TP
⋆
Réalisation d'une application simple

10/01/2017

1 Objectif

Pour ce TP, nous allons réaliser une application utilisant l'afficheur LCD et un capteur de distance à ultrasons. Le but est d'afficher sur l'afficheur, à intervalles réguliers, la distance mesurée mais aussi d'afficher un message d'erreur si le capteur est déconnecté.

2 Utilisation de l'afficheur LCD

Nous n'utiliserons que les capacités d'affichage textuel de l'afficheur LCD (pas de graphique, ni de retour tactile). l'ensemble des fonctions disponibles pour le TP se situent dans le fichier `lcd/lcd.h`. La connexion de l'afficheur est la suivante :

broche du LCD	*carte Discovery*
`VCC`	`3V`
`GND`	`GND`
`CS`	`PA8`
`RESET`	`PC8`
`D/C`	`PC9`
`SDI` (MOSI)	`PC12`
`SCK`	`PC10`
`LED`	`PD0`
`SDO` (MISO)	`PC11`

Pour tester l'afficheur LCD, on se contente dans un premier temps d'une application minimale avec une tâche périodique permettant d'afficher le temps écoulé depuis le démarrage de la carte (format `mm:ss`) dans le coin supérieur droit.

La structure cette première version est donnée à la figure 1.

FIGURE 1 – Application de test

Question 1 *Programmez cette première application.*

Question 2 *Mesurer le temps d'exécution de la tâche. On pourra notamment utiliser une sortie GPIO positionnée à 1 à l'entrée de la tâche et à 0 à la fin du code pour visualiser le temps d'exécution à l'oscilloscope.*

3 Application de capteur à ultrasons

3.1 Première approche simple

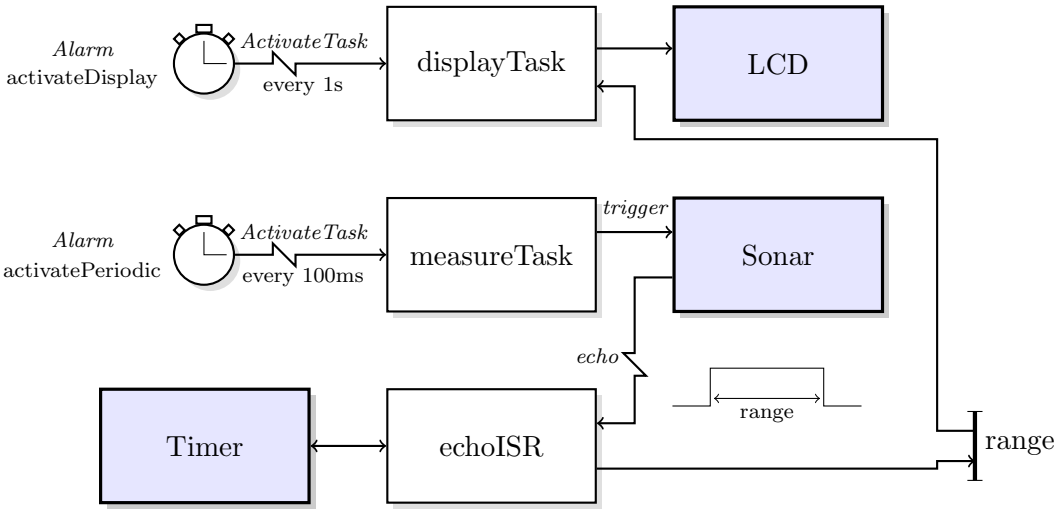

FIGURE 2 – Structure de l'application

L'architecture proposée fournie sur la figure 2. La tâche *measureTask* est activée toutes les 100ms. Elle pilote le capteur et déclenche une mesure de distance.

Lorsque le capteur répond, il produit une impulsion dont la largeur est proportionnelle à la distance mesurée. L'ISR *echoISR* a pour rôle de mesurer la largeur de cette impulsion et est donc déclenchée sur les front montants *et* descendants de l'impulsion. Par une lecture de l'état de l'entrée, elle détermine si il s'agit d'un front montant ou descendant. Un *free running timer*, initialisé à chaque front montant, permet de mesurer la largeur de l'impulsion. La distance mesurée est écrite dans une variable *range* de type *unsigned long*.

La tâche *displayTask* est activées soutes les secondes. Elle lit la variable *range* et l'affiche sur l'afficheur.

Question 3 *Programmez l'application décrite*

Question 4 *Que se passe-t-il si le capteur est déconnecté entre 2 mesures (défaillance matérielle) ?*

4 Gestion de la défaillance capteur

On rajoute une tâche *timeoutTask* selon le schéma 3. La tâche *measureTask* arme une alarme *activateTimeout* qui a pour but de signaler que le capteur ne répond pas. Si une impulsion est mesurée, *echoISR* annule l'alarme *activateTimeout* (cette interaction ne figure pas sur le schéma). Lorsque l'alarme *activateTimeout* expire, elle active la tâche *timeoutTask* qui a pour rôle de signaler une erreur sur l'afficheur : *capteur deconnecte*.

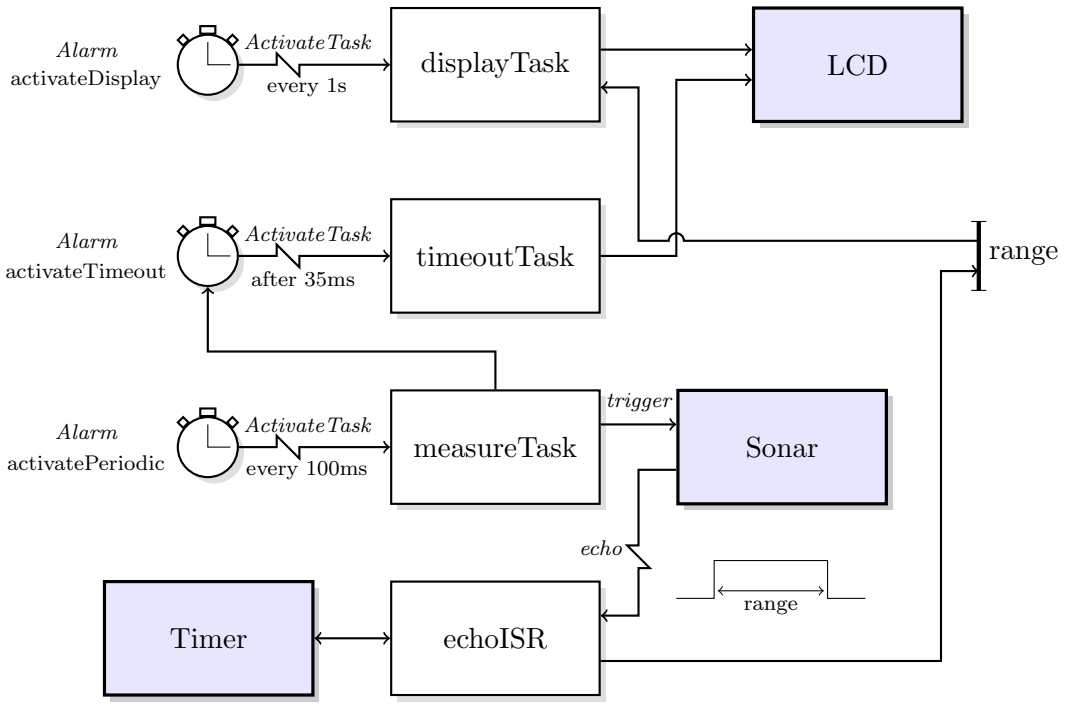

FIGURE 3 – Structure de l'application

Question 5 *Programmez l'application décrite*

Lorsque le capteur est déconnecté, la tâche *displayTask* continue d'afficher une valeur erronée.

Question 6 *Modifiez l'application sans ajouter de variable globale et en n'utilisant que*

les appels système concernant les alarmes de manière à suspendre et à rétablir l'affichage de la distance.

Il est possible que dans certaines circonstances, la valeur lue dans la variable *range* par *displayTask* soit erronée alors que la valeur écrite par *echoISR* est correcte. Il est également possible que les messages affichés s'entrelacent et soient incohérents.

Question 7 *Pourquoi ? Modifiez l'application de manière à assurer que cette variable est toujours correcte et que les message affichés soient toujours cohérents. On utilisera les ressources.*