TP 5 Guirlande

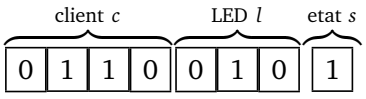
TP noté de 3h. Tous documents autorisés. Communication (orale ou par le réseau) interdite. Veuillez lire le sujet jusqu'à la fin avant de commencer. Vous pouvez utiliser le code des séances précédentes, en particulier la correction de l'exercice 1 du TP 5 (`2_uart_polling/`). Vous devez utiliser la bibliothèque HAL et le `makefile` fourni ; le code récupéré sur internet ou généré automatiquement entraînera une pénalité. Vous avez à votre disposition un oscilloscope et des fils de pontage. À la fin de la séance, envoyez une archive du répertoire constituant votre solution à `matthias.puech@lecnam.net`.

Vous allez réaliser une guirlande lumineuse : une fois le code écrit, nous connecterons toutes les cartes entre elles pour former un réseau, et le "serveur" (une des cartes) contrôlera alors les LEDs des autres cartes ; il émettra des commandes qui feront décrire aux LEDs un motif plaisant.

1 Description du protocole

Le réseau que nous allons construire a une topologie en anneau (Fig. 1). Chaque noeud (i.e. chaque carte) peut envoyer des données au suivant sur l'anneau, et peut en recevoir du précédent. On utilise un périphérique UART par noeud pour le transport physique des données : son entrée (Rx) est connectée au noeud précédent et sa sortie (Tx) est connectée au noeud suivant. Le nombre total de noeud n'est pas fixé : le protocole s'adapte au nombre de noeuds présents (maximum 15).

Un noeud est désigné comme étant le *serveur* ; les autres sont les *clients*, numérotés de 0 à n. Tous les clients se comportent strictement de la même manière. Chaque *message* échangé en UART est constitué d'un seul octet, et commande la (dés)activation d'une des 8 LEDs d'un client donné. Son format est le suivant :



Le client à qui faire parvenir le message est désigné par c (de 0 à 15, codé en binaire non signé), le numéro de la LED sur ce client par l (de 0 à 7) ; si $s = 0$, on commande l'*allumage* de cette LED ; si $s = 1$ on commande son *extinction*. On notera (c, l, s) le message correspondant.

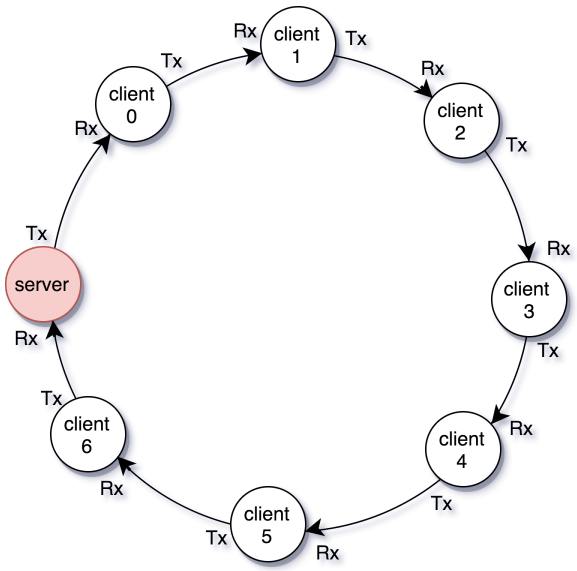


FIGURE 1 – Topologie en anneau

Seul le serveur initie l'envoi d'un message. À la réception d'un message (c, l, s) un client doit :

— ou bien allumer/éteindre la LED l si $c = 0$ (le message lui est destiné) ;
— ou bien retransmettre le message $(c - 1, l, s)$ sur sa sortie si $c \neq 0$.

Ainsi, si le serveur émet le message $(2, 5, 0)$, le client 0 transmettra $(1, 5, 0)$ au client 1, qui lui même transmettra $(0, 5, 0)$ au client 2, qui allumera sa 6ème LED. Si un message (c, l, s) revient au serveur, cela veut dire qu'il a fait le tour de l'anneau, i.e. qu'il l'a adressé à un client $c + n$ qui n'existe pas.

2 Interface utilisateur

On demande le comportement suivant de chaque carte :

— Quand on maintient le bouton pressé, elle est en *mode serveur* : elle émet alors toutes les 32 millisecondes un message : $(0,0,0)$, puis $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, $(0,2,0)$, $(0,2,1)$, ... $(0,7,1)$, $(1,0,0)$, $(1,0,1)$, ... $(15,8,1)$, puis $(0,0,0)$, etc. Si elle reçoit un message quelconque, alors le compteur est remis à $(0,0,0)$.

— Quand le bouton est relâché, elle est en *mode client* : elle traite tous les message de la forme $(0,l,s)$ et émet $(c-1,l,s)$ quand elle reçoit (c,l,s) avec $c \neq 0$.

On fera attention à n'appuyer sur le bouton que d'une seule carte à la fois sur le même réseau.

3 Travail demandé

Implémentez le protocole, côté client et côté serveur, en utilisant la bibliothèque HAL. N'utilisez de préférence pas d'attente active mais des interruptions, et endormez le MCU dès que possible. On utilisera le périphérique USART3 dont PB10 est la sortie (Tx) et PB11 l'entrée (Rx) ; la configuration voulue est 9600 bauds, 8 bits, bidirectionnel, pas de bit de parité. On jugera le travail non seulement sur le fonctionnement du code, mais surtout sur sa clarté : isolez donc bien la gestion matérielle dans des drivers, la decription du protocole, et la logique principale du programme. Argumentez vos choix en commentaire si nécessaire.

Bonus : décrivez (en commentaire) et implémentez une extension de ce protocole permettant de gérer plus de 15 clients et plus de 8 LEDs (vous soumettrez alors deux archives distinctes).