Rapport de projet

## Choix des composants et schéma électronique

### Microcontrôleur

1. Le microcontrôleur de la gamme ESP32 intègre de manière native une antenne wifi et possède des librairies prêtes à l’emploi ainsi que plusieurs fonctionnalités pour son utilisation dans de multiples projets.

Support : UART/PWM/GPIO/ADC/I2C/I2S interface, temperature sensor, pulse counter

1. Support : UART port local upgrade and remote firmware upgrade (FOTA)

Il sera nécessaire de mettre en place un connecteur physique pour un premier flashage du firmware malgré qu’il soit compatible FOTA.

### Alimentation / Régulation de la tension

1. Une cellule Li-Po à une tension nominal de 3,7 volts.

La tension d’une batterie est constante jusqu’à son déchargement.

La tension d’alimentation d’un bus USB est de 5 volts.

La tension d’entrée de l’ESP32-C3-WROOM-02-H4 à une tension d’entrée variant de 3.0V à 3.6V

Pour alimenter notre ESP32-C3-WROOM-02-H4, il sera nécessaire d’utiliser un régulateur de tension.

1. En transmission wifi, on a une consommation d’environ 300 mA. Pour recharger correctement la batterie, on peut utiliser un buck ETA3425 car, il est adapté pour la recharge des batteries LI-PO et qu’il a une plage de tension adapté.

Bonus : Le LDO dissipe l’énergie excédante sous forme de chaleur, il est donc possible d’avoir un risque de trop forte chaleur et donc d’incendie. On peut utiliser le Régulateur de tension, LM317T même s’il est bien trop surdimensionné pour notre utilisation.

1. La consommation du composant de régulation en charge ou non est de 1uA + la perte sur les résistances.

### Capteur thermique

1. On peut utiliser le capteur thermique Si7051 qui à une plage de fonctionnement possible de 1.9 V à 3.6V qui est adapté à l’ESP32 avec une précision de ±0.1 °C. Le prix pour une pièce lors d’un achat de 100 pièces est de 1.5 €. Il est adapté à notre tension de travail et la tension en sortie que peut nous sortir l’ESP32 avec une grande plage de donnée de température détectable et une faible consommation électrique.

### Autres

1. Choix des composants de notre carte :
   1. Le connecteur USB-C : 640-USB4800-03-A, connecteur de base utilisé dans l’industrie.
   2. Pour l’interrupteur : SLW-883935-2A-D, un interrupteur de base qui permet une commutation d’état simple.
   3. Pour la LED, j’ai choisi une LED RGB qui permettra d’avoir toutes les couleurs dans deux LED :

*DESCRIPTION* : HSMF-C118

Modèle : 150120M183000

* 1. Pour le régulateur de tension de 5V en 3.3V pour alimenter l’esp32 on peut utiliser en un TPS7A21

1. Il sera nécessaire d’ajouter des résistances que l’on va devoir ajouter au schéma électrique, des capacités de découplages et pour de la sécurité, il est possible de mettre des diodes pour protéger contre le retour de courant.

### Choix de la batterie

1. Il sera nécessaire d’utiliser deux modes bien distincts :

* 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 :

* 1. Contrôleur RTC
  2. Périphériques RTC (y compris le processeur ULP)
  3. Mémoires RTC (lentes et rapides).
* 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 Traffic 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.

Source : <https://letmeknow.fr/fr/blog/142-tutoriel-les-sleep-modes-de-lesp32>

Dans une journée de 24 heures, notre IOT devra faire une mesure toutes les heures donc 24 fois dans une journée, on considère une minute pour faire une acquisition/étalonnage correcte ce qui nous fera 24 minutes.

Tx = *250 mA*

Rx = *90 mA*

Deep Sleep = *5 uA*

Mode modem = *20 mA*

Led RGB en moyenne = *5mA*

Consommation en veille = (Deep sleep \* 1416) / 60

= *1.18 \* 10¨^-4 mA pendant* 1416 min

Consommation en fonctionnement = ((Mode modem + Tx + Rx + LED RGB) \* 24) /60

= *152 mA pendant 24 min sur un heure*

Soit sur 7 jour :

(*1.18 \* 10¨^-4 mA + 152 mA) \* 7 = 1064 mA*

1. On peut donc prendre une batterie comme celle-ci avec une marge de sécurité sur la capacité d’environ 20% pour les éventuels pics de surconsommation :

LP902358 1280mAh 3.7V Rechargeable LiPo Battery with connector JST ZHR-3 B

### Recharge de la batterie

1. Il est nécessaire d’utiliser le MCP73831T-2DCI/MC qui peut délivrer une tension de 4.2 volts, tension qui permettra de recharger, la batterie Li-po à sa tension de recharge de 4.2 volts avec un courant de 500mA. Notre batterie mettra environ 1h30 pour se recharger, ce qui reste loin de nos 4 heures de recharges max.