C5-160551-INFO
Objets Connectés :
Programmation Microcontrôleurs - TP1

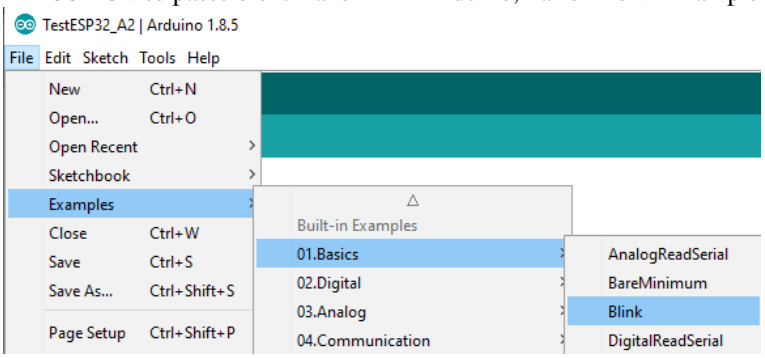
©Bernard Besserer

année universitaire 2021-2022

1 Tests : Environnement de développement et platines

Pour cette partie pratique, on utilisera de préférence l'environnement de développement (IDE) Arduino ou bien Visual Studio Code avec l'extension Plateform IO. Si vous travaillez avec votre ordinateur personel, l'IDE doit être installé et configuré (ajout des *boards* ESP32) préalablement à la séance de TP. Une procédure d'installation et de configuration de cet environnement est disponible sur Moodle.

Le "Hello World" du monde des microcontrôleurs, c'est le programme "blink" qui fait clignoter une LED, et qui permet de vérifier que la chaîne CODE SOURCE → COMPILATION → ÉDITION DE LIENS → TÉLÉVERSEMENT → EXÉCUTION se passe bien. Dans l'IDE Arduino, Faire File -> Example -> Basics -> Blink.



- Enregistrer le fichier sous un autre nom, c'est préférable.
- Sélectionnez la bonne carte (Tools -> Board, choisir la carte Adafruit Feather32 , choisir le port COM (il faut bien sur que la platine ESP32 soit branchée). **N'oubliez pas de faire cette opération lors des TP suivants**
- Compilez et téléversez ( ). Le code est très simple et doit faire clignoter la LED interne présente sur la platine.

A faire Le programme ci-dessous reprends le code de "Blink" et y ajoute l'initialisation de la liaison série permettant de dialoguer avec le PC. **Complétez ce programme** : Il faut y ajouter un compteur pour compter le nombre de clignotements et l'afficher sur le terminal série.

Pour voir la fenêtre du "terminal série", faire Outils -> Moniteur Série. Vérifiez que le terminal soit réglé sur la même vitesse de transmission (115200 bauds) que la vitesse de vous definissez dans votre code. **Laissez passer quelques clignotements et faites une copie d'écran montrant à la fois le code et la fenetre du terminal série. Uploadez cette copie d'écran.**

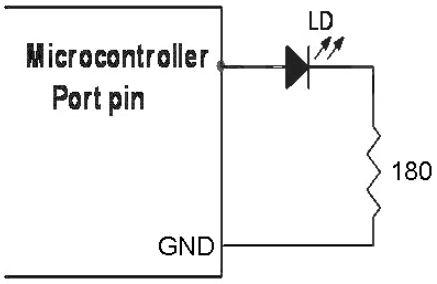
```
#include <Arduino.h>                    // only needed if you are using VS code + Platform IO
void setup() {                          // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Blink !");
  pinMode(LED_BUILTIN,OUTPUT);
}

void loop() {                           // put your main code here, to run repeatedly:
  digitalWrite(LED_BUILTIN,HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN,LOW);
  delay(500);
  Serial.println("Loop number :");
}
```

2 Entrées / Sorties numériques

2.1 Utilisation d'une LED Externe

Maintenant, placez la platine ESP32 sur une plaquette d'expérimentation
Inserer photo
Câbler une LED externe selon le schéma donné ci-après.



- A vous de choisir la broche pour connecter votre LED externe (sauf la 13). Consulter le brochage de la platine ESP32 a la fin de ce document. N'oubliez pas d'initialiser correctement le port(s) d'E/S choisi dans votre programme.
- Une LED est un composant polarisé, avec une Anode (+) et une Cathode (-). Il faut donc respecter le sens de branchement ; mais pour éviter les problèmes vous pouvez utiliser les LEDs rectangulaires qui regroupent dans le même boîtier deux LEDs "tête-bêche" : quelque soit le sens de branchement, il y en une qui s'allume. La résistance de protection **obligatoire** aura une valeur a choisir entre 100 à 500 Ohm (brun pour la 3eme bague).

☑ Faites clignoter la LED externe en **opposition** par rapport à la LED interne (LED externe allumée lorsque LED_BUILTIN éteinte). Que constatez vous ?. Donnez une brève explication sur l'espace prévu sur Moodle.

2.2 Configuration d'une broche en entrée

Branchez votre bouton poussoir entre une broche de votre choix et la masse GND (vous pouvez débrancher la LED externe, on utilisera la LED build in pour la suite, et donc réutiliser la broche).
Le programme que vous devez écrire devra :
- Dans le setup, configurer correctement la broche choisie (on va la nommer X) en entrée : `pinMode(X,INPUT_PULLUP);`
- Dans loop(), **lire** l'état de cette broche (`value = DigitalRead(X)`, on tourne dans une boucle tant que la valeur lue `value` n'est pas LOW),
- Si la valeur est LOW (donc appui sur le bouton), faire flasher la LED (deux impulsions de 400ms, séparé par 200ms).

Si votre programme fonctionne, remplacer `pinMode(X,INPUT_PULLUP);` par `pinMode(X,INPUT);` (on n'utilise donc plus la résistance de tirage interne).
Vérifiez si le fonctionnement est toujours satisfaisant. Si ce n'est pas le cas, rétablissez un fonctionnement correct en ajoutant une résistance tirage externe de l'ordre de quelques dizaines de KOhms (3eme bague orange ou jaune).

3 Systèmes séquentiels

3.1 Contraintes temporelles sur la prise en compte des entrées

Reprendre la structure de code utilisé pour la section 2.2, la version avec la résistance de tirage interne. Modifiez votre code afin qu'un appui sur le bouton allume la LED et un second appui l'éteigne. Cela semble simple, mais... on vous laisse chercher...
Pour vous aider à le résoudre : n'oubliez pas que l'appui sur le bouton (temps pendant lequel la broche sera à l'état bas) durera quelques dizaines ou centaines de milliseconde et que pendant ce temps, loop() aura le temps de

☑ s'exécuter des milliers de fois...
Enfin, si votre programme fonctionne, déposez votre code sur Moodle.

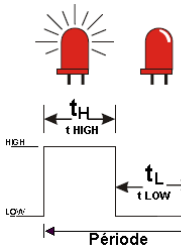
3.2 Action re-déclenchable

L'appui sur le bouton doit allumer la LED pendant 5 secondes. Tout appui sur le bouton durant cette période doit être pris en compte et la LED restera allumé pendant 5 secondes à partir du dernier appui sur le bouton.
Cet exercice nécessite une gestion du temps autre que l'utilisation de `delay()` (`delay()` est une fonction bloquante !). Pensez plutôt à utiliser la fonction `unsigned long millis()` (la valeur renvoyé par cette fonction correspond au temps de fonctionnement en millisecondes, depuis la mise sous tension de la platine). Déposez le

☑ code sur Moodle. **Vous trouverez une ressource intéressante ici** :
`https://www.carnetdumaker.net/articles/la-gestion-du-temps-avec-arduino/`

4 Actions séquentielles simples, fonction bloquantes

4.1 Clignotement contrôlé



Ecrire une fonction void `myFlash(int period, int nb)` qui fait clignoter `nb` fois la LED en respectant une période `period` qui sera exprimée en millisecondes. On rappelle que la période correspond à la durée du cycle, donc la somme $t_H + t_L$. Effectuer un test : un appel à `myFlash(2000,5)` doit faire 5 clignotements, la LED étant allumée pendant 1sec et éteinte pendant 1sec.

- On va utiliser la liaison série pour transmettre le nombre de clignotements, saisi par l'utilisateur sur le terminal, à la fonction myFlash. Dans `loop()`, ajouter les lignes :

```
int nombre
while(!Serial.available());     // attente bloquante d'une frappe clavier dans le terminal
nombre = Serial.parseInt();     // si le code ASCII saisi est un entier, alors il est converti
myFlash(2000,nombre);           // appel de la fonction
```

  Attention, on ne récupère qu'un seul chiffre ici. Si vous voulez 42 clignotements, ici c'est le chiffre 4 que vous récupérez.
- Mettez la saisie depuis la liaison série en commentaire. Ecrire un programme qui fait appel à `myFlash` pour émettre un SOS en morse, soit 3 points (clignotement court), 3 traits (durée de 3× la durée du point) et 3 points.
- Modifier le code afin de déclenchez le SOS que sur l'appui sur le bouton poussoir externe. Présentez votre SOS version lumineuse à votre intervenant. ☑
- Modifiez myFlash(period, nb) en myFlash(period, duration). Dans cette version, la LED doit clignoter avec la periode `period` et cela pendant une durée `duration` ; le nombre de cycles correspond donc au calcul duration/period que votre programme devra exécuter. Cela semble simple mais... on ne peut effectuer qu'un nombre **entier** tours de boucle (et pas 3.14 tours de boucle). Testez avec le morse. myFlash(1000,1000) doit clignoter 1x (LED allumée pendant 0.5sec puis éteinte pendant 0.5sec) et myFlash(3000,3000) doit aussi clignoter 1x, mais la LED devra rester allumée 1.5sec (puis eteinte 1.5 sec)

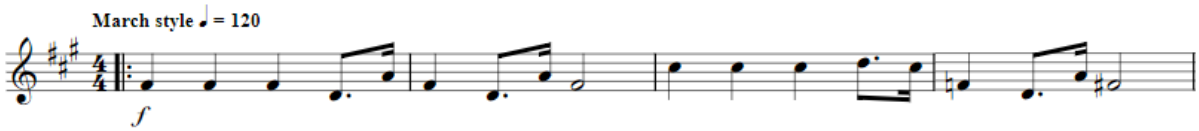
4.2 Version sonore

Évidemment, vous n'avez pas le droit d'utiliser la fonction tone() du framework !
Dupliquez votre fonction `void myFlash(int period, int duration)` et renommez-la `void myTone(int period, int duration)`. Au lieu de piloter une LED, nous allons piloter un buzzer.
- Il faudra adapter la commande, le buzzer n'est puissant que s'il est connecté entre deux broches et que ces broches sont commandées en opposition (cf. TD). Branchez le buzzer entre deux broches utilisables en sortie. N'oubliez pas d'initialiser ces broches.
- Il faudra adapter la période (qui est, rappelons-le, l'inverse de la fréquence) : pour que le signal soit audible, il faut une fréquence entre 200Hz et 10000Hz. La période est donc très courte. Remplacer dans votre code `delay()` par `delayMicroseconds()` et toute les valeurs seront alors exprimés en microsecondes.

La fonction myTone nécessite moins d'une dizaine de lignes de code. Pour tester, émettez par exemple un son de 440Hz pendant 1 sec. La periode est de 1/440 = 0.00227227 sec soit 2272 microsecondes. L'appel de la fonction sera alors : `myTone(2272,1000000)`. Vous pouvez aussi écrire une fonction `myTone (int freq, int duration)` en calculant la période dans la fonction, mais attention à ne pas vous tromper avec les unités (seconde, millisecondes, microsecondes ! ! !). Quand cela fonctionnera et que le signal est audible, deposez le code sur Moodle. ☑
Après un l'appui sur le bouton poussoir, votre microcontrôleur devra donc émettre un SOS dans sa version sonore. La différence entre un point et un trait ne se fait plus dans la durée, mais la hauteur (fréquence) du signal audio : soit 3 sons aigus (disons 4KHz) avec une durée de 0.5sec (ne pas oublier une pause entre chaque point), suivi de 0.5sec de pause puis de 3 sons graves (disons 500Hz) d'une durée de 0.5sec, suivi de 0.5sec de pause, puis à nouveau 3 sons aigus.

4.3 Synthèse sonore

Enfin, écrivez un programme principal qui interprète un thème musical bien connu après l'appui sur le bouton-poussoir(voir ci-après pour la partition). Si la fonction `myTone` est déjà adaptée pour recevoir en paramètre une fréquence et une durée, vous pouvez ranger les valeurs dans deux tableaux et déplacer un index. L'appel de la fonction `myTone` ressemblera à : `myTone(note[i],duree[i]);` Faites écouter à l'intervenant. ☑
Vous trouverez une ressource intéressante ici :
`https://openclassrooms.com/courses/perfectionnez-vous-dans-la-programmation-arduino/generez-des-sons`

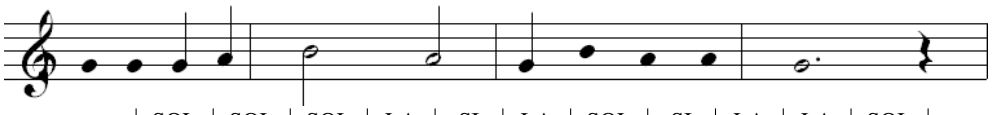


Note	FAd	FAd	FAd	RE	LA	FAd	RE	LA	FAd	DOd	DOd	DOd	RE	DOd	FA	RE	LAd	FAd
Fréquence	370	370	370	293	440	370	293	440	370	554	554	554	587	554	349	293	466	370
Durée (nombre de 1/8 de sec)	4	4	4	3	1	4	3	1	8	4	4	4	3	1	4	3	1	8

soit, pour un tableau contenant les fréquences des notes :

```
const int note[] = {370,370,370,293,440,370,293,440,370,554,554,554,587,554,349,293,466,370};
```

ou alors :



Note	SOL	SOL	SOL	LA	SI	LA	SOL	SI	LA	LA	SOL
Fréquence	392	392	392	440	493	440	392	493	440	440	392
Durée (en nombre de 1/8 de sec)	4	4	4	4	8	8	4	4	4	4	12

A Brochage de la platine Huzzah32 de Adafruit