

# Objectif du projet

Le système mécatronique est un banc de transfert avec son système d'évacuation commandé via une carte Arduino. L'objectif est de commander ce système en adoptant un point de vue industriel et non pas scolaire. Cela signifie qu'en plus du cycle de base : Aller-Evacuation-Retour, il faudra envisager d'éventuelles défaillances (ou aléas de production) et des modes de fonctionnement.

Pour suivre une démarche d'ingénierie, vous aurez à écrire pour chaque partie du projet une spécification comportementale en State Diagram. Il vous faudra ensuite réaliser cette spécification en écrivant un programme pour l'unité de commande Arduino. Cela vous permettra de valider votre spécification. D'un point de vue matériel, la partie opérative (banc de transfert) est déjà réalisée. Vous n'aurez qu'à « fabriquer » l'Interface Homme Machine pour les dernières parties de ce projet à l'aide d'une plaque d'essai et de composants (LED, Bouton poussoir, Interrupteur, joystick, etc.)

Ce projet est conçu de façon incrémentale : On part de la commande de base qui sera enrichie petit à petit. À chaque fois vous devez faire une spécification State Diagram de la commande puis faire un nouveau programme pour la réaliser.

**Rendus : vous déposerez sur Moodle pour chaque partie traitée :**

- La spécification State Diagram : Pour chaque partie un projet Cameo intitulé :**vosNomsPartieNuméro**
- Un programme Arduino nommé obligatoirement : **vosNomsPartieNuméro**

# Partie 1 : Cycle de base

## 1.1) Partie 1 : Cahier des charges

- Initialement le chariot se trouve à l'arrêt en fin de course côté moteur (FCM)
- Dès que la présence d'une pièce est détectée le chariot avance jusqu'à la fin de course côté Arduino qui correspond au poste de déchargement.
- Lorsque le chariot est au poste de déchargement : Une LED évacuation est allumée et on procède à l'évacuation de la pièce :
  - Le vérin sort jusqu'à absence pièce
  - Le vérin rentre jusqu'à sa position rentrée
- Pendant que le vérin rentre le chariot recule jusqu'à la fin de course côté moteur et la LED évacuation est éteinte
- Quand il atteint la fin de course côté moteur, le cycle est fini

## 1.2) Partie 1 : Spécification du cycle de base

- **Les entrées/sorties du système** (respectez impérativement les noms donnés ici) :

Les entrées externes peuvent être de type booléen ou numérique (par exemple quand elles proviennent d'un capteur analogique). Pour les entrées booléennes vous utiliserez un déclencheur de transition de type SignalEvent. Pour les entrées numériques utilisez un déclencheur de transition de type ChangeEvent avec une comparaison.

**Entrées :**    **fca** : fin de course côté Arduino (entrée booléenne)

**fcm** : fin de course côté moteur (entrée booléenne)

**pp** : valeur numérique renvoyée par la photorésistance indiquant la présence ou pas de pièce (entrée numérique)  
**verin\_position** : valeur numérique de la position angulaire du vérin (entrée numérique)

**Sorties :**    **AvanceChariot** : pour faire déplacer le chariot vers le côté Arduino

**ReculChariot** : pour faire déplacer le chariot vers le côté moteur CC.

**SortirVerin** : pour faire sortir le vérin

**RentrerVerin** : pour faire rentrer le vérin

**AllumerLED** : pour allumer LED de l'IHM qui signale : chariot en évacuation

**Écrire une spécification State Diagram de la commande du cycle de base (en n'utilisant que ces entrées/sorties).**

Pour les entrées internes comme les variables représentant l'état d'un état i utilisez la garde [in i].

**Contrainte :** Les aspects **transfert** et **évacuation** doivent être décrits par 2 diagrammes respectivement numérotés à partir de 1 et 10.

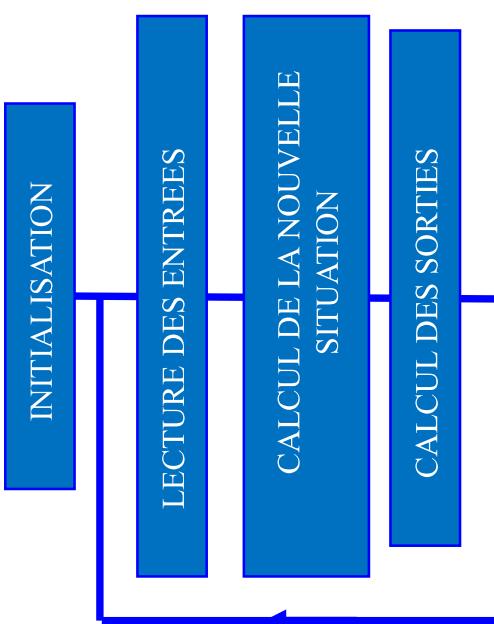
## 1.3) Partie 1 : Programmation de la commande du cycle de base

Dans le programme à compléter **ProjetCSAU2025P1Base** les broches utilisées sont déjà définies voir diapo 12.

Le programme devra réaliser la commande d'un système à évènements discrets. Ce type de programme réalise toujours le cycle suivant :

### 1.3.1) Initialisation (Setup)

Il faut indiquer quelles sont les états actifs initialement. Dans cette partie on traitera également toutes les autres initialisations et configurations nécessaires à ce programme. Initialement le moteur est à l'arrêt, le vérin est rentré et la LED est éteinte.



### 1.3.2) Lecture des entrées (Loop)

Cette partie du programme consiste à donner des valeurs aux entrées booléennes et numériques qui apparaissent dans les States Diagrams : **fcm, fca, pp, verin\_position** en scrutant les broches correspondantes. Toutes les variables correspondantes sont déjà déclarées et initialisées dans le programme de base à compléter.

- Pour les entrées externes **fcm** et **fca** : il suffit de lire l'état des broches FCM et FCA connectées à ces capteurs. Les symboles des broches utilisées sont déjà définis dans le programme de base.

Pour les entrées numériques :

- pp** lire la valeur numérisée renvoyée par la photorésistance sur la broche Capteur\_pp.
- verin\_position** sera calculé à l'aide de la fonction read() de la bibliothèque Servo.

Après avoir donné des valeurs aux variables d'entrée, calculez chacune des conditions de transition de votre state Diagram de spécification. Attention au seuil pour la présence pièce, il peut changer selon les conditions d'éclairage de la salle.

Il n'est pas possible de nommer les transitions sur Cameo, mais pour la lisibilité de votre programme il est préférable qu'elles le soit. Par convention : une transition qui a un état source i sera nommée Ti. Dans votre programme la notation Ti désignera également la condition associée.

Si 2 transitions partent d'un même état source j, elles seront nommées Tja et Tjb.

**Remarque** : Écriture des conditions de transition avec variables numériques:

```
Exemple T10 =when(temperature=100) [PorteFermee]
T20 = when(20<=temperature<=60) [PorteOuverte]
Pour les programmer vous écrivez : T10=(temperature==100) && PorteFermee ;
T20=(temperature>=20 && temperature<=60) && PorteOuverte ;
```

### 1.3.3) Moteur d'évolution (Loop)

**Écrire les formules d'évolution pour chacune des transitions de votre state Diagram de spécification**

### 1.3.4) Gestion des sorties (Loop)

Pour ne pas alourdir votre programme, les sorties utilisées dans votre state diagram de spécification n'apparaissent pas dans le programme de base car pas indispensable. Vous aurez simplement à gérer directement les broches liées aux actionneurs. Établir la correspondance entre les sorties décrites dans votre grafcat et les sorties Arduino.

- Chaque sortie doit être traitée en une seule fois. Pour chaque sortie écrire **toutes** les conditions d'activation de cette sortie. Comme toutes les sorties sont traitées par des activités de type « do », la condition d'activation d'une sortie est que le ou les états où elle apparaît soient actifs.
- Pour le moteur il faudra bien indiquer que si la condition n'est pas satisfaite, le moteur est arrêté. Sinon une fois mis en marche il ne s'arrêtera plus.
- Dans cette partie vous devez agir sur :
- Le moteur : en pilotant une des broches AVANCE et RECUL en PWM. Vous commanderez le moteur à 50% de sa puissance maximale, ce qui correspond à un duty cycle Arduino de 100.
- La broche VERIN en utilisant la bibliothèque servo. Pour le vérin, son mouvement étant limité (il s'arrête quand il a atteint la position voulue) il n'est pas nécessaire, contrairement au moteur, de l'arrêter.
- Complétez le programme ProjetCSAU2025P1Base pour programmer votre spécification State Diagram. Lisez bien les commentaires. Vous n'avez aucun montage à faire dans cette partie car la LED utilisée est celle qui se trouve sur la carte Arduino. Il faut simplement savoir que la cathode de cette LED est connectée au GND et son anode à la broche 13.

# Partie 2 : Cycle avec plusieurs possibilités de postes d'évacuation

## 2.1) Cahier des charges :

Initialement le système est dans la situation : Chariot à l'arrêt en fin de course côté moteur, le vérin en position rentrée et la LED éteinte.

On attend que l'opérateur rentre via le moniteur série la position d'évacuation voulue = distance entière en cm entre le capteur ultrason et la position d'évacuation.

Il y a deux possibilités :

- Si la position est  $> 6$  cm et  $< 36$  cm l'évacuation a lieu à la position choisie
- Si la position est  $< 7$  ou  $> 35$  cm et différente de 0 : dans ce cas l'évacuation aura lieu en fin de course côté Arduino.

Une fois la position choisie on attend qu'une pièce soit présente sur le chariot.

Quand la pièce est présente (et que la position a été choisie) le chariot avance jusqu'à la position d'évacuation choisie.

Une fois la position atteinte le moteur s'arrête, on se trouve en évacuation.

Pendant la phase d'évacuation une LED est allumée, le vérin sort jusqu'à ce que la pièce soit absente. Dès que la pièce est évacuée (absente) la LED s'éteint, le chariot revient à sa position de repos (FCM) et le vérin rentre en position initiale. Le cycle se termine lorsque le chariot arrive en fin de course côté moteur FCM.

## 2.2) Spécification State Diagram de commande du banc de transfert partie 2.

### Les Entrées Sorties du système :

#### Entrées :

fca : fin de course côté Arduino

fcm : fin de course côté moteur

**position** : variable numérique contenant la position de d'évacuation saisie. Vaut 0 si aucune saisie.

**distance\_cm** : variable numérique contenant la valeur de la mesure par le capteur ultrason de la position du chariot

**pp** : valeur numérique renvoyée par la photorésistance indiquant la présence ou pas de pièce

**verin\_position** : valeur numérique de la position angulaire du vérin

#### Sorties :

**AvanceChariot** : pour faire déplacer le chariot vers le côté de l'Arduino

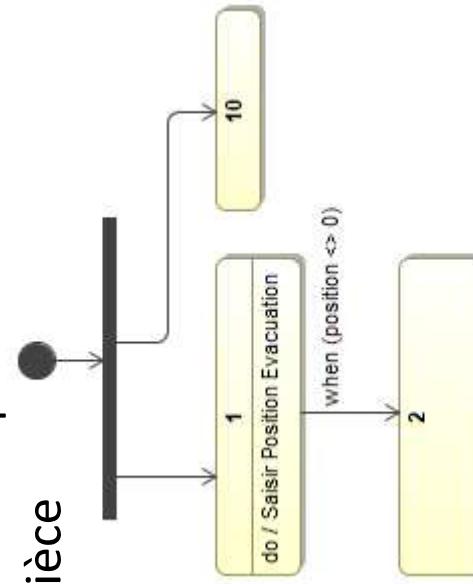
**ReculChariot** : pour faire déplacer le chariot vers le côté du moteur CC.

**SortirVerin** : pour faire sortir le vérin

**RentrerVerin** : pour faire rentrer le vérin

**Saisir position évacuation** : Demande et attente pour saisir la position d'évacuation via le moniteur série

**AllumerLED** : LED de l'IHM qui signale : chariot en évacuation



Complétez la spécification state diagram ci-contre en utilisant les entrées/sorties listées ci-dessus.

Contrainte : Idem que partie 1 : 2 diagrammes avec les mêmes contraintes de numérotation.

## 2.3) Programmation de la commande du banc de transfert

Idem que pour la partie 1 avec en plus la gestion de la sortie « Saisir position évacuation » et les nouvelles variables à utiliser :

- **measure** (résultat de la mesure du capteur ultrason en  $\mu\text{s}$ ),
- **distance\_cm** (résultat de la mesure du capteur ultrason en cm)
- **position**

Ces variables sont déjà déclarées dans le programme de base projetCSAU2025P2Base à compléter.

- Le traitement de la sortie « Saisir position évacuation » est un peu particulier car il se fait via le moniteur série. C'est pour cette raison que cette partie est déjà traitée dans le programme de base à compléter. Cette sortie est associée à l'étape 1. Donc lorsque 1 est active la position est demandée. Puis on attend que l'opérateur rentre une valeur. Cette valeur est stockée dans la variable entière **position**.
- Conseil : Pour vérifier que le chariot est à la position demandée il faudra comparer **position** avec **distance\_cm**. Ne faites pas une égalité car on risque de « rater » la bonne valeur. Utilisez plutôt  $<=$ .

## Partie 3 : partie 2 avec gestion de la perte de la pièce

Nous voulons surveiller l'éventuelle perte de la pièce au cours du transfert.

Pour prendre en compte cet aléas de production, on va compléter le StateChart de la partie 2 avec une nouvelle région dans l'état orthogonal englobant. Cet ajout a pour rôle de déclencher l'envoi d'un message d'alarme sur le moniteur série dès qu'il détecte la perte de la pièce pendant que le chariot avance vers la position d'évacuation.

Quant au chariot, il termine son avance jusqu'à la position voulue puis recule jusqu'à FCM **sans procéder à l'évacuation**. Quand le chariot atteint FCM, on n'envoie plus de message d'alarme sur le moniteur série.

Nous aurons donc une nouvelle sortie : **Alarme** qui consiste à envoyer un message sur le moniteur série.

Établir un nouveau State Diagram de spécification de la commande du banc de transfert qui comprendra la partie de surveillance de perte pièce. Cet ajout devra être numéroté à partir de 20 (idem pour les transitions).

Il devra surveiller s'il n'y plus présence pièce pendant le transfert vers la position d'évacuation. Le plus pratique pour savoir que le chariot est en transfert vers la position d'évacuation est d'utiliser l'état dans lequel se trouve le transfert.

! Il y aura de petites modifications à apporter aux diagrammes « Transfert » et « Évacuation » de la partie 2.

Pour l'aspect programmation, calculez les nouvelles conditions de transition, écrire les nouvelles formules d'évolution (n'oubliez pas de déclarer les nouveaux  $X_i$  et les nouvelles transition) et gérez la nouvelle sortie **Alarme**

## Partie 4 : partie 3 avec marche d'initialisation/test de la partie opérative

Nous reprenons la commande de la partie 3 à laquelle on ajoute une mise en référence et un test de la partie opérative. Reprenez le State diagram de la partie 3 et complétez-le.

Initialement le chariot peut se trouver n'importe où sur le banc, le vérin dans n'importe quelle position.

Au lancement de la commande, le système doit procéder à une mise en référence de la partie opérative accompagnée d'un test du bon fonctionnement du chariot. Cela consiste à :

- Faire avancer le chariot et faire sortir le vérin jusqu'à ce que le chariot atteigne la fin de course côté Arduino
  - Puis faire reculer le chariot et faire rentrer le vérin jusqu'à ce que le chariot atteigne la fin de course côté moteur
- Une fois fait, les diagrammes Transfert, Évacuation et Alarme Perde Pièce pourront évoluer tel que décrit dans la partie 3.

Complétez le state diagram de la partie 3 avec un nouveau diagramme chargé de l'initialisation qui sera numéroté à partir de 30. De même pour les transitions. Il est possible qu'il y ait de petites modifications à apporter.

Pour l'aspect programmation : calculez les nouvelles conditions de transitions, écrire les nouvelles formules d'évolution (n'oubliez pas de déclarer les nouveaux Xi et les nouvelles transitions) et modifiez la gestion des sorties.

# Les entrées/sorties Arduino et symboles utilisés dans le programme

Description	Symbolé utilisé dans le programme	N° broche
Moteur sens direct	AVANCE	5 (PWM)
Moteur sens inverse	RECUL	6 (PWM)
LED évacuation	LEDEVAC	13
Fin de course côté Arduino (fca)	FCA	7
Fin de course côté moteur (fcm)	FCM	8
Photorésistance	Capteur_pp	A0
Vérin	VERIN	9
Capteur Ultrason Trigger	TRIGGER_PIN	10
Capteur Ultrason Echo	ECHO_PIN	11

**Remarque :** pour le vérin, VERIN est le nom donné à sa broche N°9. Par contre dans le programme qui utilise la bibliothèque « servo.h » l'instance de servo est nommée LeVerin.

## Partie 5 : Prise en compte de modes de marche supplémentaires.

### Hierarchisation de la commande

Dans cette partie, en plus de ce qui a été fait dans la partie 4, nous voulons gérer les marches manuelles.

Dans cette application, les marches manuelles seront programmées (et non pas câblées)

Les marches manuelles consistent à pourvoir agir sur le moteur et le vérin par l'intermédiaire d'un joystick sur l'IHM. Elles permettent donc de tester le bon fonctionnement des actionneurs et aussi d'intervenir en cas de problème ou pour des opérations de réglages ou maintenances.

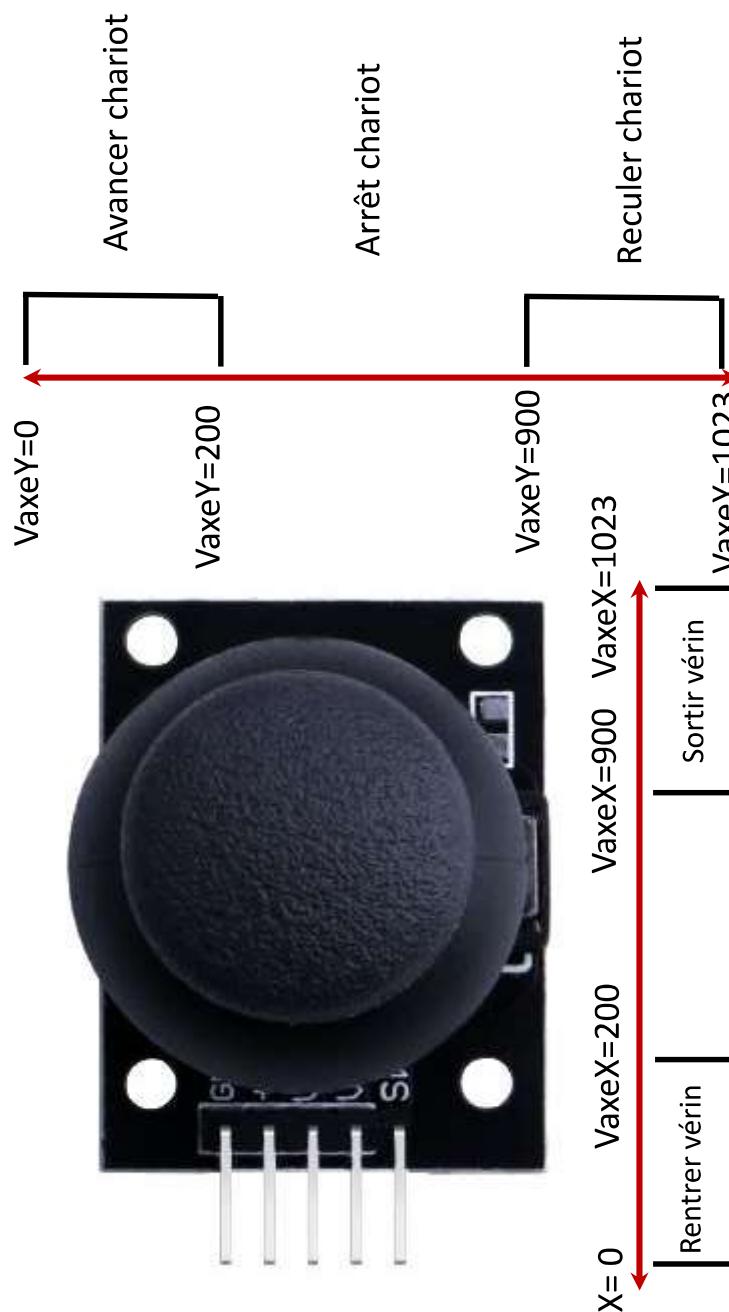
Le fonctionnement du mode manuel est décrit par la figure ci-contre :

Au joystick sont associées 2 nouvelles entrées numériques externes : **VaxeX** et **VaxeY**

Sur l'IHM nous aurons également :

Un sélecteur MdM de marche 2 positions : **autom** ou **manu** correspondant à 2 nouvelles entrées booléennes externes de même nom.

Un sélecteur M/A 2 positions : **marche** ou **arrêt** correspondant à 2 nouvelles entrées booléenne externes de même nom.



## 5.1) Partie 5 : Spécification State Diagram

Dans cette partie, il faudra créer un diagramme de gestion de la marche manuelle telle que décrite diapo précédente. Il devra être partitionné en 2 diagrammes : un qui gère le moteur, un autre qui gère le vérin.  
Il faudra également gérer les passages entre les différents modes de fonctionnement du système : L'initialisation/Test, le fonctionnement normal en mode automatique et les marches manuelles. Pour cela, vous écrivez un diagramme de conduite chargé de gérer ces passages selon le scénario suivant :

Phase 1) :

- Au démarrage le système procède à la phase d'Initialisation/Test de la partie opérative telle que décrite dans la partie 4.
- Une fois la phase d'Initialisation/Test terminée, selon la position du sélecteur MdM :

Phase 2) :

- Si le sélecteur MdM est en position **manu** : on exécute le mode manuel tel que décrit en diapo précédente.
- Si le sélecteur MdM est en position **autom** et que le sélecteur M/A est en position **marche** : On exécute le cycle automatique. La position d'évacuation est demandée la première fois, mais pas pour les cycles suivants : l'évacuation s'effectue toujours dans cette même position.
- On peut sortir du cycle automatique si le sélecteur M/A passe en position **arrêt** : dans ce cas on termine le cycle en cours. Et on revient à la phase 2)
- On peut également sortir du cycle automatique si le sélecteur MdM passe en position **manu**. Dans ce cas, on quitte immédiatement le mode automatique (le moteur doit s'arrêter) et on exécute le mode manuel.
- On peut sortir du mode manuel à n'importe quelle moment en mettant le sélecteur MdM en position **autom**. Dans ce cas on revient à la phase 1).

## 5.2) Partie 5 : Spécification State Diagram

Dans cette partie, nous allons structurer et hiérarchiser la commande.

Le diagramme de conduite sera au sommet de la hiérarchie. Il comprendra 3 super états:

- Un dont la sous machine sera le diagramme d'initialisation. Ce super état sera forcément initial.
- Un dont la sous machine sera les 2 diagrammes du mode manuel.
- Un dont la sous machine sera les 3 diagrammes décrivant le fonctionnement normal : Transfert, Évacuation, Alarme Perte Pièce.

Pour spécifier le diagramme de conduite bien suivre le scénario de la diapo précédente. Comme le fonctionnement normal y sera décrit par un super état, il faudra veiller à le désactiver , suite à une demande d'arrêt, une fois le cycle en cours terminé, et non pas immédiatement.

Pour les diagrammes de fonctionnement normal et d'initialisation reprenez le travail précédent. Toutefois il y aura des modifications à apporter : définir les états de départ, tenir compte du fait que la saisie de la position d'évacuation ne se fera qu'une fois, au moment de l'activation du super état mais pas pour les autres cycles.

**Numérotation :** à partir de 40 pour le diagramme de conduite ; à partir de 50 pour la gestion manuelle du moteur; à partir de 60 pour la gestion manuelle du vérin.

## 5.3) Partie 5 : Les nouvelles entrées

marche : Vraie lorsque le sélecteur M/A est en position marche (entrée booléenne externe)

arrêt : Vraie lorsque le sélecteur M/A est en position arrêt(entrée booléenne externe)

autom : Vraie lorsque le sélecteur MdM est en position autom (entrée booléenne externe)

manu : Vraie lorsque le sélecteur MdM est en position manu (entrée booléenne externe)

VaxeX : valeur numérique de la position du joystick sur l'axe X (entrée numérique (analogique) externe)

VaxeY : valeur numérique de la position du joystick sur l'axe Y (entrée numérique (analogique) externe)

Gestion du joystick : voir figure diapo 15 pour les valeurs à prendre en compte selon l'action désirée.

Exemples :

Condition pour faire avancer le chariot écrire :  $Rxx = [VaxeY \leq 200]$

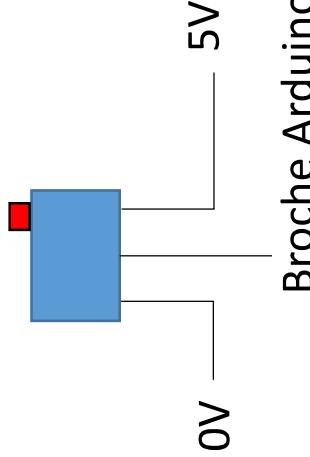
Condition pour arrêter le mouvement d'avance du chariot écrire :  $Ryy = [VaxeY > 200 . VaxeY < 900] + fcm$

## 5.4) Partie 5 : Montage de l'IHM

Les sélecteurs seront réalisés par un interrupteur à glissière 2 positions câblé de la manière suivante :

Attention à son fonctionnement : Lorsque le bouton est du côté du 5V l'interrupteur est ouvert, il renvoie 0V

Lorsque le bouton est du côté du 0V l'interrupteur est fermé, il renvoie 5V



Utilisez une plaque d'essai sur laquelle vous connecterez la ligne « + » au 5V de l'Arduino et la ligne « - » au GND de l'Arduino. A partir de ces 2 lignes vous pourrez alimenter les composants utilisés

Broche Arduino

Le sélecteur MdM sera connecté à la broche **A1** de l'Arduino qui sera nommée **IntMdM** dans votre programme

Le sélecteur M/A sera connecté à la broche **A2** de l'Arduino qui sera nommée **IntMA** dans votre programme

La correspondance avec les variables d'entrée du programme sera la suivante :

Sélecteur MdM à 0V : **manu=1** et **autom=0**

Sélecteur MdM à 5V : **manu=0** et **autom=1**

Sélecteur M/A à 0V : **arrêt=1** et **marche=0**

Sélecteur M/A à 5V : **arrêt=0** et **marche=1**

Le joystick comporte 5 broches. La broche SW ne sera pas utilisée. Reliez les 2 broches d'alimentation à la plaque d'essai. Les 2 broches VrX et VrY qui renvoient un signal analogique proportionnel à la position du joystick sur les axes correspondants seront connectées respectivement aux broches **A3** et **A4** qui seront respectivement nommées **AxeX** et **AxeY** dans votre programme et qui permettront par conversion analogique numérique de donner leur valeurs aux variables **VaxeX** et **VaxeY**. Vous pouvez tester le joystick à l'aide du programme **TestJoystick**.

## 5.5) Partie 5 : Programmation

Définissez les nouveaux symboles de broche :

```
#define intMA A2  
#define intMdM A1  
#define AxeX A3  
#define AxeY A4
```

- Déclarez les nouveaux  $X_i$  et transitions nécessaires. Déclarer les nouvelles variables booléennes arrêt, marche, autom, arrêt et les nouvelles variables entières VaxeX et VaxeY.
- Dans le setup gerez les nouvelles configurations et initialisations
- Rajouter dans la partie lecture des entrées la gestion des nouvelles variables.
- Écrire les formules d'évolution pour le diagramme de conduite en oubliant pas que lorsque l'on active un super état on active également les sous états définis comme initiaux. Lorsque l'on désactive un super état on désactive également tous les états de sa sous machine.
- Écrire/modifier les formules d'évolution pour les autres diagrammes en conditionnant l'exécution de ces formules par le fait que le super état correspondant soit actif.