Compteur de bicyclette

Indications

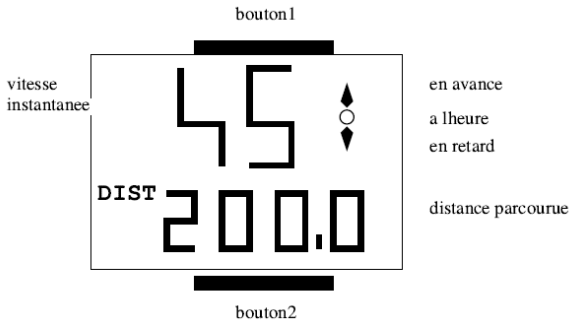
Le projet doit être réalisé en utilisant les outils Lustre étudiés en TP. . Des instructions d'installation des outils Lustre sur des postes personnels sont décrites dans le document « ***Instructions_Installation_Lustre.pdf*** ».

Un rapport final doit être rendu sous format **papier** au plus tard le **lundi 06 janvier 2020**.

Les codes sources doivent être envoyés sous forme d'une archive **tgz** par mail au plus tard le lundi **06 janvier 2020**. Le titre du mail « Projet n4 – Nom1-Nom2-Nom3 »

Description de l'application

L'objectif de cet exercice est de programmer le contrôleur d'un compteur kilométrique de bicyclette. L'interface de ce compteur est décrite ci-dessous:

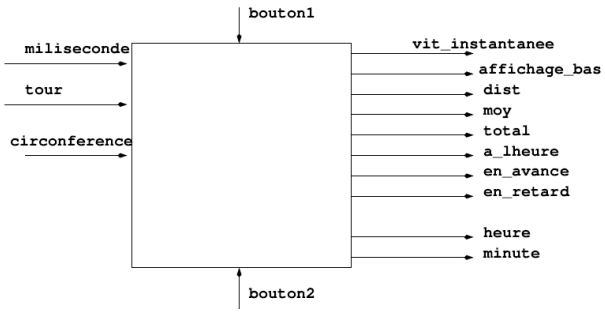


- Le compteur possède deux boutons `bouton1` et `bouton2`.
- Il affiche en permanence la vitesse instantanée.
- Il affiche en permanence une information permettant de savoir si le cycliste est à l'heure, en avance ou en retard. Cette information est obtenue en comparant la vitesse instantanée avec la vitesse moyenne depuis la dernière remise à zéro.
- Sur le cadran du bas, le compteur affiche l'une seulement des informations suivantes:
  - la vitesse moyenne depuis la remise à zéro. Dans ce cas, l'information `MOY` apparaît.
  - la distance parcourue depuis la remise à zéro. Dans ce cas, l'information `DIST` apparaît.
  - la distance totale. Dans ce cas, l'information `TOT` apparaît. Cette distance n'est pas remise à zéro (sur un compteur, il y a un bouton "caché" et un mode permettant de réinitialiser cette distance).
  - l'heure sous la forme `heure:minute` (sur 4 chiffres)

On passe séquentiellement d'un affichage à l'autre (suivant l'ordre donné ci-dessus) en enfonçant le bouton du bas (`bouton2`)

- la remise à zéro s'effectue en enfonçant les boutons `bouton1` et `bouton2`.

L'objectif est de programmer le contrôleur de ce compteur kilométrique en Lustre. Le contrôleur est représenté par le schéma suivant:



et dont l'interface Lustre est donnée par:
node controleur**(**miliseconde, tour, bouton1, bouton2: bool; circonference: int**)**
returns **(**vitesse_instantanee, affichage_bas:int; dist, moy, total: bool; en_avance, en_retard, a_lheure: bool**);**

Q1. Ecrire un nœud Lustre chargé de calculer la vitesse instantanée. La vitesse instantanée est donnée par le temps écoulé entre deux tours de roues. On suppose pour cela que le temps écoulé entre deux occurrences vraies du signal `miliseconde` est égal à la miliseconde et que la circonférence est donnée en centimètre. La vitesse instantanée doit être donnée en kilomètres par heure. Le signal tour est vrai à chaque tour de roue.
Q2. Ecrire un nœud Lustre de calcul de la vitesse moyenne. Vous devrez être attentif aux problèmes de débordement. On supposera que la vitesse moyenne est inférieure à `vitesse_max`.
Q3. On souhaite indiquer si le cycliste est en avance, en retard ou à l'heure. Pour cela, on compare la vitesse instantanée avec la vitesse moyenne. Le cycliste est en avance lorsque la vitesse instantanée est supérieure à la vitesse moyenne plus un; il est en retard lorsque la vitesse instantanée est inférieure à la vitesse moyenne moins un; sinon, il est dit à l'heure. Ecrire le nœud Lustre correspondant produisant les trois sorties nécessaires.
Q4. Ecrire un nœud de calcul de l'heure (heure/minute). Ce nœud a la signature suivante:
node heure**(**heur0, minute0: int; set: bool;
mili: bool**)** **returns** **(**heure, minute: int**);**
`heur0` et `minute0` sont les valeurs d'initialisation et doivent être prises en compte lorsque `set` est vrai.
Q5. Le compteur affiche en permanence la vitesse instantanée. Par contre, l'écran du dessous peut afficher plusieurs informations différences: la vitesse moyenne, la distance parcourue et la distance totale.
L'écran du bas affiche séquentiellement la vitesse moyenne, la distance parcourue, la distance totale puis la vitesse moyenne, etc. à chaque fois que le bouton du bas est enfoncé. Ecrire un nœud dont l'interface est la suivante:
node affichage_sequentiel**(**bouton: bool; vit_moy, dist_parcourue, dist_totale: int; heure, minut: int**)**
returns **(**affichage: int; vit, dist, tot: bool**);**
qui permet de passer d'un affichage à l'autre à chaque fois que bouton est vrai. Pour simplifier, on supposera que affichage est un entier pouvant contenir l'information heure/minute ou bien la vitesse moyenne ou bien la distance parcourue ou bien la distance totale.
Q6. On peut remettre à 0 le compteur (i.e, réinitialiser la distance parcourue et la vitesse moyenne) lorsque l'on enfonce les deux boutons pendant au moins une seconde. Ecrire un nœud dont l'interface est la suivante: **node** raz**(**mili, b1, b2: bool**)** **returns** **(**ok: bool**);**
Q7. Ecrire le corps du contrôleur en instanciant les divers nœuds réalisés.
Q8. Simuler et tester les nœuds réalisés.
Q9. Donner le code C généré et tester son fonctionnement.
Q10. Générer le nœud final sous forme d'un automate à états finis.
Q11. Proposer et vérifier des propriétés que doit respecter le système (en utilisant xlesar).