	Institut Supérieur des Etudes Technologiques de Sousse **Département Génie Electrique**	***Année universitaire*** *: 2017/2018* ***Semestre*** *: 2* ***Date :*** ***07 Juin 2018*** ***Durée :*** *1h30mn*
Matière *: Système Temps Réel*	Examen	***Classes*** *: EQI2*
Documents *: Non autorisés*	***Enseignant*** *: Ali HMIDENE*	***Nb. Pages*** *: 2 + 1 Annexe*

Problème (12 points)

Les panneaux photovoltaïques assurent une production décentralisée pour alimenter des matériels portatifs ou satisfaire des besoins locaux en des lieux isolés, mais ils participent aussi à la politique énergétique globale grâce à leur connexion aux réseaux de distribution d'électricité.

Pour accroitre le rendement du système photovoltaïque, il est possible d'utiliser des panneaux solaires capables de suivre le soleil, ce qui permet de récupérer constamment le maximum d'énergie lumineuse. Suivre le soleil permet une amélioration de rendement de l'ordre de 40% par rapport à un système fixe.

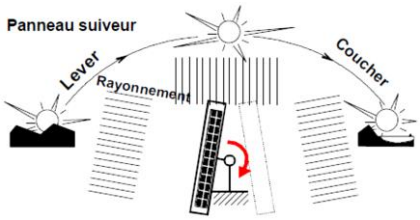

Figure 1

On se propose dans ce sujet de faire la commande simplifiée d'un suiveur de soleil. Le principe utilisé repose sur l'exploitation du déséquilibre créé entre deux résistances LDR (*Light Dependent Resistor*) séparées par une paroi opaque au rayonnement solaire. Le circuit à LDR est monté sur le cadre du panneau solaire et donc lié à ses mouvements. En fonction de la valeur du déséquilibre et de son sens le moteur du vérin est actionné. Ce système est commandé par une carte électronique à base d'un microcontrôleur PIC18f4520 (figure 3).

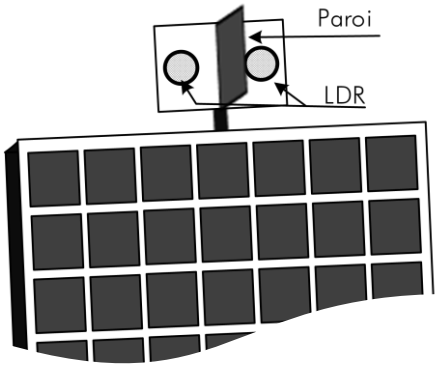

Figure 2

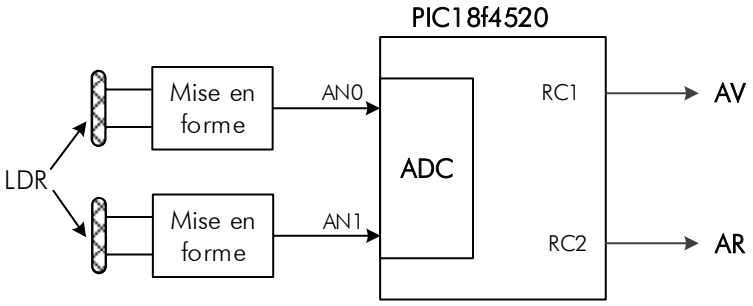

Figure 3

Le circuit de mise en forme fournit en fonction de l'intensité lumineuse, une tension variable de 0 à V_{DD}. Les sorties AV et AR commandent le moteur du vérin dans le sens AVant ou ARrière.

Le programme de commande peut être décomposé en trois tâches (figure 4):

- Acquition1 : convertie la tension appliquée à l'entrée AN0 du convertisseur analogique numérique ADC. La valeur convertie est chargé dans la variable globale Val_LDR1.
- Acquition2 : convertie la tension appliquée à l'entrée AN1 du convertisseur analogique numérique ADC. La valeur convertie est chargé dans la variable globale Val_LDR2.

- Commande : cette tâche compare Val_LDR1 et Val_LDR2 et commande le moteur dans le sens **AV** si Val_LDR1>Val_LDR2 et dans le sens contraire si Val_LDR1<Val_LDR2. En cas d'égalité le moteur s'arrête pour maintenir le panneau en face du soleil.

Le digramme de conception selon la méthode DARTS est donné à la figure 4.

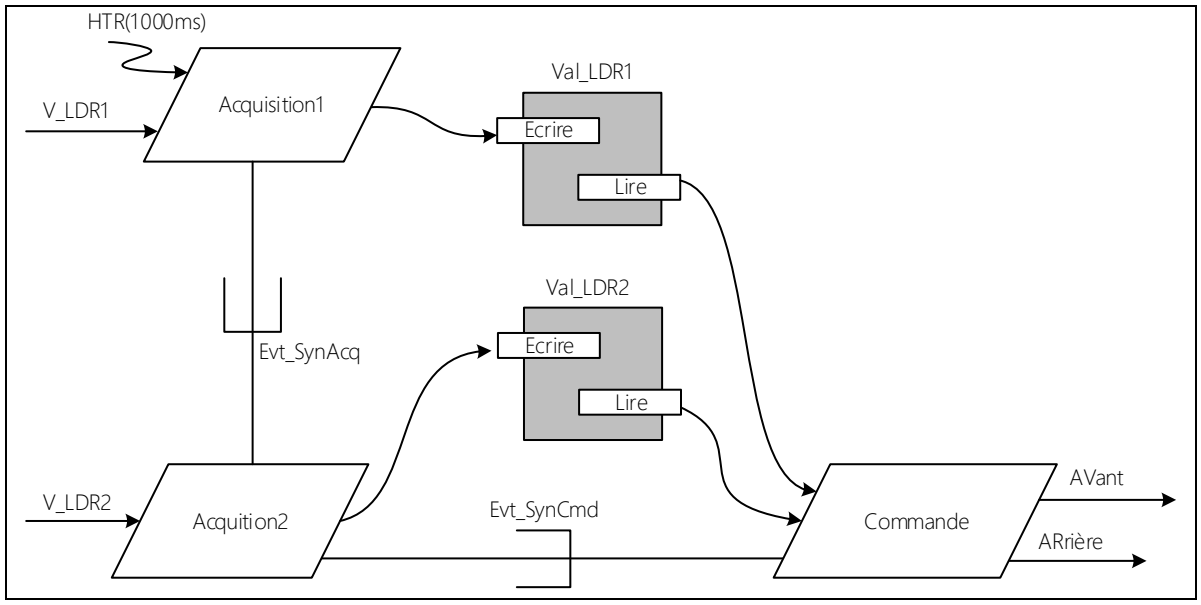

Figure 4

Travail demandé :

Q1. Compléter le programme proposé.
Q2. Peut-on activer la tâche Acquisition2 par un HTR ? Si oui, préciser la période d'activation ainsi que les précautions à prendre.
Q3. Peut-on remplacer la variable globale **Val_LDR2** et l'événement **Evt_SynCmd** par une boîte aux lettres ? Justifier votre réponse.

Exercice (8 points)

On considère une configuration **T** de trois tâches périodiques et indépendantes à échéance sur requête. Les tâches sont définies par les paramètres temporels suivants :

T1 (r0= 0, C1 = 5, P1 = 12)
T2 (r0= 0, C2 = 3, P2 = 8)
T3 (r0= 0, C3 = 1, P3 = 6)

On applique à la configuration T un ordonnancement préemptif à priorité statique selon la plus petite période (RM).

1. Donner le facteur d'utilisation U et conclure sur l'ordonnancement par RM en utilisant le test d'ordonnancement.
2. Donner la valeur de la période d'étude et tracer la séquence d'exécution correspondante.

On applique à la configuration T un ordonnancement préemptif à priorité dynamique selon l'échéance la plus proche (EDF).

3. Etant donné le facteur d'utilisation U, est-ce que le test d'ordonnancement nous permet de conclure sur la faisabilité de l'ordonnancement par EDF?
4. Tracer la séquence d'exécution et identifier les temps creux (libre).

ANNEXE

1. Création d'une tâche

```
BaseType_t xTaskCreate(     TaskFunction_t pvTaskCode,
                            const char * const pcName,
                            unsigned short usStackDepth,
                            void *pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t *pxCreatedTask
                           );
```

2. vTaskDelay

`void vTaskDelay(const TickType_t xTicksToDelay );`
on peut diviser la paramètre par **portTICK_RATE_MS** pour avoir le temps en ms.

3. void vTaskStartScheduler(void) : Démarre le processus du noyau temps réel

4. Sémaphore binaire

Création de sémaphore

`SemaphoreHandle_t xSemaphoreCreateBinary( void );` : renvoie l'identifiant du sémaphore binaire. Le sémaphore est créé à l'état 'vide', ce qui signifie que le sémaphore doit d'abord être libéré en utilisant la fonction API xSemaphoreGive ().

Libérer le sémaphore

`xSemaphoreGive( SemaphoreHandle_t xSemaphore );`

Prendre le sémaphore

`xSemaphoreTake( SemaphoreHandle_t xSemaphore, xTicksToWait );` :

`xTicksToWait` prend `portMAX_DELAY` pour une attente infinie.

5. Conversion analogique Numérique

BusyADC() : En cours de conversion

ConvertADC() : démarrer la conversion

int ReadADC() : renvoie le résultat de conversion

SetChanADC(canal) : sélectionner le canal de conversion (ADC_CH0, ADC_CH1,…)

6. Condition suffisante pour l'ordonnançabilité RM

$U \leq n(2^{\frac{1}{n}} - 1)$: n : nombre de tâches