachant que les créneaux-géné sont appliqués à l'entrée logique D2 de la carte arduino, la tension image-Upuiss à l'entrée analogique A3 et les LEDs Rouges Vertes Bleues respectivement aux sorties logiques D5, D4 et D13, en utilisant les objets de la bibliothèque "Simulink Support Package for Arduino", effectuez le remplacement des objets indiqués.

Nota 1 : L'acquisition de la tension image\_Upuiss s'effectue par une entrée analogique de la carte arduino, une conversion analogique-numérique sur 10 bits a ensuite lieu dans le microcontrôleur => lorsque la tension analogique varie de 0 à 5V on a 210 valeurs possibles de la valeur numérique en partant de 0. Il faut donc insérer entre le bloc d'entrée analogique et l'entrée image\_Upuiss du diagramme d'états un bloc gain fournissant la valeur de la tension en volts.

Nota 2 : le temps d'échantillonnage (Sample time) des blocs d'entrée doit être réglé à -1. Cela signifie qu'il sera automatiquement réglé par Matlab. Le pas de calcul, Cf. Onglet Modeling, Model Settings, Solver, Fixed-step size (fundamental sample time), doit être réglé à 1ms (1e-3).

Renommez votre modèle et enregistrez-le.

Corrigé modèle ***STM\_RunOnHardware\_cor1\_sans\_interrupt\_rxxx.slx*** (la partie "stateflow" est inchangée)

******

-4.2- L'environnement matériel étant en place, testez votre modèle en Run On Hardware mode Monitor & Tune. Vérifiez si le fonctionnement est "temps réel" à la fréquence déterminée à la question 1 puis recherchez la fréquence limite. Le pas de calcul étant imposé on observe que la limite est atteinte lorsque le comptage des créneaux n'est plus correct c’est-à-dire qu'on n'obtient pas le nombre de tours prévu au bout d'un temps donné,

A 182 Hz le fonctionnement est "temps réel". La fréquence limite se situe aux environs de 400 Hz.

**Amélioration des performances avec utilisation des interruptions externes**

On constate que la fréquence limite du fonctionnement "temps réel" avec la carte arduino est relativement basse, sachant par ailleurs que seule une partie du traitement est demandée dans notre prototypage, ne figure pas notamment la communication bluetooth. Il s'ensuit que si le traitement est plus long, cela abaisse la fréquence limite et on peut ne plus remplir le cahier des charges.

Nota : cela montre aussi une limite du prototypage rapide par génération automatique de code. Le code généré par matlab est loin d’être optimisé ; un codage direct dans un langage approprié permettrait une exécution plus rapide.

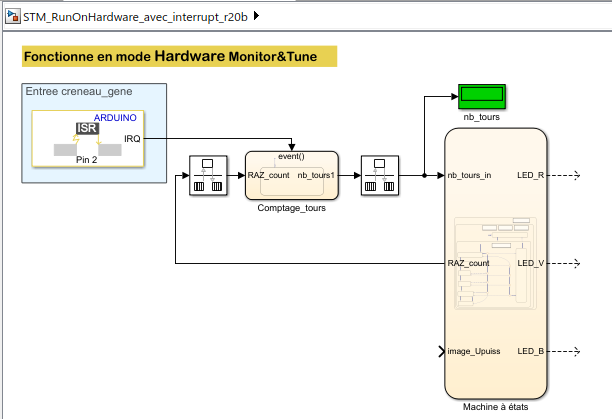
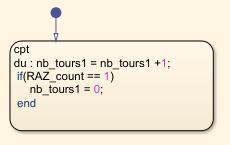
Le moyen classique pour obtenir un fonctionnement "temps réel", c'est-à-dire ici de ne manquer l'acquisition d'aucun tour, et donc de garantir un comptage correct, consiste à utiliser une "interruption".

La technique des "interruptions" sera étudiée en détail dans l'activité ***07\_Hydrao\_Programmation*** . Elle consiste à détecter qu'un "événement externe" a eu lieu sur une entrée logique, par exemple qu'elle est passée de 0 à 1 (rising) ou de 1 à 0 (falling). Il s'ensuit que l'exécution du programme "normal" est interrompue pour exécuter le programme associé à l'interruption, appelé ISR pour "Interruption sub-routine" dans Matlab. Dans notre cas, l'ISR incrémente le compteur de créneaux d'un pas. L’exécution de l’ISR étant terminée, l'exécution du programme normal se poursuit.

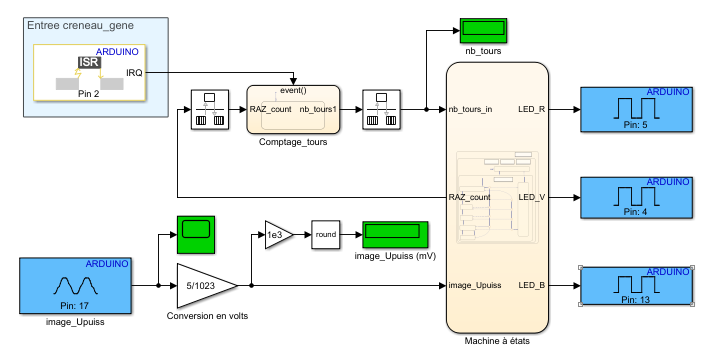
Nota : Les événements externes ayant leur cadence d'occurrence propre, dans Matlab cela nécessite l'utilisation de blocs "Rate transition" entre la partie ISR et le reste du modèle afin de permettre le transfert de données entre blocs ne fonctionnant pas de façon synchrone.

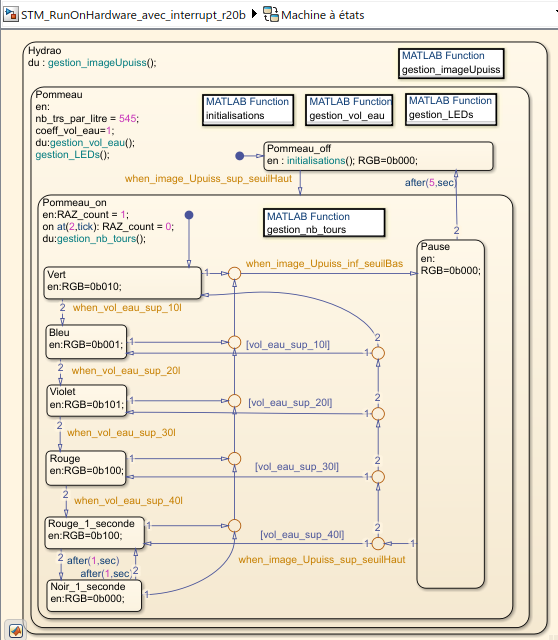
-4.3- Intégration du comptage des créneaux géné par interruptions

Ouvrez le modèle ***STM\_RunOnHardware\_avec\_interrupt\_r2xxx.slx***.



Corrigé modèle ***STM\_RunOnHardware\_cor2\_avec\_interrupt\_rxxx.slx***





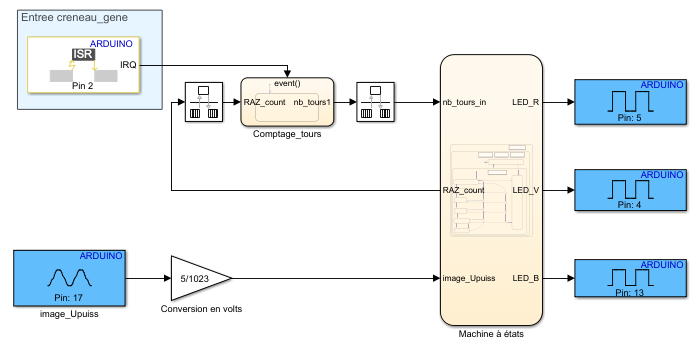
* A partir du modèle des questions précédentes, effectuez un copier-coller des circuits liés à l'entrée image-Upuiss et aux sorties LEDs de la machine à états.
* Testez votre modèle en Run On Hardware mode Monitor & Tune. Recherchez la fréquence limite de fonctionnement "temps réel" et commentez le résultat.

La fréquence limite se situe entre 5 et 10 kHz donc elle est au moins 10 fois supérieure à celle du mode d'acquisition "normal". Cela montre bien que l'utilisation des interruptions externes est tout à fait appropriée pour la détection d'événements brefs.

**Question 5 : Hardware** mode **Deploy Build Deploy & Start** (Standalone)

-5.1- Dans le modèle précédent, supprimez les éléments destinés à "monitorer" le fonctionnement en interaction avec le PC. Renommez votre modèle et enregistrez-le.

Corrigé modèle ***STM\_standalone1\_rxxx.slx***



Testez le modèle.

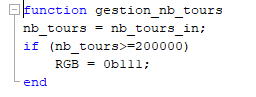
Nota : le fonctionnement de la carte arduino est maintenant en mode "standalone" (autonome), donc indépendant du PC. Toutefois il faut conserver la liaison USB pour alimenter la carte en 5V.

-5.2- Etudiez et testez un aménagement du modèle permettant de vérifier la limite de fonctionnement "temps réel" de la carte arduino.

Propositions de solutions : on rajoute une commande des LEDs en blanc lorsque le comptage atteint un certain seuil et on vérifie que, pour une fréquence de GBF donnée, le passage au blanc s'effectue bien au bout du temps prévu.

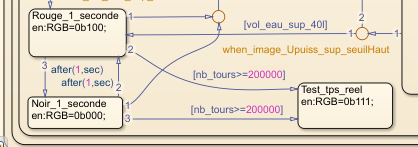
* Soit dans la fonction matlab "gestion\_nb\_tours"

Corrigé modèle ***STM\_standalone2\_avecRGB111\_ver1\_rxxx.slx***



* Soit en rajoutant un état "Test\_tps\_reel" en fin de séquence (moins élégant)

Corrigé modèle ***STM\_standalone2\_avecRGB111\_ver2\_rxxx.slx***



On peut vérifier que la limite est supérieure à 10 kHz

**Annexe 1 : diagramme états-transitions de la  
machine « technologique » retenu pour la programmation**

Ce diagramme états-transitions de la machine « technologique » retenu pour la programmation (python) a été conçu dans l’exercice 1 de Hydrao\_Solution\_2.

