|  |  |  |
| --- | --- | --- |
| **Sciences de**  **l’Ingénieur** | **Compétences travaillées :**  - Analyser le comportement d’un objet à partir d’une description à événements discrets  - Traduire le comportement attendu ou observé d’un objet  - Traduire un algorithme en un langage exécutable  - Documenter un programme informatique | **Connaissances associées :**  -Diagrammes états-transitions  - Algorithme  - comportement séquentiel,  - structures algorithmiques (variables, fonctions, structures séquentielles, itérations, répétitives, conditionnelles)  - Diagramme d’états-transitions  - langage de programmation  - commentaires de programmes |
| **Durée :** 2 séances de 1,5 h |
| Référence : Hydrao\_Prog2 |
| **Matériel :**  Système didactique HYDRAO DIDACT (cartes puissance, LED et carte d’évaluation STM32 Nucleo-WB55)  Générateur basse fréquence (GBF) |
| **Logiciels :**  PyScripter portable  Emulateur PuTTY |
| **Documents :**  Dossier technique du système |
| **Pré-requis :**  - Notions d’algorithme, de programme source  - Programmation du microcontrôleur STM32WB55  - Syntaxe de base du langage python | |

# Problème scientifique et technologique : traduire le comportement attendu du pommeau de douche

Afin de réduire la consommation d’eau et la consommation d’énergie (chauffage de l’eau) lors d’une douche, la startup Hydrao mise en partie sur une incitation ludique en fournissant une information instantanée sous forme lumineuse à l’utilisateur. On se propose ici de programmer en micropython la fonction de calcul du volume d’eau écoulé et la fonction de commande des LEDs.

**Conditions de réalisation de l’activité :**

Vous disposez des éléments suivants du système didactique HYDRAO DIDACT : carte électronique de puissance, carte LEDs, carte à microcontrôleur STM32 Nucleo WB55 avec un « firmware » micropython installé.

Vous disposez également d’un ordinateur PC, d’un logiciel « terminal de communication » avec la carte microcontrôleur « PuTTY », d’un logiciel éditeur de scripts « Portable Python » lancé par « Pyscripter-Launcher.exe » et d’une liaison physique par cordon USB type A vers USB mini-B.

Cette activité présente les principes d’une programmation d’une machine états-transitions SysML simple et son application au pommeau de douche.

On s’intéressera seulement au calcul du volume d’eau écoulé et à la commande des LEDS. Bien entendu, le programme proposé ici est fortement allégé par rapport au programme réel implanté dans le microcontrôleur du pommeau de douche et qui doit répondre à de nombreuses contraintes : temps réel, précision, échanges en Bluetooth.

Pour la mise en service, on pourra soit utiliser le système didactisé en reliant les sorties de la génératrice du pommeau à la carte de puissance (figure 1a), soit remplacer la génératrice par un générateur basse fréquence pouvant fournir un signal sinusoïdal d’au moins 10 V crête à crête (figure 1b). Ce générateur est à connecter au bornier bleu de la carte de puissance (bornes M1 et M2).

|  |  |
| --- | --- |
| ***a : utilisation du système didactisé*** | ***b : utilisation d’un GBF*** |

***Figure 1 : mise en œuvre du programme de gestion des LED***

# Travail à effectuer :

**Nota :** Un dossier réponse documenté à l’aide de copies d’écran pertinentes est à réaliser au fur et à mesure de la réalisation de l’activité.

**Présentation :**

Nous nous intéressons à l’exigence 1.1 **«*Afficher la consommation d’eau*»** du diagramme des exigences du SysML (cf. fichier ***Hydrao\_Aloe\_SysML.pdf*** du dossier technique du système).

Pour satisfaire cette exigence, un programme présent dans la carte électronique embarquée du pommeau doit commander l’éclairage des LED pour les paliers de consommations prédéfinis ci-dessous :



**Rappel :** Le volume d’eau écoulé est calculé à partir du nombre de tours effectués par la turbine.

Le diagramme SysML de définition des blocs donne la caractéristique nommée « **déplacement** » de la turbine, c’est-à-dire le nombre de tours qu’elle effectue par litre écoulé. On peut noter que le déplacement vaut **545 tours/litre**.

Cette activité est scindée en deux parties :

* Une première partie présente les principes de la programmation d’une machine à états à travers l’étude de l’exemple d’un circuit monostable.
* La seconde partie traite le cas de la programmation de la machine états transitions correspondant au fonctionnement du pommeau de douche Hydrao.

## Partie 1 : programmation d’un monostable

Avant d’entamer la programmation en micropython de la machine d’états du pommeau de douche, nous allons illustrer ce type de programmation à travers l’exemple d’un monostable.

Il s’agit d’un circuit électronique dont la sortie se trouve dans un état électrique stable (par exemple à l’état logique bas) quand il est au repos, et qui, lorsqu’il reçoit une impulsion de déclenchement (*start*), sa sortie bascule dans l’état électrique inverse (par exemple à l’état haut). Cet état est dit instable car la sortie retrouve son état initial au bout d’un « certain temps ».



***Figure 1 : schéma de principe d’un circuit monostable***

Les entrées / sorties de ce circuit sont :

**start** : c’est l’entrée qui fournit l’impulsion de déclenchement (ou impulsion d’entrée). Il s’agit de l’évènement qui conduit la sortie du monostable à changer d’état. Selon le choix technologique, il peut s’agir d’une courte impulsion ou d’un changement d’état logique (front montant ou descendant) sans retour à l’état initial.

**stop :** c’est l’entrée qui permet de remettre le monostable dans son état stable avant la fin de la durée du « certain temps ».

**clock :** c’est l’entrée servant de base de temps. La durée du « certain temps » est un multiple de la période du signal clock.

**s :** c’est la sortie du monostable. Elle peut être à l’état haut ou à l’état bas (état stable) et sur présence d’une impulsion de déclenchement, passer à l’état opposé, bas ou haut (état instable).

**Redéclenchable** ou **non redéclenchable** : un monostable non redéclenchable délivre une impulsion de sortie dont la durée est immuable, même si pendant l’état instable, surviennent plusieurs impulsions de déclenchement. Un monostable redéclenchable délivre une impulsion de sortie qui se « réactive » à chaque nouvelle impulsion de déclenchement.

Le document ressource « ***Ressource\_Programmation\_Machine\_Etats\_SysML\_V2022-06-19*** » donne la programmation d’un monostable non redéclenchable. L’algorithme et le programme python y sont détaillés.

**Travail demandé :**

**Question 1 :** A l’aide du document « ***Guide\_utilisation\_environnement\_de\_programmation\_micropython.docx*** », copiez le programme micropython dans le fichier *main* de *PyScripter-Launncher* et transférez-le dans la carte NUCLEOSTM32WB55. Testez le programme et visualisez les changements d’états de la sortie en fonction des appuis sur SW1 et SW2 et de la durée de l’état instable programmée. Vérifiez que le comportement est conforme à la machine d’état SysML.

**Question 2 :** Combien d’évènements comporte cette machine d’états ? Entourez dans l’algorithme et le programme python les noms de ces évènements.

**Question 3 :** Quelles sont les fonctions python utilisées pour le traitement de ces évènements ?

**Question 4 :** Combien d’états comporte cette machine d’états ?

## Partie 2 : Acquisition – gestion des LEDs et calcul du volume d’eau : mise en œuvre d’une solution utilisant une machine états-transitions

## 2.1. Solution technologique simple sans l’état « pause »

Le fonctionnement du pommeau de douche est décrit par la machine d’états-transitions partielle ci-dessous. Dans cette description, on ne tient pas compte de l’état « pause » dans lequel il y a interruption de l’écoulement de l’eau.

Vous disposez des diagrammes SysML et du schéma synoptique du pommeau Aloé (fichier « Hydrao\_Aloe\_SysML.pdf »). Si besoin, des informations sur les diagrammes SysML figurent dans la ressource : éléments sur quelques diagrammes SysML (fichier « Ressource\_Elements\_sur\_Diagrammes\_SysML »). Les activités « Hydrao\_solution\_1 » et « Hydrao\_solution\_2 » traitent plusieurs aspects fonctionnels et technologiques au niveau système et peuvent également être consultées.



volume\_eau > 10 L

volume\_eau > 30 L etc

L’algorithme et le programme python correspondants à cette machine d’états sont donnés en **annexe 1**.

**Question 5**

En analysant le diagramme états-transitions, répondez aux questions suivantes :

Quelle est la condition de passage de l’état « *pommeau\_off* » à l’état « *pommeau\_on* » ?

**Question 6**

Quelles sont les actions réalisées à l’activation de l’état « *pommeau\_on* » ?

**Question 7**

En tenant compte des paliers prédéfinis de consommation d’eau, complétez le diagramme états-transitions, l’algorithme et le programme python pour rajouter l’état violet.

**Question 8 : configuration de l’entrée d’acquisition du signal généré par la turbine**

La génératrice synchrone entraînée par la turbine génère une tension sinusoïdale ***Ugene*** dont la fréquence est égale à la fréquence de rotation de la turbine. Cette tension sinusoïdale est transformée en signal carré ***creneaux\_gene*** de même fréquence et compatible avec les entrées du microcontrôleur (cf. schéma synoptique en page 9 du document ressource : « ***Hydrao\_SysML\_mai21a.pdf*** » du dossier technique).

La mesure du volume d’eau repose sur le comptage des fronts montants et/ou descendants du signal ***creneaux\_gene*** en utilisant la technique de requête d’interruption matérielle ou IRQ.

Cf. ***Ressource\_interruptions\_programmation.doc.***

En vous reportant au programme python de l’annexe 1, indiquez les conditions de déclenchement de l’interruption et la conséquence sur le déroulement du programme.

**Question 9**

En utilisant ***Guide\_utilisation\_environnement\_de\_programmation\_micropython.docx***, faites un copier-coller de votre programme complété dans l'onglet Main de PyScripter-Launcher et transférez-le dans la carte NUCLEO-STM32WB55. Testez votre programme et mettez-le au point si nécessaire.

## 2.2.  Ajout de l’état pause et sans la transition « after\_duree\_max\_pause »

Dans le diagramme états-transitions simplifié ci-dessous, on a ajouté l’état « pause » correspondant à une interruption l’écoulement de l’eau (le temps d’appliquer un shampoing par exemple).



**Question 10**

Complétez le diagramme états-transitions, l’algorithme et le programme de l’activité 2.1 pour ajouter l’état pause.

## 2.3. Ajout de la transition « after\_duree\_max\_pause.

Le diagramme états-transitions précédant est complété par la transition « *after\_duree\_max\_pause* » correspondant au dépassement du délai maximal de pause.



**Question 11**

Complétez le diagramme états-transitions, l’algorithme et le programme de l’activité 2.2 pour ajouter cette transition. On rappelle que la durée maximale de l’état pause est de deux minutes (qu’on réduira à cinq secondes pour les essais).

## 2.4. Pour aller plus loin : gestion de l’éclairage rouge clignotant

Pour faire clignoter les LEDs, on utilisera deux états : « rouge\_1seconde » et « noir\_1seconde » correspondant respectivement à l’allumage des LEDs rouges, puis l’extinction de toutes les LEDs.

Le diagramme états-transitions est donné ci-dessous.

**Question 12**

Complétez le programme python pour ajouter et gérer ces deux états « rouge\_1seconde » et « noir\_1seconde ».

On utilisera un Timer virtuel pour générer un évènement toutes les secondes.



**Annexe 1 : Algorithme et programme python simplifiés du diagramme états-transitions « technologique » sans l’état « pause »**

------------------------------------------------------------------------------------------

// déclarations et initialisations

// seuils de l’image de Upuiss : imageUpuiss = Upuiss/10

seuil\_haut := 0,23

// nombre de tours effectués par la turbine pour un litre

nb\_tours\_par\_litre := 545

// nombre de tours effectués par la turbine

nb\_tours := 0

nb\_tours\_photo\_precedent := 0

// états

pommeau\_off := **faux**

pommeau\_on := **faux**

// pour simplifier on écrit simplement vert plutôt que pommeau\_on\_vert car il n’y pas de confusion possible

vert := **faux**

bleu := **faux**

rouge := **faux**

// evenements

when\_image\_Upuiss\_sup\_seuil\_haut := **faux**

when\_nb\_tours\_sup\_10 := **faux**

when\_nb\_tours\_sup\_30 := **faux**

// entrees et photographies

image\_Upuiss\_photo\_precedent := 0

// sorties RGB

RGB := 000

// procédure appelée à chaque interruption déclenchée par « creneaux\_gene », donc à chaque tour de la turbine

// mise à jour en temps réel

**procédure** interruption\_creneaux\_gene()

**début**

**si** (pommeau\_on) **alors**

nb\_tours := nb\_tours + 1

**fin\_si**

**fin**

// procédure d’acquisition des entrées, de calcul des prédicats et des occurrences des événements

**procédure** acquisition\_entrees\_calcul\_predicats\_et\_occurrences()

**début**

// photographie des entrées pour la cohérence des calculs

nb\_tours\_photo := nb\_tours

image\_Upuiss\_photo := entree\_analogique\_processeur

// calcul des prédicats

// calcul des occurrences des événements

**si** (nb\_tours\_photo > 10 x nb\_tours\_par\_litre) **et** (nb\_tours\_photo\_precedent <= 10

x nb\_tours\_par\_litre) **alors**

when\_nb\_tours\_sup\_10 := **vrai**

**sinon\_si** (nb\_tours\_photo > 30 x nb\_tours\_par\_litre) **et** (nb\_tours\_photo\_precedent <= 30

x nb\_tours\_par\_litre) **alors**

when\_nb\_tours\_sup\_30 := **vrai**

**fin\_si**

nb\_tours\_photo\_precedent := nb\_tours\_photo

**si** (image\_Upuiss\_photo > seuil\_haut) et (image\_Upuiss\_photo\_precedent <= seuil\_haut)

**alors**

when\_image\_Upuiss\_sup\_seuil\_haut := **vrai**

**fin\_si**

image\_Upuiss\_photo\_precedent := image\_Upuiss\_photo

**fin**

**procédure** traitement\_when\_image\_Upuiss\_sup\_seuil\_haut()

**début**

when\_image\_Upuiss\_sup\_seuil\_haut := **faux**

// état pommeau\_off

**si** (pommeau\_off) **alors**

pommeau\_off := **faux**

pommeau\_on := **vrai**

nb\_tours := 0

nb\_tours\_photo\_precedent := 0

vert := **vrai**

RGB := 010

**fin\_si**

**fin**

**procédure** traitement\_when\_nb\_tours\_sup\_10()

**début**

when\_nb\_tours\_sup\_10 := **faux**

**si** (vert) **alors**

vert := **faux**

bleu := **vrai**

RGB := 001

**fin\_si**

**fin**

**procédure** traitement\_when\_nb\_tours\_sup\_30()

**début**

when\_nb\_tours\_sup\_30 := **faux**

**si** (violet) **alors**

bleu := **faux**

rouge := **vrai**

RGB := 100

**fin\_si**

**fin**

// transition initiale de la machine

pommeau\_off := **vrai**

// programme principal

**itérer**

acquisition\_entrees\_calcul\_predicats\_et\_occurrences()

// choix priorisé des occurrences des événements

**si** (when\_image\_Upuiss\_sup\_seuil\_haut) **alors**

traitement\_when\_image\_Upuiss\_sup\_seuil\_haut()

**sinon\_si** (when\_nb\_tours\_sup\_10) **alors**

traitement\_when\_nb\_tours\_sup\_10()

**sinon\_si** (when\_nb\_tours\_sup\_30) **alors**

traitement\_when\_nb\_tours\_sup\_30()

**fin\_itérer**

// fin du programme

------------------------------------------------------------------------------------------

**Programme Python**

# Déclaration des bibliothèques à importer

**import** pyb

# Configuration des broches d’entrées/sorties digitales pilotant les LEDs

# Configuration de PC10 en sortie pour la commande des LED rouges

red\_leds **=** pyb**.**Pin**(**pyb**.**Pin**.**cpu**.**C10**,** pyb**.**Pin**.**OUT\_PP**)**

# Configuration de PA15 en sortie pour la commande des LED vertes

green\_leds **=** pyb**.**Pin**(**pyb**.**Pin**.**cpu**.**A15**,** pyb**.**Pin**.**OUT\_PP**)**

# Configuration de PA5 en sortie pour la commande des LED bleues

blue\_leds **=** pyb**.**Pin**(**pyb**.**Pin**.**cpu**.**A5**,** pyb**.**Pin**.**OUT\_PP**)**

# Configuration de la broche A0 en entrée analogique

adc\_A0 **=** pyb**.**ADC**(**pyb**.**Pin**(**'A0'**))** # mesure de image\_Upuiss (= Upuiss/10)

# seuils de image\_Upuisance : image\_Upuissance = Upuissance / 10

seuil\_haut **=** 285 # 0,23 volt

# Declaration/Configuration du déplacement de la turbine (nbre de tours par litre)

nb\_tours\_par\_litre **=** 545 # 545 tours par litre pour le pommeau Aloe

# Declaration/Initialisation des variables du système

nb\_tours **=** 0 # Nombre de tours effectués par la turbine

nb\_tours\_photo\_precedent **=** 0

RGB **=** **(**0**,** 0**,** 0**)**

# Déclaration/Initialisation des états

pommeau\_off **=** **False**

pommeau\_on **=** **False**

vert **=** **False**

rouge **=** **False**

bleu **=** **False**

# evenements

when\_image\_Upuiss\_sup\_seuil\_haut **=** **False**

when\_nb\_tours\_sup\_10 **=** **False**

when\_nb\_tours\_sup\_30 **=** **False**

# entrees

image\_Upuiss\_photo\_precedent **=** 0

# sorties RGB

red\_leds**.**off**()**

green\_leds**.**off**()**

# Fonction appelée à chaque interruption déclenchée par front montant

# « creneaux\_gene », donc à chaque tour de la turbine

# mise à jour en temps réel

**def** interruption\_creneaux\_gene**(**none**):**

**global** nb\_tours

**if(**pommeau\_on**)** **:**

nb\_tours **=** nb\_tours **+** 1

# Initialisation interruption externe

IT\_turbine **=** pyb**.**ExtInt**(**pin **=** pyb**.**Pin**(**pyb**.**Pin**.**cpu**.**C1**),** mode **=** pyb**.**ExtInt**.**IRQ\_RISING**,** \

pull **=** pyb**.**Pin**.**PULL\_NONE**,**callback **=** interruption\_creneaux\_gene**)**

# mode = choix du mode de déclenchement de l'interruption (ici: sur front montant)

# pull = possibilité d'appliquer une PULL\_UP/PULL-DOWN ou ne rien faire PULL\_NONE

# callback = fonction appelée par l'interruption

# procédure acquisition des entrées, de calcul des prédicats et des occurrences des événements

**def** acquisition\_entrees\_calcul\_predicats\_et\_occurrences**():**

**global** image\_Upuiss\_photo\_precedent

**global** when\_image\_Upuiss\_sup\_seuil\_haut**;**

**global** nb\_tours\_photo**;** **global** nb\_tours\_photo\_precedent

**global** when\_nb\_tours\_sup\_10**;**

**global** when\_nb\_tours\_sup\_30

# photographie des entrées pour la cohérence des calculs

nb\_tours\_photo **=** nb\_tours

image\_Upuiss\_photo **=** adc\_A0**.**read**()**

# calcul des occurrences des événements

**if((**nb\_tours **>** 10 **\*** nb\_tours\_par\_litre**)** **and** **(**nb\_tours\_photo\_precedent **<=** 10 **\*** nb\_tours\_par\_litre**)):**

when\_nb\_tours\_sup\_10 **=** **True**

**elif((**nb\_tours **>** 30 **\*** nb\_tours\_par\_litre**)** **and** **(**nb\_tours\_photo\_precedent **<=** 30 **\*** nb\_tours\_par\_litre**)):**

when\_nb\_tours\_sup\_30 **=** **True**

nb\_tours\_photo\_precedent **=** nb\_tours\_photo

**if((**image\_Upuiss\_photo **>** seuil\_haut**)** **and** **(**image\_Upuiss\_photo\_precedent **<=** seuil\_haut**)):**

when\_image\_Upuiss\_sup\_seuil\_haut **=** **True**

image\_Upuiss\_photo\_precedent **=** image\_Upuiss\_photo

**def** traitement\_when\_image\_Upuiss\_sup\_seuil\_haut**()** **:**

**global** when\_image\_Upuiss\_sup\_seuil\_haut

**global** pommeau\_off**;** **global** pommeau\_on

**global** nb\_tours**;** **global** nb\_tours\_photo\_precedent

**global** nb\_tours\_sup\_10**;**

**global** nb\_tours\_sup\_30

**global** vert**;** **global** bleu**;** **global** rouge**;**

**global** RGB

when\_image\_Upuiss\_sup\_seuil\_haut **=** **False** # RAZ memorisation de l'occurence evenement

# état pommeau\_off

**if(**pommeau\_off**)** **:**

pommeau\_off **=** **False**

pommeau\_on **=** **True**

nb\_tours **=** 0

nb\_tours\_photo\_precedent **=** 0

vert **=** **True**

RGB **=** **(**0**,** 1**,** 0**)** # vert

**def** traitement\_when\_nb\_tours\_sup\_10**()** **:**

**global** when\_nb\_tours\_sup\_10

**global** vert**;** **global** bleu**;** **global** RGB

when\_nb\_tours\_sup\_10 **=** **False** # RAZ evenement

**if(**vert**)** **:**

vert **=** **False**

bleu **=** **True**

RGB **=** **(**0**,** 0**,** 1**)** # bleu

**def** traitement\_when\_nb\_tours\_sup\_30**()** **:**

**global** when\_nb\_tours\_sup\_30

**global** rouge**;** **global** RGB

when\_nb\_tours\_sup\_30 **=** **False** # RAZ evenement

**if(**violet**)** **:**

bleu **=** **False**

rouge **=** **True**

RGB **=** **(**1**,** 0**,** 0**)** # rouge

# Transition initiale de la machine

pommeau\_off **=** **True**

# Debut du programme principal

# Séquence d'allumage des LEDS en fonction du volume d'eau écoulé

**while** **True** **:** # Debut boucle infinie

acquisition\_entrees\_calcul\_predicats\_et\_occurrences**()**

# choix priorisé des occurrences des événements

**if(**when\_image\_Upuiss\_sup\_seuil\_haut**):** # Couleur verte jusqu'à 10L

traitement\_when\_image\_Upuiss\_sup\_seuil\_haut**()**

**elif(**when\_nb\_tours\_sup\_10**):** # Couleur bleue de 10L à 20L

traitement\_when\_nb\_tours\_sup\_10**()**

**elif(**when\_nb\_tours\_sup\_30**):** # Couleur rouge fixe de 30 à 40L

traitement\_when\_nb\_tours\_sup\_30**()**

# Actualisation des valeurs des sorties digitales RGB

red\_leds**.**value**(**RGB**[**0**])**

green\_leds**.**value**(**RGB**[**1**])**

blue\_leds**.**value**(**RGB**[**2**])**

# Fin boucle infinie

# Fin du programme principal