

Projet-ESP32

Alexandre Cauty

January 2024

Contents

1	Introduction	3
2	Montage	3
3	Fonctionnalités	4
4	Gestion des périphériques	4
4.1	Température	4
4.2	Luminosité	5
4.3	L'écran et LED	5
4.4	Routes web server	7
4.5	Fonctions Outils	8
5	Résultats	9
6	Conclusion	9
7	Annexes	9

1 Introduction

Dans le cadre de l'UE HAI912I Développement mobile avancé, IOT et embarqué. Le but du projet était de créer une API REST sur une carte ESP32. Cette API doit être capable de récupérer les valeurs des capteurs connectés à l'ESP32. Les capteurs utilisés seront une thermo-résistance et une photo-résistance. Il doit être possible de contrôler l'allumage et l'extinction d'une LED. Dans ce projet, j'ai rajouté un buzzer pour la création d'une alarme.

2 Montage

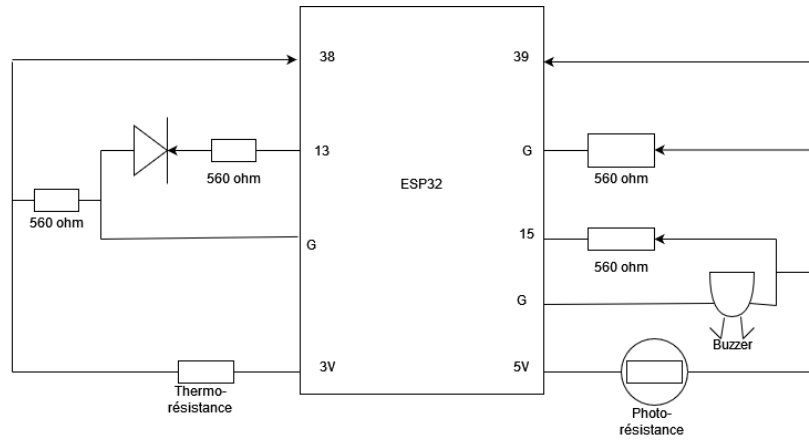


Figure 1: Schéma électrique du projet

Dans ce montage, le circuit se sépare en 2 parties : On retrouve la photo-résistance connectée à la broche 5V et à la broche 39 avec une résistance liée de 560 ohm à la terre. Un buzzer relié à la branche de la terre et à une résistance de 560 ohm lié à la broche 15.

Dans la seconde partie, on retrouve une thermo-résistance connectée à la broche 3V et à la broche 38. Une résistance de 560 ohm relie une led au circuit. La led est connectée à la terre et à une résistance 560 ohm qui est reliée à la broche 13.

3 Fonctionnalités

Voici les fonctionnalités disponible du projet à partir de chaque requête:

- Allumer la led
- Eteindre la led
- Récupérer les données de la photo résistance
- Récupérer les données de la thermo résistance
- La lumière s'allume ou s'éteint en fonction du seuil
- Augmenter le seuil
- Diminuer le seuil
- Afficher les capteurs utilisés

Pour chaque requête, un fichier JSON est créé dans le but de savoir ce qu'il y a été récupéré ou produit par la requête.

4 Gestion des périphériques

L'objectif était de récupérer les valeurs grâce à la thermo-résistance et la photo-résistance. Ces informations seront disponibles d'une part sur l'écran TTGO et d'une autre part l'API possède un deux liens qui retournent les valeurs sous format JSON.

4.1 Température

Pour la température, il a fallut convertir la tension en degrés. Voici le code qui permet cette conversion :

```
1  Vout = (analogtemp * VCC) / adc_resolution;  
2  Rth = (VCC * R2 / Vout) - R2;  
3  
4  float temperature = (1 / (A + (B * log(Rth)) + (C * pow((  
5      log(Rth)),3))));  
6  
   temperature = temperature - 273.15;
```

Listing 1: Code formule de conversion température

analogtemp correspond à la valeur de la tension où se situe la thermo-résistance. Cette formule a été trouvé sur internet. VCC correspond au voltage utilisé, dans ce cas 3V. R2 correspond à la valeur de la résistance associé à la thermo-résistance, dans ce cas 560 ohm. A,B et C correspondent à des constantes qui permettent de transformer la valeur de la résistance en degrés kelvin.

4.2 Luminosité

Pour la lumière, il faut convertir la valeur de la tension en Lux. Voici le code qui permet la conversion:

```
1 // Conversion analog to voltage
2 float Vout = float(raw) * (VIN / float(1024));
3 // Conversion voltage to resistance
4 float RLDR = (R * (VIN - Vout))/Vout;
5 // Conversion resistance to lumen
6 int phys=500/(RLDR/1000);
7 return phys;
```

Listing 2: Code formule de conversion lux

raw correspond à la valeur brut en V de la tension reçue par la broche. VIN correspond à la tension initiale, dans ce cas 5V. R correspond à la valeur de la résistance, dans ce cas 560 ohm.

4.3 L'écran et LED

L'écran affiche les valeurs qu'on a converti. La couleur du fond change en fonction de la valeur de la luminosité par rapport au seuil ou du mode.

```
1 // Affichage des donnees des capteurs et du seuil
2 tft.setCursor(0, 50, 2);
3 tft.print("Luminosite :");
4 tft.print(lux);
5
6 tft.setCursor(0, 70, 2);
7 tft.print("Temperature :");
8 tft.print(temperature);
9 tft.print(" C");
10
11 tft.setCursor(0, 90, 2);
12 tft.print("Seuil :");
13 tft.print(seuilAlarme);
```

Listing 3: Code pour l'écran

```

1  if(mode == 0)
2  {
3      tft.fillScreen(TFT_BLACK);
4      digitalWrite(ledPin, HIGH); // Allume la led
5  }
6  else if(mode == 1)
7  {
8      tft.fillScreen(TFT_BLACK); // Eteint la led
9      digitalWrite(ledPin, LOW);
10 }
11 else
12 {
13     if(lux > seuilAlarme)
14     {
15         tft.fillScreen(TFT_RED);
16         digitalWrite(ledPin, LOW); // Eteint la LED
17         myTone(BUZZER_PIN,100);
18     }
19     else
20     {
21         tft.fillScreen(TFT_GREEN);
22         digitalWrite(ledPin, HIGH); // Allume la LED
23         myTone(BUZZER_PIN,0);
24     }
25 }

```

Listing 4: Code condition de la led

On retrouve 4 états possibles pour la led :

- mode = 0, la led s'allume
- mode = 1, la led s'éteint
- la led s'éteint si la valeur en lux capturé par la photo résistance est plus élevée que le seuil initialisé à 1500 lux.
- la led s'allume si la valeur en lux capturé par la photo résistance est moins élevée que le seuil initialisé à 1500 lux.

4.4 Routes web server

La bibliothèque WebServer a permis de mettre en place un webServer sur le microprocesseur et la bibliothèque Wifi a été nécessaire pour la mise en réseau. Grâce à cela, des appareils externes ont la possibilité d'envoyer des requêtes à notre serveur. Une multitude de route a été mis en place pour pouvoir réaliser les différentes fonctionnalités proposées.

```
1 server.on("/on", allume);
2 server.on("/off", eteindre);
3 server.on("/light", getLight);
4 server.on("/temp", getTemp);
5 server.on("/AugmenteSeuil", AugmenteSeuil);
6 server.on("/DiminueSeuil", DiminueSeuil);
7 server.on("/getCapteurs", getCapteurs);
8 server.onNotFound(handleNotFound);
9 server.begin();
```

Listing 5: Code des routes web

Pour pouvoir tester toutes les routes, Postman a été utilisé pour envoyer des requêtes HTTP au serveur. Chaque lien http renvoie un fichier JSON contenant des fonctions en fonction de la requête.

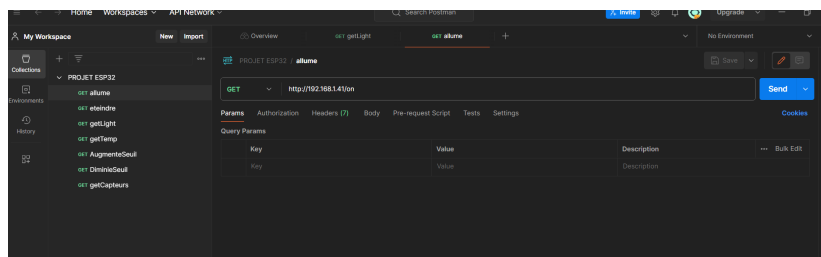


Figure 2: Dossier contenant les liens vers les requêtes

4.5 Fonctions Outils

Pour avoir un retour sous format JSON des informations sur la luminosité, température et les capteurs. Deux fonctions ont été conçues :

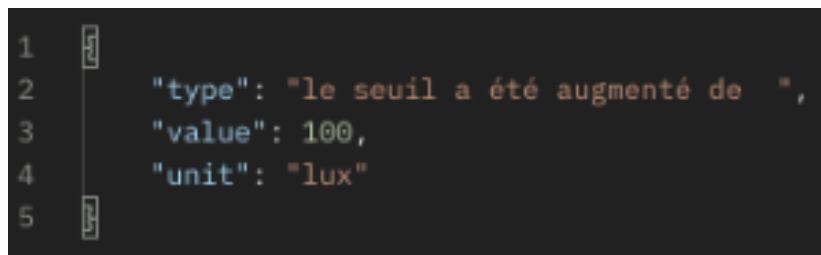
- L'une permet la création d'un fichier JSON
- L'autre permet de remplir les champs dans le fichier JSON existant

```
1 void create_json(char *tag, float value, char *unit)
2 {
3     jsonDocument.clear();
4     jsonDocument["type"] = tag;
5     jsonDocument["value"] = value;
6     jsonDocument["unit"] = unit;
7     serializeJson(jsonDocument, buffer);
8 }
```

Listing 6: Code création d'un JSON

```
1 void add_json_object(char *tag, float value, char *unit)
2 {
3     JsonObject obj = jsonDocument.createNestedObject();
4     obj["type"] = tag;
5     obj["value"] = value;
6     obj["unit"] = unit;
7 }
```

Listing 7: Code ajouter des champs au format



```
1 {
2   "type": "le seuil a été augmenté de ",
3   "value": 100,
4   "unit": "lux"
5 }
```

Figure 3: Exemple de sortie JSON : Utilisation de la requête AugmenteSeuil

On retrouve aussi une fonction qui permet d'activer un son avec le buzzer. Le buzzer permet de créer une alarme lorsque le seuil de luminosité est dépassé. Voici la fonction créée:

```
1 void myTone(byte pin, int freq)
2 {
3     ledcSetup(0, 2000, 8); // setup beeper
4     ledcAttachPin(pin, 0); // attach beeper
5     ledcWriteTone(0, freq); // play tone
6 }
```

Listing 8: Code émission d'un son

5 Résultats

Vous pouvez retrouver une vidéo dans le dossier qui donne l'aspect principal du projet.

6 Conclusion

Pour conclure, ce projet m'a permis la mise en place et le développement de système embarqué. Toutes les fonctionnalités demandées ont été développées et fonctionnelles. Après avoir programmé sur le micro-contrôleur ESP32, l'utilisation de la photo résistance, thermo résistance. Il serait intéressant de se pencher vers de nouveaux outils comme les caméras.

7 Annexes

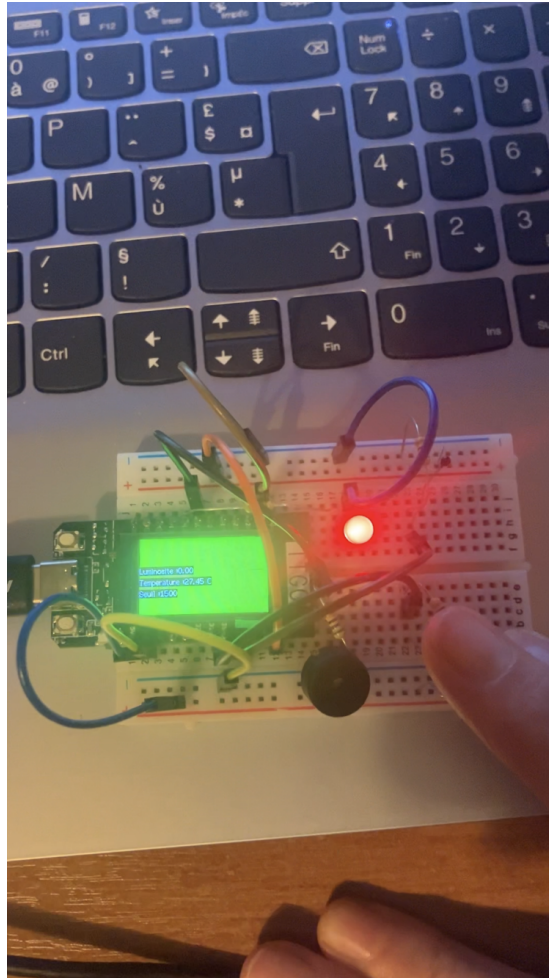


Figure 4: Projet

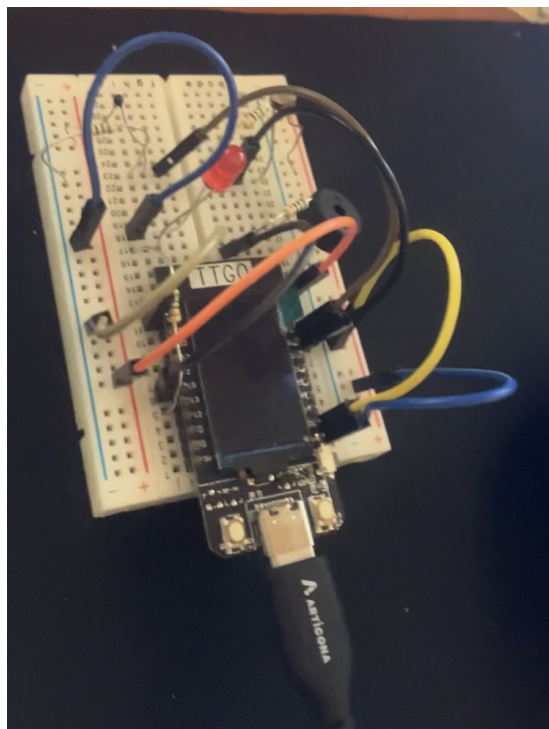


Figure 5: Projet