ARD5	ESP32 : Fonctions avancées	Durée 4h. Ind.

G.COLIN

Objectif : Passer en revue des fonctions avancées sur ESP32 pour répondre au cahier des charges des projets.

1 – Modifier les pattes attribuées à une liaison série

Les pattes des liaisons séries ont une attribution par défaut, mais il est possible de les modifier.

Exemple : Pour attribuer IO25 à U1RX et IO26 à U1TX

```
#include <HardwareSerial.h>
```
Inclure ce fichier

```
HardwareSerial MaSerie(1);
```
Déclarer un nouveau nom à la liaison série1 , en dehors des fonction

```
MaSerie.begin(9600,SERIAL_8N1,25,26);
```
Dans le setup(), initialiser la liaison - RX sur IO25, TX sur IO26

Pour tester

Relier U1TX sur U2RX et U2TX sur U1RX avec des câbles comportant une résistance en série pour éviter tout court-circuit (demander les câbles au prof.). U2TX est par défaut sur IO17, U2RX sur IO16.

⇨ Tester le programme suivant :

```
#include <HardwareSerial.h>

String ST1;

HardwareSerial MonGPS(1);

void setup() {
 Serial.begin(115200);
 MonGPS.begin(9600,SERIAL_8N1,25,26);
 MonGPS.setTimeout(10);
 Serial2.begin(9600);
 Serial2.setTimeout(10);
}

void loop() {
 Serial2.print("ENVOI");
 delay(100);
 if (MonGPS.available()){
  ST1=MonGPS.readString();
  Serial.print("Reçu sur 1 :");Serial.println(ST1);
  MonGPS.print("OK");
  delay(100);
 }
 if (Serial2.available()){
  ST1=Serial2.readString();
  Serial.print("Reçu sur 2 :");Serial.println(ST1);
 }
 delay(2000);
}
```

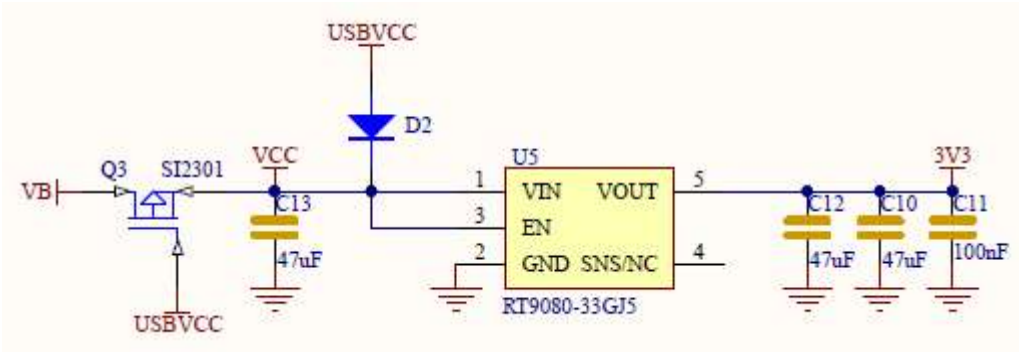
⇨ De la même manière, déplacer la liaison U2 sur IO12 (U2RX) et sur IO4 (U2TX).

Remarque : Tous les GPIO n'acceptent pas cette modification. C'est pour cette raison qu'il est important de faire des essais avant, en prenant les précautions nécessaires.

2 – Placer l'ESP32 en mode veille

Les essais doivent se faire sur une version FireBeetle (et non NodeMCU-ESP32) pour une des raisons suivantes :

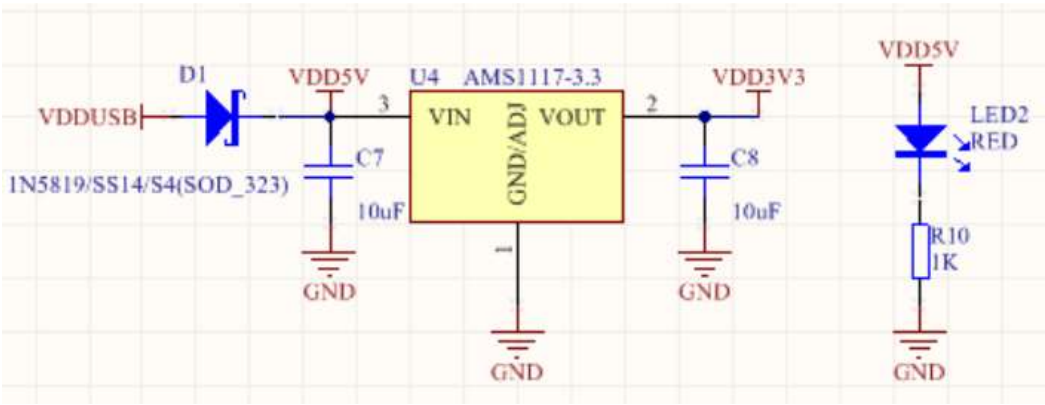
Sur la **FireBeetle** , le régulateur est un **RT9080**



Il consomme à vide : 2µA typique (4µA max.)

V_{CC} Consumption Current	I_Q	I_{LOAD} = 0mA, V_{OUT} ≦ 5.5V	--	2	4	µA

Sur la NodeMCU-ESP32, le régulateur est un AMS1117 :



Le courant à vide est de 5mA typique – Sans compter la LED2 alimentée en permanence.

Quiescent Current	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	(V_{IN} - V_{OUT}) = 1.5V		5	11	mA

La version NodeMCU-ESP32 n'est donc pas destinée à une utilisation basse consommation (pile ou batterie).

⇨ Tester le programme suivant :

```
#include "esp_sleep.h"

int i;

void setup() {
 esp_sleep_enable_timer_wakeup(5000000);
 pinMode(LED_BUILTIN, OUTPUT);
 Serial.begin(115200);
 Serial.println("Setup");
}

void loop() {
 digitalWrite(LED_BUILTIN, HIGH);
 Serial.print("Réveil : ");
 Serial.println(i);
 i++;
 delay(3000);
 digitalWrite(LED_BUILTIN, LOW);
 esp_deep_sleep_start();
}
```

Validation du réveil par le Timer toutes les 5000000 µs

Mise en veille, mode DEEP SLEEP MODE

⇨ Indiquer :

- Si au réveil le µP exécute à chaque fois le setup()
- Si le contenu de la variable i est conservé.

Mesure du courant consommé :

Cette manipulation doit être réalisée avec un ampèremètre de laboratoire.

⇨ Proposer une méthode pour relever le courant consommée.

⇨ Demander le matériel nécessaire au prof.

⇨ Faire vérifier le câblage avant toute mise sous tension.

La manipulation se fera avec le prof.

⇨ Relever les courants en mode normal et en mode DEEP SLEEP MODE

⇨ Comparer par rapport aux informations de la documentation technique suivantes :

- Deep-sleep mode
  - The internal 8 MHz oscillator, 40 MHz high-speed crystal, PLL and radio are disabled.
  - The digital core is powered down. The CPU context is lost.
  - The supply voltage to the RTC core drops to 0.7V.
  - 8 x 32 bits of data are kept in general-purpose retention registers.
  - The RTC memory and fast RTC memory can be retained.
  - Current consumption: ~ 6.5 μA.
  - Wake-up latency: less than 1 ms.
  - Recommended for ultra-low-power infrequently-connected Wi-Fi/Bluetooth applications.

⇨ Modifier le programme pour le faire passer en mode LIGHT SLEEP MODE
esp_light_sleep_start(); à la place de esp_deep_sleep_start();

⇨ Débrancher l'alimentation du module, le reprogrammer avec la liaison USB et observer l'affichage sur le moniteur.

⇨ Indiquer :
- Si au réveil le µP exécute à chaque fois le setup()
- Si le contenu de la variable i est conservé.

⇨ De la même manière que précédemment, mesurer le courant consommé et comparer par rapport aux informations de la documentation technique.

- Light-sleep mode
  - The internal 8 MHz oscillator, 40 MHz high-speed crystal, PLL, and radio are disabled.
  - The clock in the digital core is gated. The CPUs are stalled.
  - The ULP co-processor and touch controller can be periodically triggered by monitor sensors.
  - Current consumption: ~ 800 μA.
  - Wake-up latency: less than 1 ms.

Rétention de variable en mode DEEP SLEEP :

En mode DEEP SLEEP, la mémoire RTC (RTC memory et fast RTC memory) est sauvegardée.

⇨ Tester de nouveau le mode DEEP SLEEP (1er cas) et déclarer la variable i en mémoire RTC de la manière suivante : RTC_DATA_ATTR int i;

⇨ Commenter le résultat.