C5-160551-INFO
Objets Connectés :
Programmation Microcontrôleurs - TP3

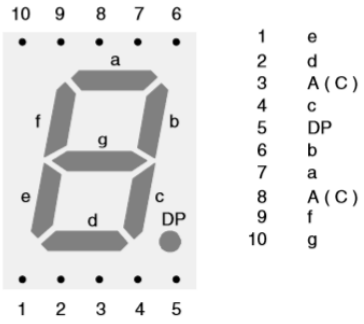
©Bernard Besserer

année universitaire 2021-2022

1 Afficheur externe 7 segments

Attention, un peu plus de câblage pour ce TP. Réfléchir à l'emplacement de l'afficheur sur la plaquette d'essai

Comme indiqué en TD, câblez l'afficheur 7 segments à partir de la broche 12 (broche 12 = segment a, etc..., broche 18 = segment g). N'oubliez pas, dans le setup(), de définir ces broches comme des sorties.



Il y une différence entre cet afficheur et celui de l'exercice proposé en TD : sur le TD l'exercice concernait sur un afficheur à cathode commune et l'afficheur que nous utilisons pour ce TP est de type **anode commune** (voir schéma). Il y a deux broches correspondant à l'anode commune à tous les LEDs formant cet afficheur : la broche 3 et la broche 8. Il suffit d'en câbler une, **en mettant une résistance de protection (de l'ordre de grandeur de la centaine d'Ohms, 3^{eme} bague = brun) entre l'anode et le +3V3**.

Attention, il est possible que le téléversement échoue si l'afficheur est alimenté. Donc, pendant le téléchargement, débranchez le fil relié au +3V3 (et qui va vers l'anode de l'afficheur via la résistance), faite le téléversement puis reconnectez le fil.

Commencez par écrire un programme (dans loop()) qui allume progressivement tous les segments (a, b, ..., g) en utilisant l'instruction classique `digitalWrite()`, avec un delai de 1000ms chaque allumage. Ensuite, vous écrirez un programme principal qui compte en boucle de 0 à 9, avec un delai de 1000ms entre l'affichage de chaque chiffre. Pour afficher les chiffres, **vous utiliserez pour cela une table de correspondance comme vu en TD**, et vous utiliserez les instructions :

```
REG_WRITE ( GPIO_OUT_W1TS_REG ,    xxxxxxxxxx) ;
REG_WRITE ( GPIO_OUT_W1TC_REG ,    xxxxxxxxxx) ;
```

(On rappelle que c'est S pour Set et C pour Clear)

2 Comptage des actions sur une entrée

Câblez un bouton-poussoir comme entrée, comme vu lors du TP1 (avec le bouton-poussoir placé entre une broche et le GND). N'oubliez par la bonne initialisation dans le setup(). Déclarez une variable qui servira de compteur jusqu'à 9 (si l'on dépasse 9, on revient à 0), et c'est une action sur le bouton-poussoir qui va incrémenter cette variable de comptage. Le nombre devra être affiché sur l'afficheur 7 segments.

Si le fonctionnement n'est pas OK (le compteur augmente de 2 ou 3 ou 4 valeur à chaque appui, c'est que votre bouton-poussoir subit des rebonds (*bounces*) et que votre programme en prend compte. Ajoutez le code nécessaire pour ne pas tenir compte des rebonds lors du comptage. Ces rebonds persistent durant 20ms à 40ms maximum.
Montrez à l'intervenant un comptage sans perturbations !

3 Discrimination appui court / appui long

Il vous faudra discriminer des actions courtes et longues (> 1000ms) sur ce bouton-poussoir. Le mode de fonctionnement souhaité est le suivant :

- Incrémentez la variable de comptage à chaque action, avec affichage immédiat de la valeur. Si la variable de comptage dépasse 9, on revient à 0.
- Si c'est une action longue, remise à zéro du compteur. L'affichage doit passer à 0 lorsque la délai correspondant à une action "longue" est atteint.
Si l'on détaille :

— Action sur le bouton-poussoir : On incrémente donc la variable de comptage, et on affiche la nouvelle valeur. On mémorise le temps (fonction millis()) et il faut continuer à exécuter du code dans loop()
· Si le bouton est relâché avant 1000ms, c'est une action courte. On peut faire flasher très brièvement la LED `LED_BUILTIN` pour monter que l'action à été prise en compte.
· Si le delai est dépassé, c'est une action longue : Remise a zéro de la variable de comptage, affichage de cette valeur, éventuellement bref flash de la LED `LED_BUILTIN` pour monter que l'action a été prise en compte. Le relâchement du bouton ne doit entraîner aucune action.

Attention aux rebonds. Tout relâchement détecté durant les 40ms après un appui ne pourra être qu'un rebond ... vous pouvez bien sûr utiliser la fonction delay() pour éviter la prise en compte des informations pendant 40ms et relire la valeur de l'entrée.

☑ Montrez le fonctionnement et mettez le code (avec commentaires !) sur Moodle.

4 Simulation d'un dé

On combine les deux parties précédentes : Lors de l'action sur le bouton-poussoir, vous devez récupérer un nombre aléatoire compris entre 1 et 6. Utilisez pour cela la fonction `esp_random()`. Réfléchissez pour trouvez comment limiter le nombre entre 1 et 6, consultez
`https://techtutorialsx.com/2017/12/22/esp32-arduino-random-number-generation/`.

☑ Affichez immédiatement le nombre sur l'afficheur 7 segments, il doit rester affiché jusqu'à la prochaine action.
Montrez le fonctionnement à l'intervenant.

5 Affichage temporel avec synchronisation

1. Reprendre le code du TP2 pour se connecter au Wifi (toujours votre smartphone en partage de connexion) et utilisez le protocole NTP pour récupérer l'heure depuis un serveur NTP.

   Ecrire un code qui affiche (sur l'afficheur 7 segments) le chiffre de l'unité des secondes.La connexion Wifi et l'interrogation du serveur NTP s'effectue dans la fonction setup(), puis la connexion WiFi est fermée. La fonction loop() est uniquement dédié à l'attente et l'affichage. **Vérifiez avec votre voisin si vous êtes synchrone.**

   Modifier votre code pour mettre le processeur en veille légère entre deux changement d'affichage, en programmant correctement le timer pour un réveil toute les secondes pour le changement d'affichage. **Vérifiez à nouveau avec votre voisin si vous êtes synchrone.**

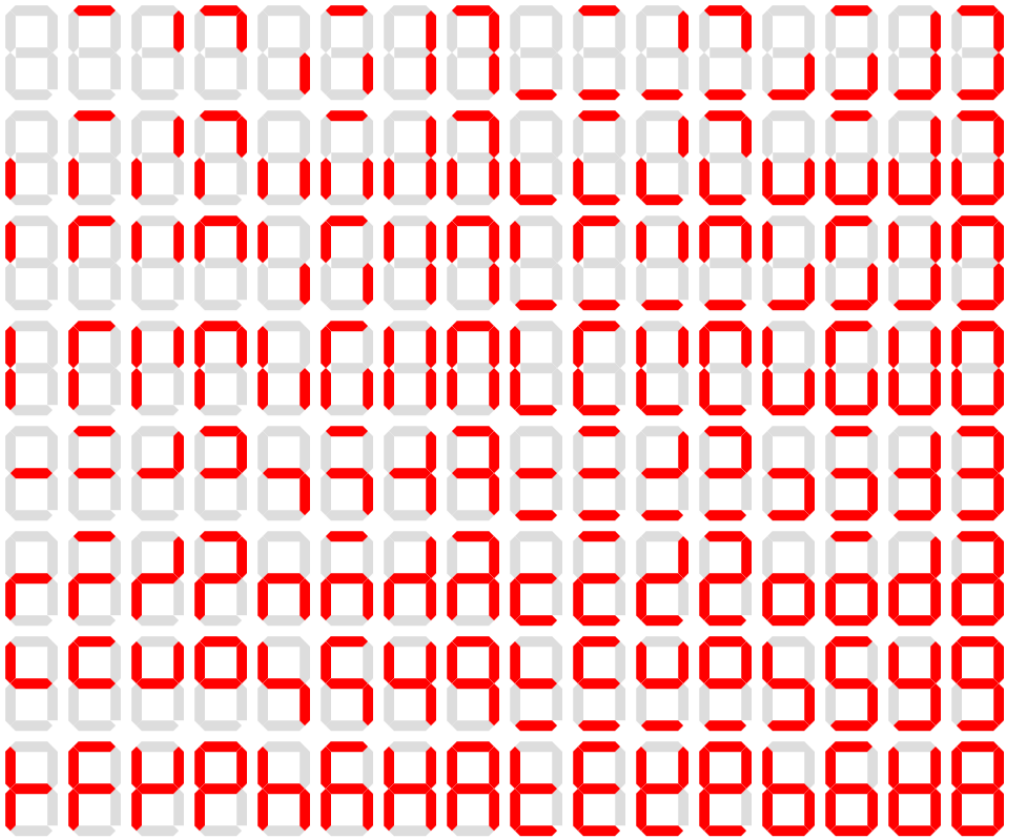
2. Modifier votre code afin de mettre en place un compte à rebours synchrone sur l'heure NTP : - Dans le setup(), vous établissez la connexion au Wifi et l'interrogation du serveur NTP.
   - Dans la fonction loop(), vous devez ensuite déterminer le nombre de secondes jusqu'au prochaine multiple de 5 minutes (par exemple, si l'heure courante est 11 :53 :24, l'heure correspondant au prochain multiple de 5 minutes est 11 :55 :00, il reste 1 minutes et 36 secondes à attendre, soir 96 secondes. Mettre le processeur en veille légère pendant 96 secondes et lors du réveil, jouer la "marche impériale" cf. TP1
   - Pour que l'opération puisse se répéter, il faut, après la fin de mélodie, recalculer le délai de mise en veille.

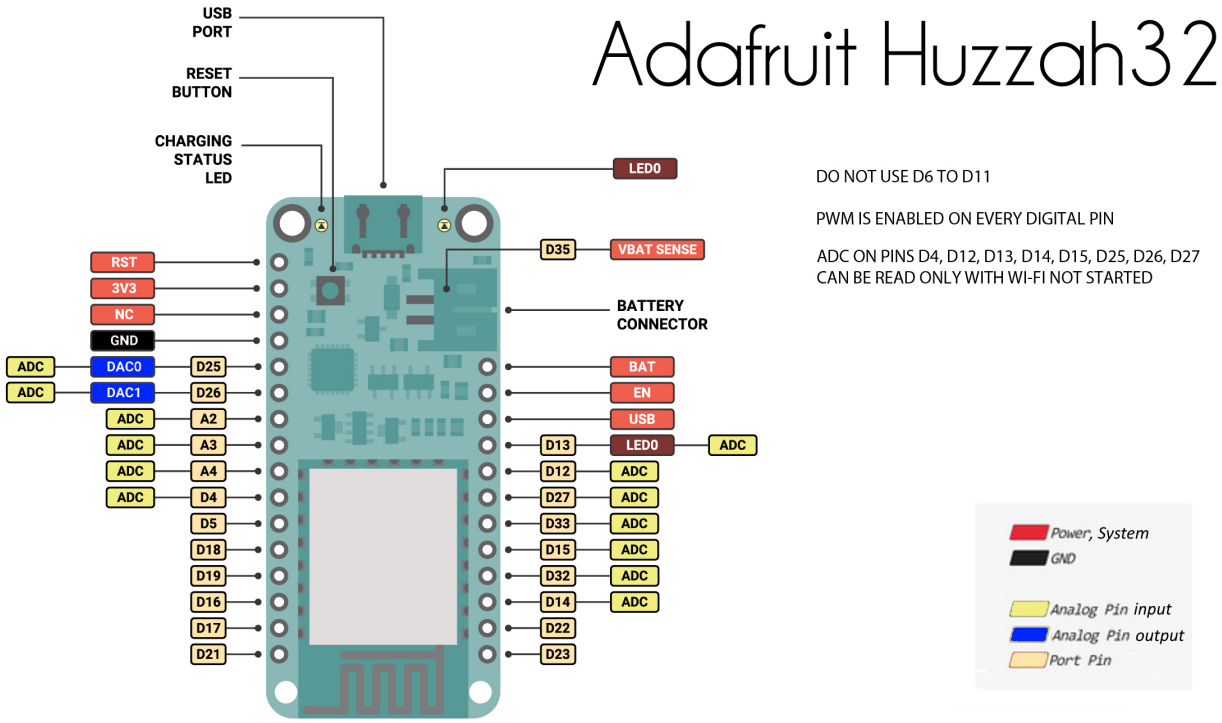
   Travail avec initiative personnelle : Améliorez votre programme pour que l'afficheur serve d'indicateur pour ce compte à rebours. vous pouvez faire preuve d'imagination et utiliser les segments de façon "détournée" pour afficher le délai restant, ou afficher alternativement les minutes puis le secondes, etc... N'oubliez pas : vous pouvez aussi utiliser le point décimal de l'afficheur 7 segments (à faire clignoter au rythme des secondes par exemple, il faut penser à le câbler). Il vous faudra éventuellement composer un nouveau tableau de correspondance entre les symboles que vous souhaitez afficher et les segments à allumer.
   - Évidemment, essayez de rester le plus longtemps possible dans un état de veille légère (*light sleep*).

   ☑ Déposez sur Moodle le code correspondant à loop(), avec suffisamment de commentaires pour comprendre la façon dont l'affichage est traité. Vous pouvez aussi ajouter un texte d'explication et/ou une courte séquence vidéo (attention à la taille maximale du fichier pour les *uploads*)

Annexes




Adafruit Huzzah32
USB PORT
RESET BUTTON
CHARGING STATUS LED
LED0
D35
VBAT SENSE
BATTERY CONNECTOR
RST
3V3
NC
GND
ADC DAC0 D25
ADC DAC1 D26
ADC A2
ADC A3
ADC A4
ADC D4
D5
D18
D19
D16
D17
D21
BAT
EN
USB
D13 LED0 ADC
D12 ADC
D27 ADC
D33 ADC
D15 ADC
D32 ADC
D14 ADC
D22
D23
DO NOT USE D6 TO D11
PWM IS ENABLED ON EVERY DIGITAL PIN
ADC ON PINS D4, D12, D13, D14, D15, D25, D26, D27 CAN BE READ ONLY WITH WI-FI NOT STARTED
Power, System
GND
Analog Pin input
Analog Pin output
Port Pin