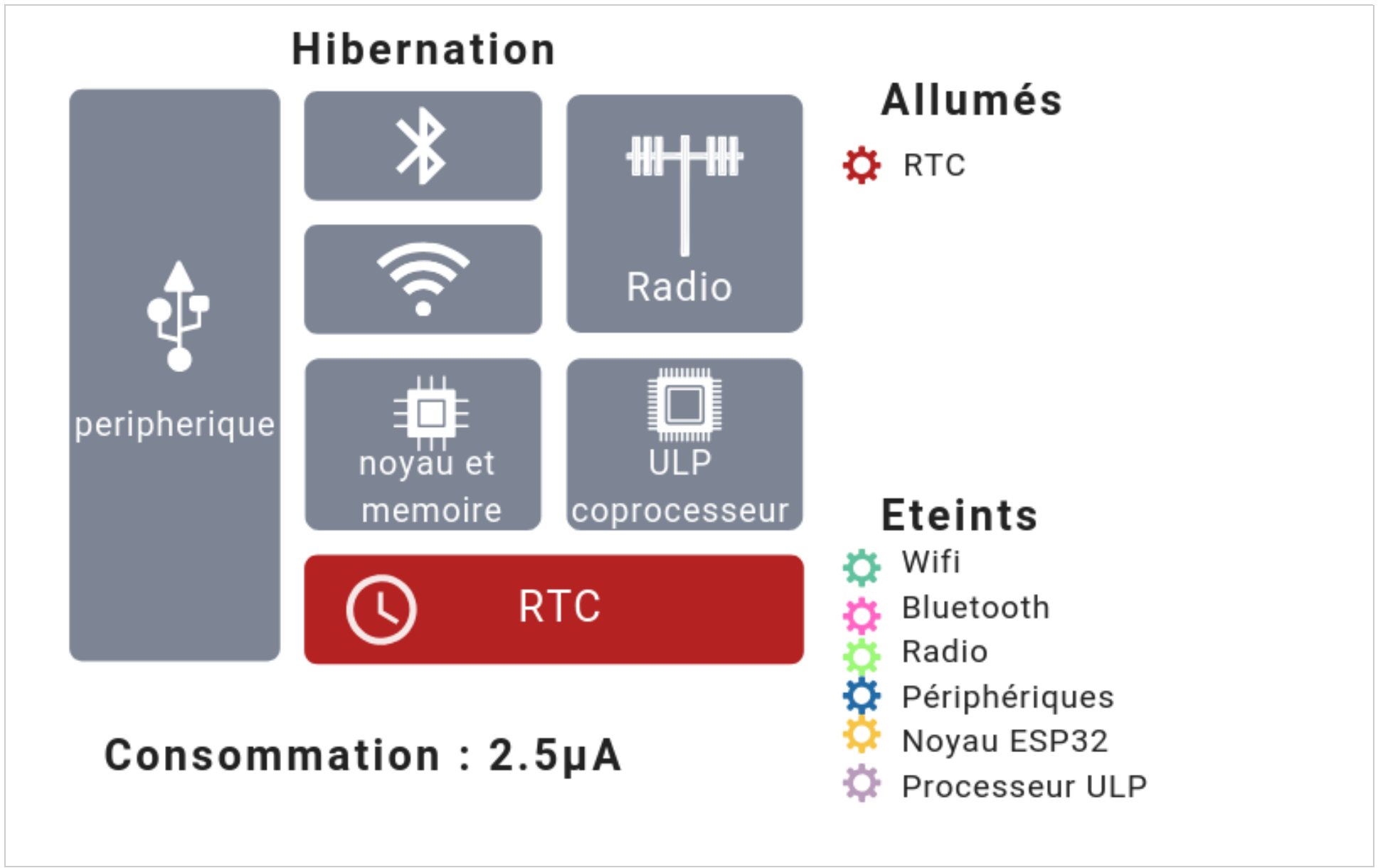
Letmeknow

[TUTO] : LES SLEEP MODES DE L'ESP32



Posté sur: août 8, 2019 | Catégories: Arduino | Tags: arduino , deep-sleep , esp32 , light-sleep , sleep-mode

Les sleep modes sont des états qui permettent de mettre en sommeil l'ESP 32. On éteint alors certains périphériques et les données sont conservées dans la RAM.

Il existe 5 modes d'économie d'énergies différents, nous allons les étudier un à un :

- mode actif (par défaut)
- mode modem
- mode light speed
- mode sommeil profond
- mode hibernation

Le mode actif : normal

Dans ce mode tous les périphériques sont allumés.

Le mode modem : WiFi, Bluetooth et radio désactivés

Pour maintenir les connexions WiFi / Bluetooth vivantes, le processeur, le Wi-Fi, Bluetooth et la radio sont réactivés à des intervalles prédéfinis. Ces modules sont responsables de la majeure partie de la consommation de la carte, c'est pourquoi nous allons chercher à les désactiver.

Pendant ce mode de veille, le mode d'alimentation bascule entre le mode actif et le mode veille du modem.

L'ESP32 peut entrer en mode veille du modem uniquement lorsqu'il se connecte au routeur en mode station (connexion sans fil).

L'ESP 32 reçoit à intervalle régulier des message DTIM (Delivery Trafic Indication Message) du routeur pour l'informer que la communication avec celui ci est possible.

Entre deux messages DTIM, les échanges de données avec le routeur sont impossibles, le WiFi et le Bluetooth sont donc désactivés durant ce laps de temps pour économiser de la batterie. Le temps de sommeil est généralement compris entre 100 et 1000 ms.

Light Sleep mode

Il comporte toutes les fonctionnalités du mode modem et plus encore. Ce mode permet d'économiser encore plus de batterie en utilisant le procédé de "Clock-Gating".

Ce phénomène est à l'oeuvre dans la RAM et le CPU où des parties du circuit sont mises hors tension pour empêcher les bascules flip-flop qui composent le circuit de changer d'état.

Ces bascules sont synchronisées par l'horloge interne de la carte (RTC). En mode actif, elles changent d'état (effectuent une action), à chaque coup d'horloge. En mode light sleep, ce n'est plus le cas, elles restent dans le même état.

C'est uniquement durant le changement d'état qu'il y a consommation, le clock gating permet donc d'économiser de la batterie.

Avant d'entrer en mode light sleep, L'ESP garde en mémoire les données et reprendra l'exécution du code là ou il s'est arrêté. L'avantage de ce mode est qu'il permet un réveil rapide tout en économisant de la batterie.

Pour observer le fonctionnement du mode light sleep, je vous propose de téléverser le code suivant et d'ouvrir le moniteur série.

```
void setup() {
 Serial.begin(115200);
 Serial.println("setup");
}

void loop() {
 esp_sleep_enable_timer_wakeup(5000000); //5 seconds
 int ret = esp_light_sleep_start();
 Serial.print("light_sleep:");
 Serial.println(ret);  }
```

Vous observerez que la carte ne reboot pas en sortie de sommeil et qu'elle reprend l'exécution du programme toutes les 5 secondes là où elle s'était arrêtée.

Deep Sleep

Pendant le deep sleep mode la majeure partie de la RAM et tous les périphériques numériques sont mis hors tension. Les seules parties de la puce qui restent sous tension sont les suivantes :

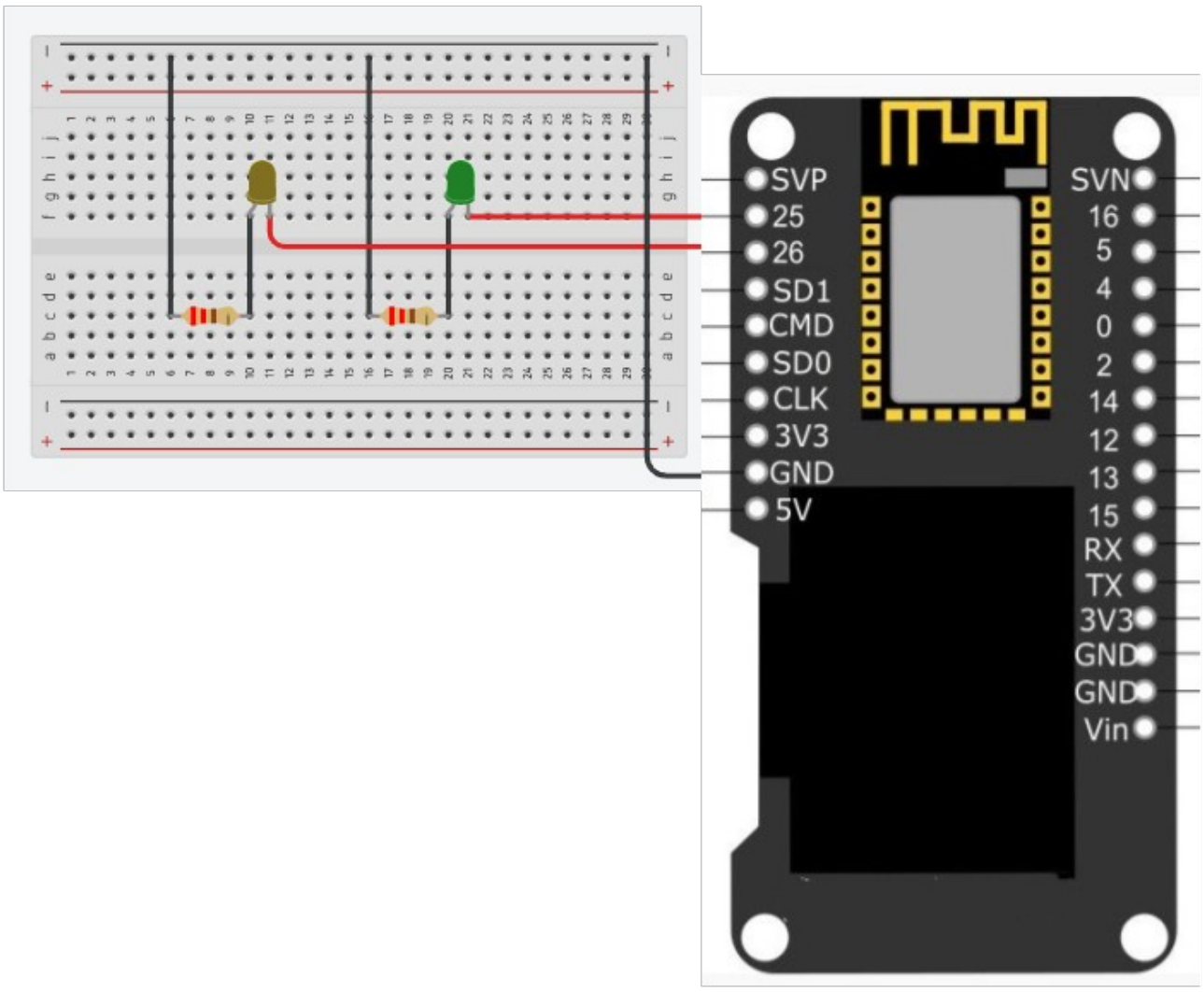
■ Contrôleur RTC
■ Périphériques RTC (y compris le processeur ULP)
■ Mémoires RTC (lentes et rapides).

Nous allons réaliser un montage visant à montrer que même en plongeant notre carte en sommeil profond, il est possible de sauvegarder des données.

Vous aurez besoin de :

■ 1 carte ESP32
■ 2 résistances 220 Ohm

- 2 LEDs (si possible de couleur différente)
- 3 fils de connexion
- 1 plaque d'essai (breadboard)



Montage à réaliser

Pour sauvegarder une variable même après avoir mis l'ESP en deep sleep mode il faut l'enregistrer dans la mémoire RTC en déclarant la variable en globale avec le type :

```
RTC_DATA_ATTR
```

Par exemple :

```
RTC_DATA_ATTR int bootCount = 0;
```

Copiez-collez le code suivant et téléversez le à l'aide de l'IDE arduino.

```
#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 3 /* Time ESP32 will go to sleep (in seconds) */
RTC_DATA_ATTR int bootCount = 0;
int GREEN_LED_PIN = 25;
int YELLOW_LED_PIN = 26;

void setup(){
 pinMode(GREEN_LED_PIN,OUTPUT);
 pinMode(YELLOW_LED_PIN,OUTPUT);
 delay(500);
 if(bootCount == 0) //Run this only the first time
 {
 digitalWrite(YELLOW_LED_PIN,HIGH);
 bootCount = bootCount+1;
 }else
 {
 digitalWrite(GREEN_LED_PIN,HIGH);
 }
 delay(3000);
 digitalWrite(GREEN_LED_PIN,LOW);
 digitalWrite(YELLOW_LED_PIN,LOW);
 esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
 esp_deep_sleep_start();
}

void loop(){     }
```

Contrairement au mode light sleep, la carte redémarre à chaque réveil et effectue donc à nouveau le setup.

Dans ce code, on instancie une variable bootcount qu'on stocke dans la mémoire RTC. La première fois que le setup est effectué, la led jaune doit s'allumer car bountcount est égale à 0.

En sortie de sommeil profond, Bountcount est égale à 1 car la variable a été incrémentée au tour d'avant. Cette fois ci, seul la led verte s'allume. La valeur de notre variable est conservée, la led jaune ne s'allume donc qu'une seule fois.

Mode Hibernation

Il désactive les mêmes fonctionnalités désactivées par le mode sommeil profond. A cela s'ajoute le fait que la mémoire RTC est aussi privée d'alimentation. Il en résulte qu'il est impossible de sauvegarder des données en utilisant ce mode d'économie d'énergie.

La consommation durant ce mode est censée être extrêmement faible (seulement quelques micro ampères)

Les façons de réveiller l'ESP32 :

Une fois mis en sommeil, il existe di﷿érentes façons de réveiller un ESP32. La déclaration de ces fonctions doit toujours se faire avant l'appel de la fonction provoquant le mode sleep. Cependant, vous pouvez toujours réveiller votre ESP32 en appuyant sur le bouton EN. Cela le redémarrera.

Timer :

Cette méthode peut être utilisée quelque soit le mode d'économie d'énergie activé. La carte se met en sommeil durant un temps défini par l'utilisateur et se réveille ensuite.

```
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP)// temps en micro seconde;
```

C'est cette méthode qui a été utilisée dans les exemples ci dessus.

Réveil externe :

Il est possible de réveiller un ESP32 après lui avoir envoyé un signal digital sur une de ces pin.

```
void setup() {
 Serial.begin(115200);
 Serial.println("setup");
}

void loop() {
 esp_sleep_enable_ext0_wakeup(GPIO_NUM_27,HIGH);
 int ret = esp_light_sleep_start();
 Serial.print("light_sleep:");
 Serial.println(ret);  }
```

Ce code permet de sortir du mode light sleep dès que la pin 27 reçoit un 1 logique. Si le signal repasse à 0, la carte retombera en mode sommeil.

Avec certains ajustements, il est aussi possible d'utiliser une interruption externe pour sortir du mode sommeil profond :

```
void setup() {
 Serial.begin(115200);
 Serial.println("setup");
}

void loop() {
 esp_sleep_enable_ext0_wakeup(GPIO_NUM_27,LOW);
 esp_deep_sleep_start();// la fonction ne retourne rien  }
```

On est plongé dans le mode sommeil profond tant que la valeur sur la pin 27 est à HIGH.

Vous êtes maintenant en mesure de comprendre comment les différents sleep modes de l'ESP32 fonctionnent et vous pouvez désormais les utiliser.