C5-160551-INFO
Objet Connectés : Programmation microcontrôleur
TD4

©B. Besserer

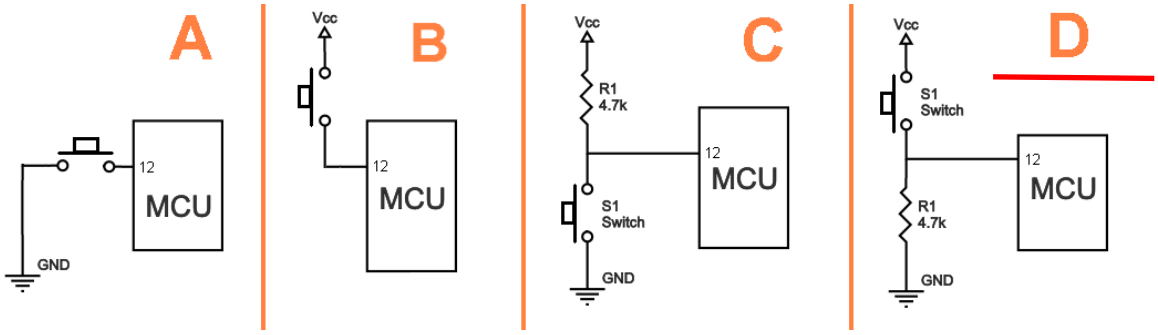
année universitaire 2021-2022

1 Objets connectés : sortie de veille

1. Soit un bouton poussoir relié au GPIO 12. A chaque action sur le bouton, on appelle `do_something()`, fonction qui nécessite un certain temps pour se terminer — on ne considère donc pas les rebonds. Le code proposé est donné ci-dessous.

```
01        void setup()
02        {
03          pinMode(12,INPUT);
04        }
05        void loop()
06        {
07          while (DigitalRead(12) != HIGH);
08          do_something();
09        }
```

   a) Quel est le bon branchement pour le bouton poussoir ? ☐

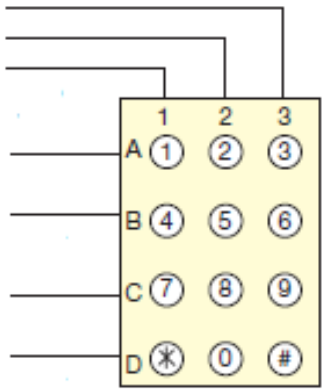


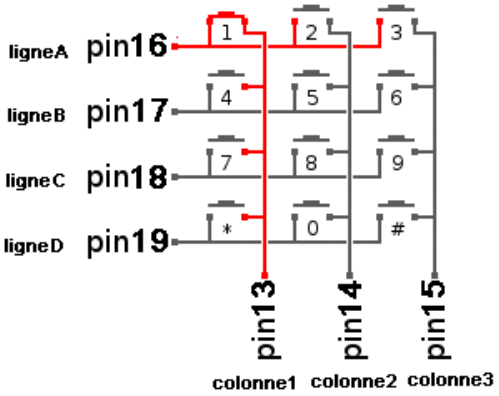
   b) en mettant `pinMode(12,INPUT_PULLUP);` au lieu de `pinMode(12,INPUT);` quel montage peut convenir pour le code ci-dessus et le fonctionnement souhaité.

2. On décide de ne plus utiliser une attente active, mais de mettre le processeur en veille légère (voir cours pour placer l'ESP en *light sleep*). Pour sortir le processeur de veille avec un changement d'état d'un GPIO, il faut déclarer cette broche comme une condition de réveil. **A la sortie de veille légère, le CPU exécute l'instruction qui se situe après l'instruction de mise en veille.**
   Concernant l'ESP32, le framework Arduino propose deux méthodes pour déclarer les conditions de sortie de veille suite à changement d'état de GPIO(s). La première méthode est à utiliser si un le changement d'état **d'un seul** GPIO provoque la sortie de la veille :
   `esp_sleep_enable_ext0_wakeup(RTC_PIN, MODE)`
   Avec le framework Arduino, il faut écrire `RTC_PIN` comme GPIO_NUM_X avec X le numéro de la broche (sont autorisés : 0,2,4,12,13,14,15,25,26,27,32,33,34,35,36,39).
   `MODE` = HIGH ou LOW selon l'état logique qui doit provoquer la sortie de veille.
   Soit le code donné au 1.1. Pour avoir un comportement identique en utilisant le *light sleep*, il y a donc :
   une ligne à ajouter au setup : ☐
   une ligne qui remplace la ligne numéro 07 : ☐

3. On passe maintenant au mode veille profonde (*deep sleep*). Dans ce mode, la sortie de veille est assimilé à un reset, et on repasse par l'appel de la fonction setup(). Le nommage de la broche GPIO qui permet de sortir de veille est identique. Re-écrivez le code en utilisant le mode *deep sleep* si possible, sur feuille libre et/ou Moodle. Que se passe t'il lors de la mise en service du microcontrôleur ?

2 Interface pour objet connecté : Clavier matriciel





Soit un clavier à 12 touches (par ex. ouverture de porte Digicode). Évidemment, on ne va pas consacrer 12 GPIO pour décoder chaque touche individuellement. En général, le clavier est organisé comme une matrice (ici, 4 lignes et 3 colonnes). Lors de l'appui sur une touche, la ligne et la colonne correspondante sont mises en contact. Dans le cadre de cet exercice, on exclut l'action simultanée sur deux touches.

1. Imaginons dans un premier temps **que les lignes A,B,C et D soient reliés à GND**, et que les colonnes 1, 2 et 3 soient reliés aux GPIO 13,14 et 15. Lors de l'appui sur une touche, une des ces colonnes va donc être relié à GND durant l'appui.
   a) Comment initialiser les broches 13,14 et 15 ?

```
pinMode(13,INPUT);
pinMode(14,INPUT);
pinMode(15,INPUT);
```

   b) Si appui, quelle est l'information que l'on obtient :
   ☒ le numéro de la colonne correspondant à la touche enfoncée ☐ la ligne correspondant à la touche enfoncée
   c) Écrire les lignes de code permettant de scruter de façon continue (donc sans mode de veille) les GPIOs (du moins ceux qui sont pertinents) jusqu'à détecter un appui et d'indiquer la colonne correspondante . Répondre sur feuille libre et/ou Moodle (environ une demi-douzaine de lignes de code)
2. Supposons maintenant que les lignes A,B,C et D ne soient plus connectées à GND, mais aux GPIO 16,17, 18 et 19 qui sont initialisées en **sortie**. L'idée est d'effectuer un balayage complet, c'est à dire de changer cycliquement (toujours sans mode de veille) le niveau logique de A, puis B puis C et enfin D jusqu'à détecter un changement de niveau sur l'une des colonnes 13, 14 ou 15, afin d'identifier à la fois la ligne et la colonne qui sont mise en relation par l'appui sur une touche. Écrire le code qui correspond au balayage complet (soit deux boucles imbriquées), permettant d'identifier la ligne et la colonne concernée par l'appui sur une touche. Répondre sur feuille libre et/ou Moodle.
3. Mettre ce code dans une fonction waitKeyboard() qui est bloquante jusqu'à appui sur une touche du clavier, et si appui sur une touche, renvoie le code ASCII du symbole correspond à la touche (chiffres de 0 à 9, et '*' et '#', a mettre dans un tableau).
4. On souhaite maintenant utiliser le mode de veille jusqu'à ce que l'action sur une touche provoque le reveil de l'ESP32. Pour que le réveil soit possible **avec plusieurs sources**, il faut utiliser l'instruction :

   ```
   esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,condition);
   ```

   sachant que `condition` peut prendre deux valeurs :
   `ESP_EXT1_WAKEUP_ALL_LOW` : Sortie de veille lorsque **toutes** les entrées validées par masque binaire `BUTTON_PIN_BITMASK` sont à l'état LOW
   `ESP_EXT1_WAKEUP_ANY_HIGH` : Sortie de veille **dès qu'une** des entrées validées par masque binaire `BUTTON_PIN_BITMASK` passe à l'état HIGH

   a) Avec le schéma ci-dessus et si l'on suppose comme précédemment que les lignes A,B,C et D sont connectées à GND, sera t'il possible d'écrire un code autorisant la mise en veille legere (light sleep) et le réveil par appui sur l'une de touches (sans savoir pour l'instant laquelle)
   ☐ oui ☐ non
   b) Est-il possible de d'obtenir le fonctionnement désiré (veille legère, réveil dès que l'on appuie sur une touche) SANS AJOUTER DES RESISTANCES DE TIRAGE au schéma précédent (même en utilisant `pinMode(X, INPUT_PULLUP)` le cas échéant ?
   ☐ oui ☐ non

c) Écrire le code complet, en tenant compte, le cas échéant, de la modification du schema.

- Initialisation des GPIOs
- Mise au niveau HIGH des GPIOs pilotant les lignes A, B, C ,D
- Signaler quels GPIOs peuvent provoquer le réveil
- Partir en veille légère (light sleep)
- Au réveil, balayage pour savoir exactement quelle touche a été actionnée (le réveil et le code qui suit est exécuté en quelques microsecondes, l'utilisateur n'a pas encore relacé la touche). On souhaite un affichage sur le terminal série du texte "mise en sommeil" lorsque le *light sleep* commence, puis par le texte "sortie du sommeil par le bouton X" lorsque l'un des boutons est activé (par exemple avec le texte "sortie du sommeil par le bouton +" ou bien "sortie du sommeil par le bouton 2").

Consultez la page `randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/`