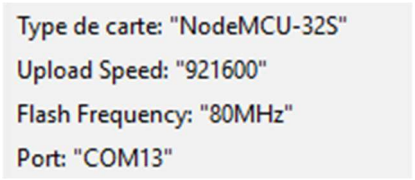
ARD1 ESP32 : Interface I/O et Interface analogique Durée 2h.

Document annexe : Implantation du module NodeMCU ESP32.
Ressources internet : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Reference
https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/index.html
Le développement se fait sous l'IDE Arduino. Choisir la carte NodeMCU-32S



Les projets sont à sauvegarder dans l'espace de travail sous KWARTZ, sous répertoire **Arduino_ESP32**.

1 – Clignotement d'une LED

La LED du kit est câblée sur GPIO2.

Configurer la vitesse du port série du moniteur
Envoyer "Démarrage" sur le moniteur
Configurer la patte reliée à la LED en sortie
TANT QUE toujours
| Allumer la LED
| Attendre 1000ms
| Eteindre la LED
| Attendre 1000ms
| Envoyer "Boucle" sur le moniteur
FIN
	void setup() {
	Serial.begin(115200);
	Serial.println("Démarrage");
	pinMode(LED_BUILTIN, OUTPUT);
	}
	
	void loop() {
	digitalWrite(LED_BUILTIN, HIGH);
	delay(1000);
	digitalWrite(LED_BUILTIN, LOW);
	delay(1000); // wait for a second
	Serial.println("Boucle");
	}

⇨ Tester le programme ci-dessus
⇨ Lancer le moniteur
⇨ Appuyer le bouton EN et conclure sur son rôle.

⇨ Modifier le nom "LED_BUILTIN" par LED0 et déclarer cette étiquette de la manière suivante en début de programme (2 pour GPIO2) : #define LED0 2
⇨ Tester

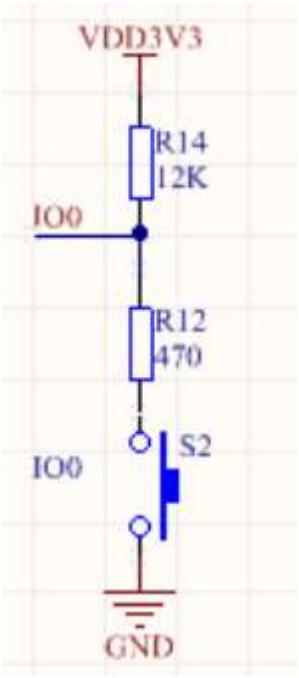
⇨ Proposer un montage pour relier une 2ème LED (LED1) sur GPIO4.
⇨ Réaliser le montage et faire vérifier par le prof.

⇨ Modifier le programme pour que la boucle principale réponde à l'algorithme suivant (clignotement alterné) :
(ne pas oublier de configurer la ligne du port)

TANT QUE toujours
| Allumer LED0
| Eteindre LED1
| Attendre 1000ms
| Eteindre LED0
| Allumer LED1
| Attendre 1000ms
FIN

Le bouton BOOT peut être utiliser comme bouton poussoir général (lorsque le µP tourne normalement).

Le bouton BOOT est câblé sur GPIO0, de la manière suivante :



GPIO0 doit alors être configuré en entrée. L'appui sur le bouton poussoir fourni un niveau bas.

⇨ Enregistrer le projet sous un autre nom (Bouton) et réaliser un programme qui réponde à l'algorithme suivant (boucle principale) :

TANT QUE toujours
SI bouton poussoir BOOT appuyé
Allumer la LED0
SINON
Eteindre la LED0
FIN
FIN

Notes :

Ne pas oublier de configurer les lignes de port en entrée ou en sortie
Déclarer le bouton par : #define BTN 0
La lecture de l'état logique se fait par la fonction : digitalRead(BTN)

⇨ Tester. Faire vérifier au prof.

2 – Convertisseur analogique numérique

Le µP ESP32 dispose de 2 CAN de 12 bits. Consulter la page suivante pour en savoir plus.
https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html#overview

Il va falloir câbler un potentiomètre sur l'entrée GPIO32, avec une plage de variation comprise entre 0 et 3V3.

⇨ Couper l'alimentation de la maquette (retirer câble USB) et câbler le potentiomètre fourni (le 3V3 est disponible sur le KIT).

⇨ Faire vérifier le câblage au prof.

⇨ Tester le programme ci-dessous et justifier la valeur maximale obtenue en modifiant la tension fournie par le potentiomètre.

```
#define POTAR 32

int VPOT;

void setup() {
  Serial.begin(115200);
}

void loop() {
  VPOT=analogRead(POTAR);
  Serial.print("Valeur:");
  Serial.println(VPOT);
  delay(1000);
}
```

Allumage automatique

On souhaite tester un montage pour réaliser un allumage automatique lorsque l'éclairage est faible.

Matériel nécessaire :

- Une photorésistance (qui diminue lorsque la lumière augmente).
- Une résistance de 1KΩ
- Une LED avec une résistance de limitation.

La photorésistance et la résistance de 1kΩ sont montées en pont diviseur de tension (photorésistance sur 3V3, et résistance de 1KΩ sur GND).
L'entrée analogique utilisée sera la patte GPIO35.
Fonctionnement : Lorsque que l'éclairage est faible (sous un certain seuil) on allume la LED, lorsque l'éclairage est fort on éteint la LED.
La LED est câblée sur GPIO4.

⇨ Dessiner le montage à main levée sur une feuille et le soumettre au prof.

Le programme principal est donné ci-dessous.

```
#define ECL 35
#define LED 4
#define ECL_FAIBLE 2200
#define ECL_FORT 2600

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
}

void loop() {
  allume_auto();
  delay(1000);
}

void allume_auto(void){
……..
}
```

⇨ Ecrire le sous-programme allume_auto() qui doit répondre à l'algorithme ci-dessous.
⇨ Tester et ajuster les niveaux ECL_FAIBLE et ECL_FORT si nécessaire.
⇨ Faire constater le fonctionnement au prof.

```
Convertir la tension du capteur
Envoyer le résultat sur le moniteur
SI éclairement < niveau faible
│       Allumer la LED
│       Envoyer "Allumage LED" sur le moniteur
FIN
SI éclairement > niveau fort
│       Eteindre la LED
│       Envoyer "Extinction LED" sur le moniteur
FIN
```