Volant à retour de force de simulateur

Le volant à retour de force Logitech G27 permet de restituer des sensations aux joueurs de course de voiture conformément au diagramme d'exigences partiel.

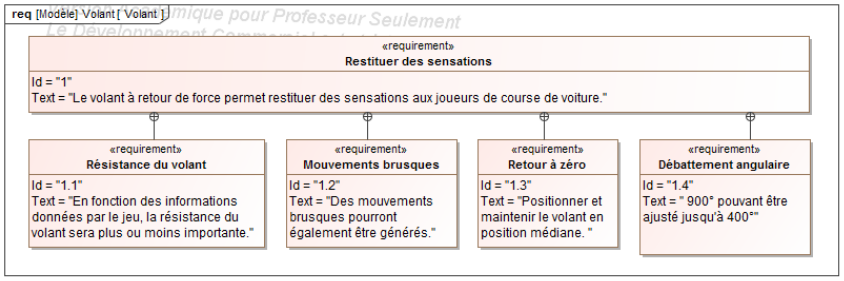
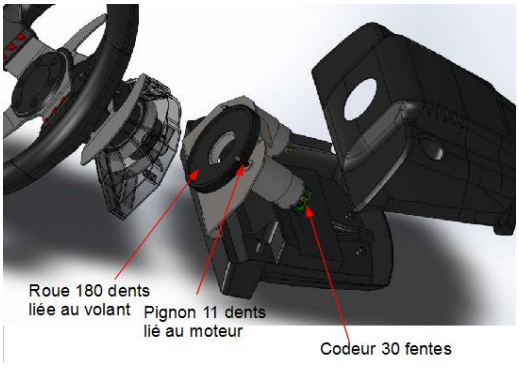


Figure 1 : diagramme des exigences partiel



Ce volant est constitué d'un motoréducteur associé au volant contrôlé par l'intermédiaire d'un codeur incrémental qui permet d'obtenir une bonne précision de positionnement.
Au lancement du PC auquel est associé le volant, il est nécessaire de réaliser une prise d'origine compte-tenu de la nature du capteur utilisé. Cette prise d'origine permet de vérifier l'exigence 1.3 "positionner et maintenir le volant en position médiane".

Objectif

L'objectif du travail proposé est de comprendre les étapes mises en place pendant cette prise d'origine et de proposer une implantation en langage Python de celle-ci.

Spécifier la prise d'origine

La figure suivante montre un relevé de l'angle du volant après lancement de la prise d'origine (le volant était en position centrée au départ). Lorsque le volant tourne vers la droite (sens horaire), l'angle mesuré est négatif.

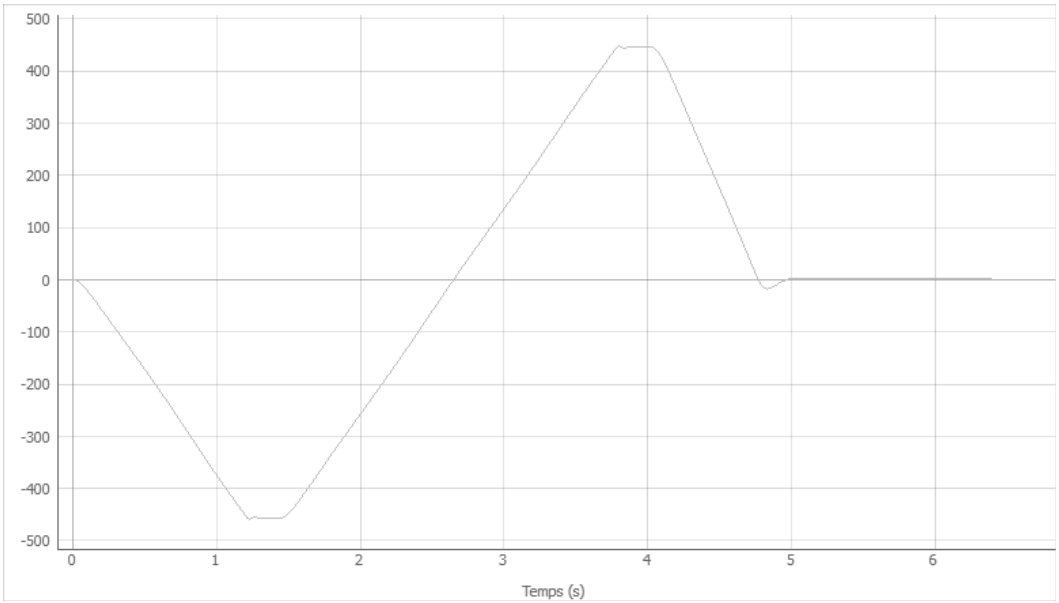


Figure 2 : position angulaire du volant

L'affichage de l'angle volant est possible si on utilise un gain permettant de passer de l'information délivrée par le codeur incrémental en tops à l'angle du volant en degrés.

Le codeur incrémental est constitué d'un disque strié contenant 30 fentes et d'une fourche optique ayant 2 voies (elle délivre ainsi 2 faisceaux lumineux) décalées d'une demi-fente (on parle de quadrature) ce qui permet d'obtenir 4 fois plus d'informations par tour moteur (120 tops).
Lors d'un déplacement, on observe le signal suivant :

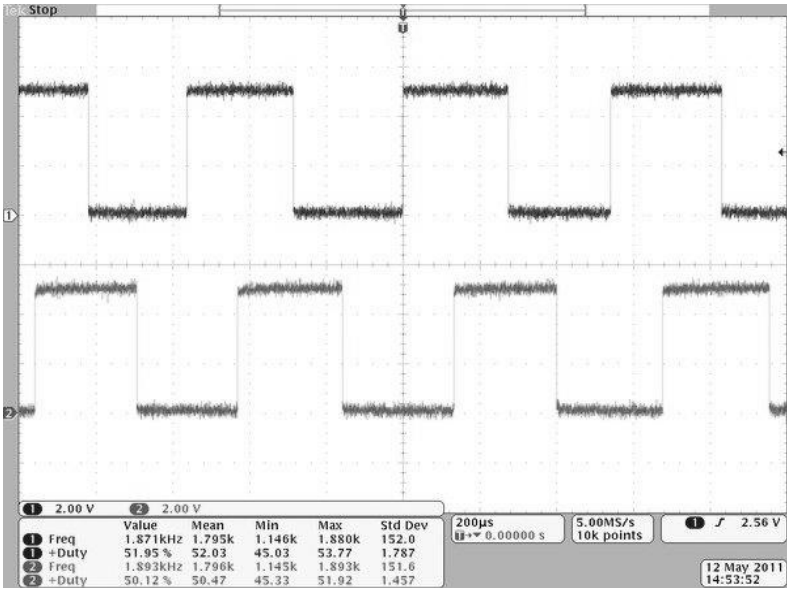


Figure 3 :

Q1. Expliquer comment déterminer la position du volant à l'aide des informations obtenues sur les deux voies. Le codeur est monté directement sur l'axe du moteur. Le réducteur situé entre le moteur et le volant possède un rapport de réduction de 11/180.

Q2. Justifier pourquoi le plus petit angle mesurable au niveau du volant est égal à 0.18°. Commenter cette valeur compte-tenu du domaine d'utilisation du volant. Déterminer le gain permettant de passer du nombre de tops donné par le capteur à l'angle du volant en degré : α= Knbtops. Quel est alors l'intérêt de placer le codeur sur l'axe du moteur et non pas du volant ?

On donne le diagramme d'état simplifié de la prise d'origine.

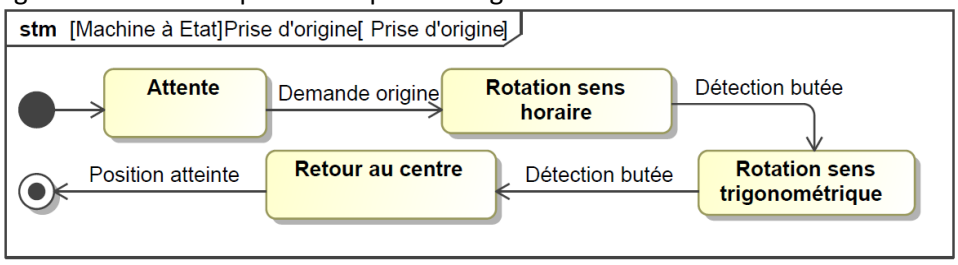


Figure 4 :

Q3. Faire correspondre les différents états, après demande de la prise d'origine, à la courbe d'évolution de l'angle volant.
Q4. Relever la plage de variation angulaire totale du volant. Quelle solution technique pourrait être employée pour détecter les butées gauche et droite ?

En réalité, aucun capteur supplémentaire n'est utilisé pour détecter les butées. On utilise uniquement l'information donnée par le codeur incrémental (appelée tops). Une première façon de détecter la butée est d'acquérir le nombre de tops à chaque instant (toutes les 10 ms) et vérifier si le nombre de tops a varié depuis la dernière acquisition.
On note *nbtops_prec* le nombre de tops acquis à l'instant précédent et *nbtops* le nombre de tops courant.
Dans les deux phases de rotation le moteur tourne grâce à une consigne (PWM) de ± 400.

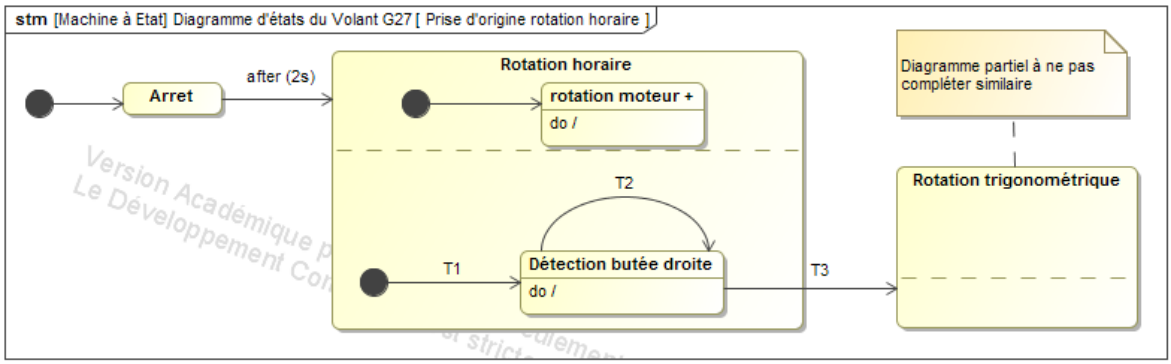


Figure 5 :

Q5. Renseigner les transitions T1, T2 et T3 sur le diagramme de l'état Rotation horaire pour prendre en compte la spécification proposée. Donner également les activités associées aux do/.

Détecter une butée

La spécification n'est cependant pas valide car on constate que le volant met un certain temps à repartir dans l'autre sens (inertie, frottements), ce qui entraîne un blocage immédiat du volant. Il faut donc vérifier si le nombre de tops ne varie pas pendant un temps donné. On propose alors le diagramme d'état complet suivant. Certaines zones n'ont pas été renseignées pour l'instant.

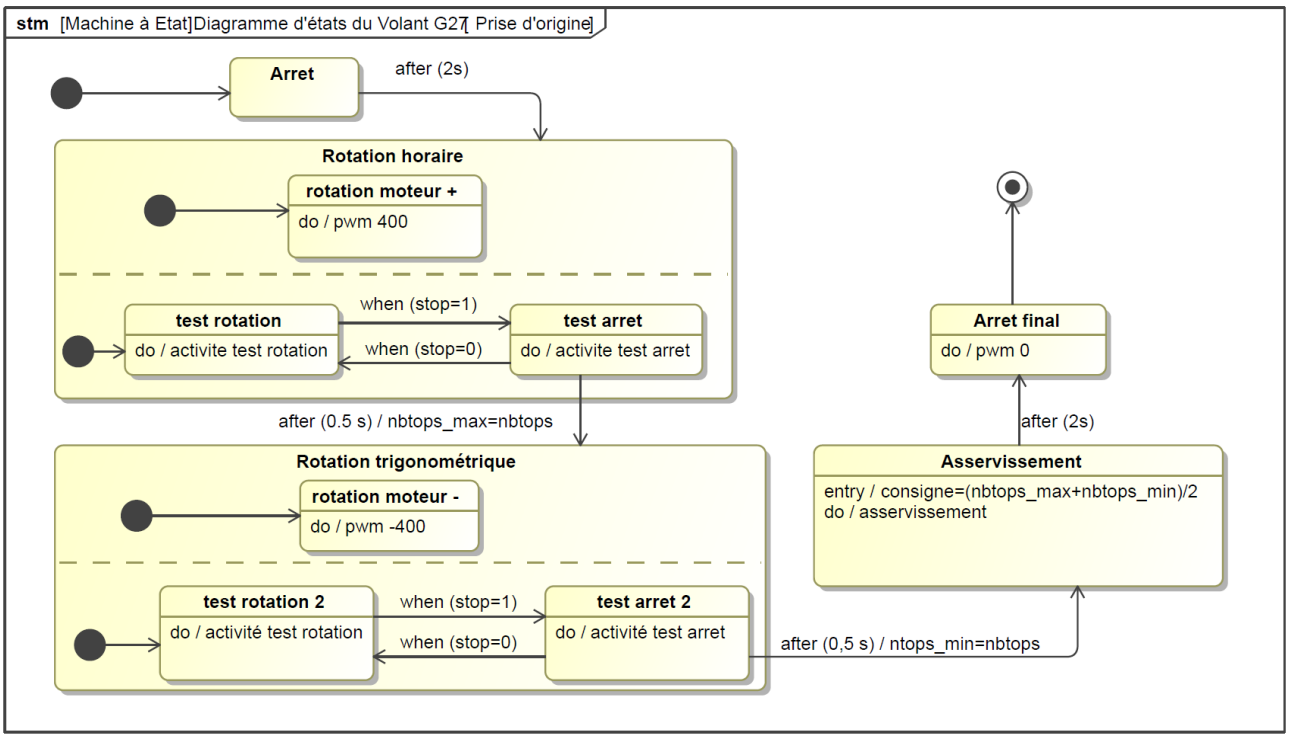


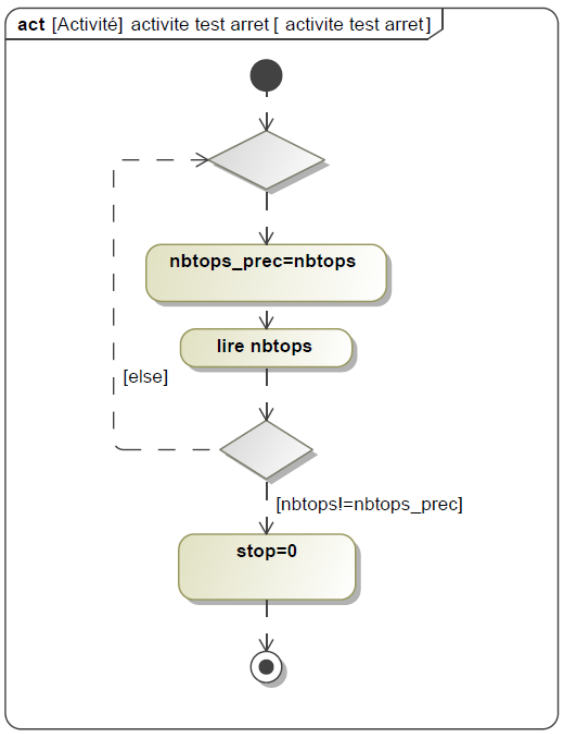
Figure 6 :

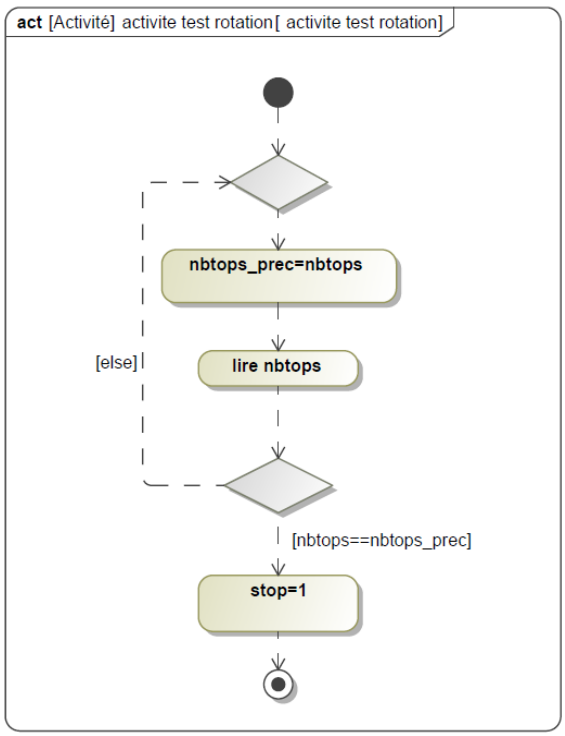
Une fonction (notée *lire_codeur()*) permet de lire et de renvoyer la position du volant (notée *nb_tops*, variable qui peut être positive ou négative). La fonction *pwm(val)* prenant en argument la valeur de consigne pwm permet de faire tourner le moteur plus ou moins vite en fonction de la valeur du pwm *val* .

Q6. A l'aide du diagramme d'état, compléter le programme Python suivant, correspondant au comportement de la prise d'origine.

```
while(etat!=...........):
    if etat==.............. :
        nb_tops_prec=0
        nb_tops=0
        sous_etat="test rotation"
        while (.......................):
            pwm(.....)
            if sous_etat== ................:
                while (.....................):
                    sous_etat=test_rotation(sous_etat)
            elif sous_etat== ....................:
                tic=time.time()
                etape="entry"
                while(..................... and etape!="exit"):
                    sous_etat,etape=test_arret(sous_etat,etape,tic)
                if etape=="exit":
                    etat=.....................
    elif etat==.........................:
        nb_tops_max=nb_tops
        sous_etat="test rotation"
        while (.........................):
            pwm(.....)
            if sous_etat== ............................. :
                while (..............................):
                    sous_etat=test_rotation(sous_etat)
            elif sous_etat== ............................ :
                tic=time.time()
                etape="entry"
                while(.......................and etape!="exit"):
                    sous_etat,etape=test_arret(sous_etat,etape,tic)
                if etape=="exit":
                    etat=.........................
    elif etat==.................................:
        nb_tops_min=nb_tops
        consigne= .....................
        while(etat!="arret final"):
            etat=asservissement(consigne)
pwm(0)
```

Les fonctions test_rotation et test_arret sont décrites par les diagrammes d'activités suivants :





Q7. A l'aide de ces diagrammes d'activité et du diagramme d'état complet ainsi que du programme Python correspondant, proposer une implantation Python des deux fonctions `test_rotation` *et* `test_arret` *qui devront mettre à jour la variable* `sous_etat` *et la variable étape en utilisant les variables* `tic`, `nb_tops`, `nb_tops_prec`.

Q8. Que se passe-t-il si on bloque à la main le volant avant qu'il atteigne la butée ? Quelle autre information peut-on utiliser pour savoir si c'est bien la butée qui est atteinte et non pas l'utilisateur qui bloque le volant.

Spécifier et implanter le retour à zéro

Le retour à zéro est réalisé grâce à un asservissement de position décrit par le schéma-bloc suivant. La difficulté est de calculer la consigne à appliquer à l'asservissement pour que le volant revienne en position centrale.

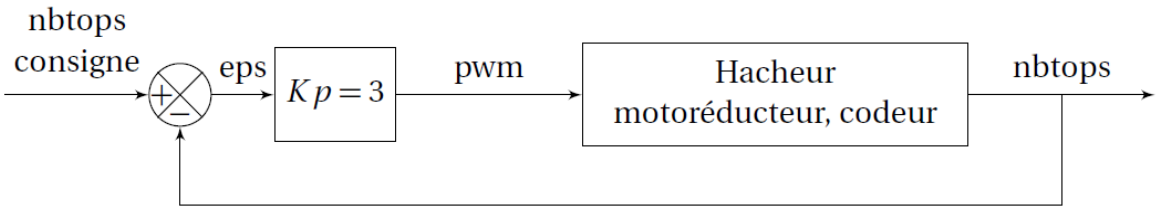


Figure 7 :

Q9. Compléter les actions associées aux transitions after(0.5s)/..... *sur le diagramme d'état complet et définir la consigne au niveau de entry de l'état asservissement.*

Q10. Proposer une implantation simple de la fonction asservissement qui corresponde au schéma-bloc en utilisant la fonction lire_codeur() *et la fonction* pwm(val). *Attention, la valeur du pwm envoyé au hacheur du moteur doit obligatoirement être comprise entre -1024 et 1023, il faut donc faire une vérification avant d'utiliser la commande.*

Jeu sur Wiiboard

La Wii Balance Board (Wiiboard) est un périphérique développé par la société Nintendo pour la console de jeu Wii. Selon Nintendo, la Wii Balance Board permet de pallier le manque d'exercice et d'entraînement qui dégrade la posture de l'homme moderne. Un certain nombre d'exercices ou de jeux sont proposés pour améliorer sa posture. Le point clé de ces jeux est la mesure de la masse et la position du centre de gravité du joueur qui monte sur la Wiiboard.

L'objectif de l'activité proposée est de développer un jeu permettant de tester les capacités d'équilibre du joueur.

LaWiiboard dispose d'un capteur d'effort dans chacun des 4 pieds sur laquelle elle repose. Les données d'efforts sont transmises en Bluetooth. On suppose connue la fonction *calc_xy()* qui récupère les données de mesure et calcule x et y coordonnées du centre de gravité du joueur. Ainsi en bougeant d'avant en arrière et de gauche à droite son corps sur la Wiiboard, le joueur déplace un point à l'écran de coordonnées x et y.

Cahier des charges du jeu

Le jeu proposé est décrit par le diagramme d'exigence partiel

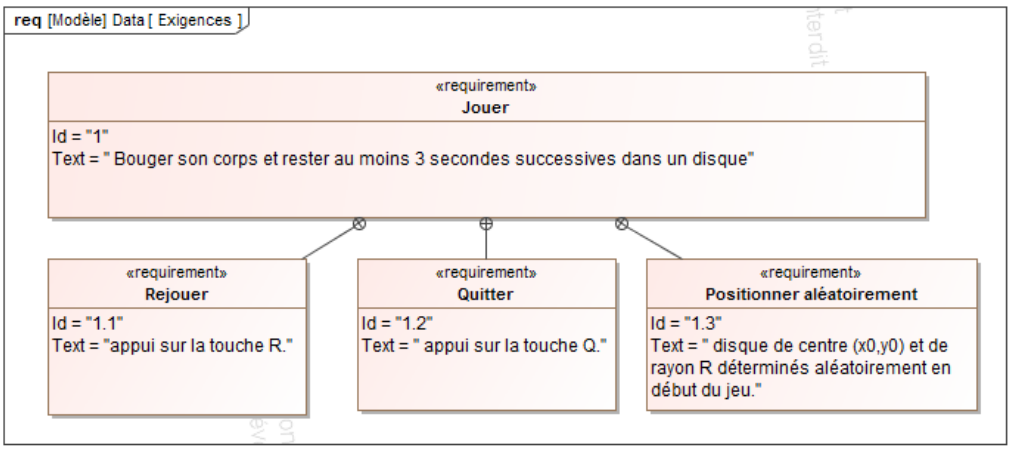


Figure 8 : diagramme des exigences partiel

Le joueur déplace un point à l'écran en s'inclinant. Un disque de centre *(x0,y0)* et de rayon *R* déterminés aléatoirement apparaît au début du jeu. L'objectif est de positionner le point dans le disque et d'y rester 3 secondes.

A chaque fois que le point sort du disque, alors il faut revenir dans le disque et rester à nouveau 3 secondes. Si la partie n'a pas été gagnée au bout de 20 secondes, alors un message indique que la partie est perdue. Une variable *gain=1* permet d'informer le joueur que la partie est gagnée.

Que la partie soit perdue ou gagnée, un message invite à la fin le joueur à recommencer une partie s'il appuie sur la touche R ou bien quitter le jeu s'il appuie sur la touche Q.

Le diagramme d'état suivant décrit la structure globale du jeu d'équilibre proposé. Trois états sont proposés.

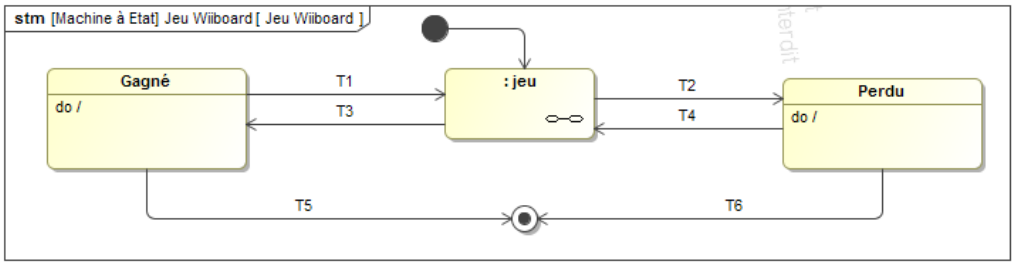


Figure 9 : diagramme d'état du jeu proposé

Q1. Renseigner les transitions permettant de passer d'un état à l'autre et compléter les comportements do/ des états Gagné et Perdu. On ne détaillera pas pour l'instant l'état composite.

L'état composite correspondant au moteur de jeu est décrit par le diagramme d'état suivant.

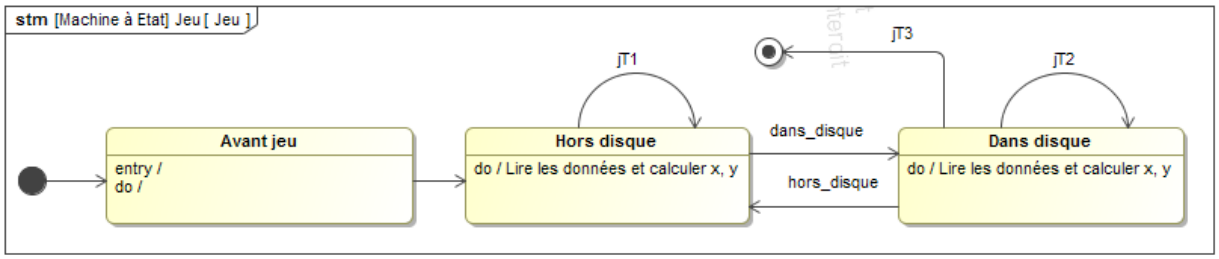


Figure 10 : diagramme d'état du moteur de jeu

Q2. Compléter l'état Avant jeu en indiquant l'action à mettre dans Entry/ vis à vis de la variable gain puis l'action qui est réalisée dans do/.
Q3. Quelle action doit-on mettre dans les états Hors disque et Dans disque ?
Q4. Indiquer les conditions qui correspondent aux événements dans_disque et hors_disque.
Compléter les transitions réflexives jT1, jT2 et indiquer la transition jT3 qui permet de sortir du diagramme et indiquer que le jeu est gagné.

La fonction *calc_xy()* renvoie en fait les coordonnées normalisées x, y comprises entre -1 et 1. La fenêtre du jeu a pour dimension 800x600. La fonction *affiche_point(x,y)* permet d'afficher le point à l'écran.

Q5. Indiquer l'opération à faire pour que les variables x et y permettent de positionner un point toujours dans la fenêtre du jeu.

La fonction *randint(a,b)* renvoie un nombre aléatoire entier compris entre a et b.
Le rayon du disque doit être compris entre 10 et 50 pour ne pas avoir un jeu trop simple ou trop compliqué. Les coordonnées *x0 et y0* sont quelconques entre 0,800 et 0,600. La fonction *affiche_disque(x0,y0,R)* permet d'afficher le disque à l'écran. On suppose que le diagramme d'état est cadencé avec une période de 10 ms ce qui veut dire que dans une boucle *while* sous Python les instructions dureront en tout 10 ms.

On utilisera la fonction time.time() qui renvoie le temps courant pour la gestion des événements du type after(x s). Pour programmer facilement le diagramme d'état, on pourra utiliser une variable dans_disque mise à 1 ou 0 qui permet de savoir dans quel état on se trouve.

Q6. Afin de satisfaire le cahier des charges, proposer une fonction jeu() qui utilise les données précédentes et permet de renvoyer la variable gain égale à 1 si la partie est gagnée ou à 0 si la durée de 20 secondes est dépassée.