

NOTES

SCHEMA ELECTRIQUE ET PROGRAMME

Le programme ainsi que le schéma électrique sont présents chacun sous deux versions : Arduino et ESP32

Le schéma électrique peut se découper en trois parties :

- Capteurs
- Moteurs
- Debug

[PARTIE CAPTEURS]

La partie Capteurs est constituée de photorésistances (LDR) ainsi que de résistances, formant des ponts diviseurs de tension vers les ports d'entrées de l'ESP32. Le but de ce pont diviseur est d'obtenir une plage de tension idéale à l'entrée des ADC (convertisseurs analogiques -> numériques) de l'ESP, à savoir : éviter toute saturation, précision et stabilité. L'ADC de l'ESP renvoie un entier entre 0 et 4095. Afin de régler la tension renvoyée par le pont diviseur, et donc la sensibilité des capteurs, il suffit de changer les résistances : plus la résistance est grande, plus la tension l'est aussi et plus la plage numérique utilisée par les capteurs est grande. Attention cependant à ne pas mettre de trop grandes résistances, au risque de saturer le signal. Il est préférable de faire des tests avec un soleil direct sur les LDR en cas de réglage.

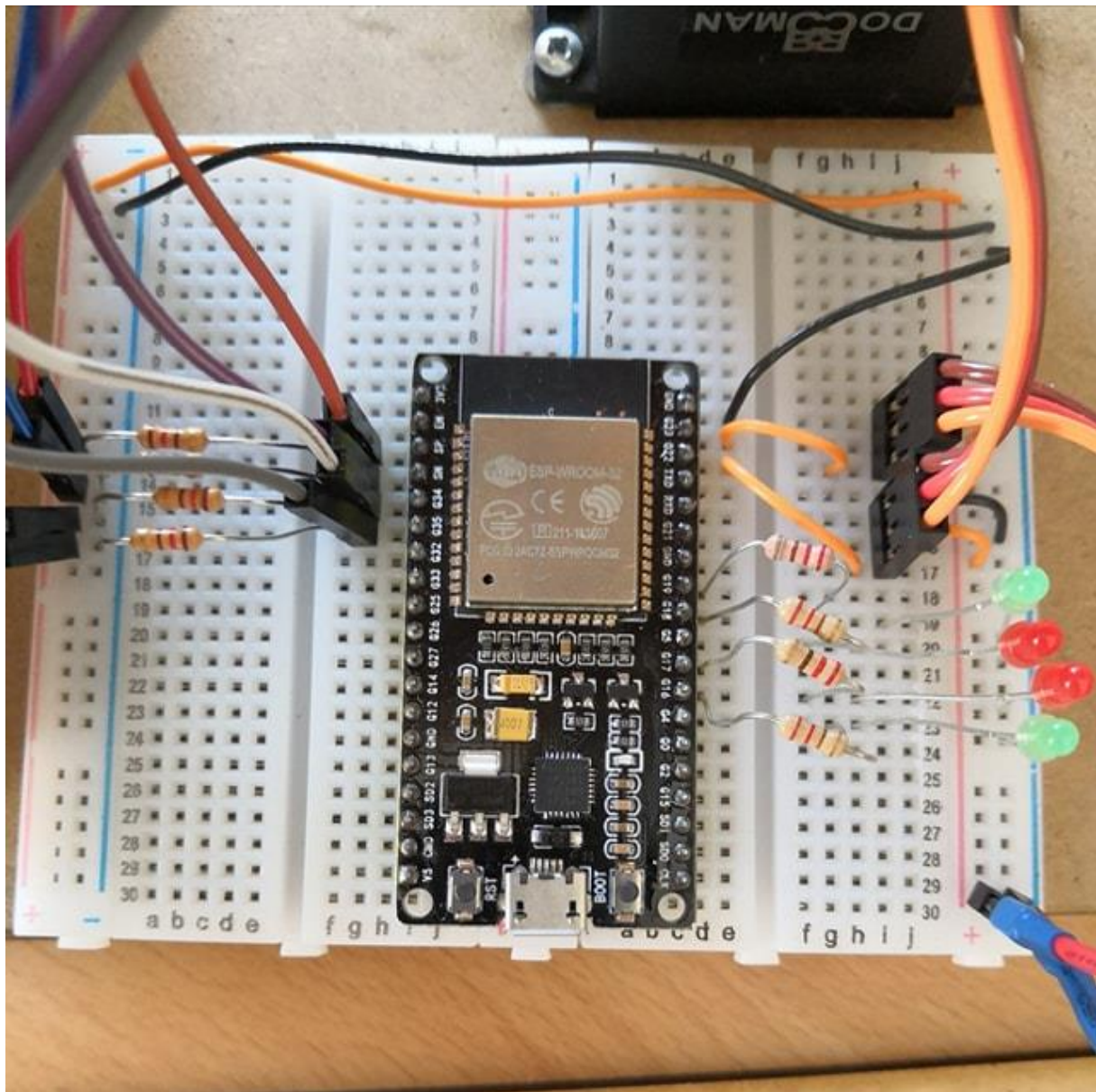
[PARTIE MOTEURS]

Composée de deux servomoteurs, les couleurs des câbles présents sur le schéma sont à respecter. Le programme impose des butées angulaires afin d'éviter toute collision et blocage mécanique.

[PARTIE DEBUG]

Cette partie du schéma est optionnelle, elle permet de visualiser avec les LEDs quels sont les LDRs qui reçoivent le plus de lumière. Les LEDs s'allument lorsque l'erreur entre les LDRs, définie dans le programme, est dépassée.

Photo du câblage avec l'ESP32 :



Différences Arduino / ESP32

	Arduino	ESP32
Plage numérique des ADC	0 – 1023	0 - 4095
Wifi	NON	OUI
Bluetooth	NON	OUI
RTC	NON	OUI mais utilisable ?

L'ESP32 peut aussi se programmer avec l'IDE Arduino, à condition d'importer les fichiers nécessaires et les bibliothèques compatibles (notamment la bibliothèque pour le contrôle des servomoteurs).

Préparer l'IDE Arduino pour l'ESP32

Inclure la carte ESP32 dans l'IDE Arduino : <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

Inclure la bibliothèque ESP32Servo : Croquis > Inclure des bibliothèques > Gérer les bibliothèques > Chercher « ESP32Servo » et installer.

Pourquoi un ESP32 et pas un ESP86 ?

Simplement à cause de l'utilisation de ports analogiques. L'ESP86 n'en possède qu'un alors que le programme actuel, utilisant les LDR, a besoin de 4 ports analogiques. En cas de portage sur un fonctionnement avec un RTC, il peut être envisageable d'utiliser un ESP86

Autres remarques

Il est important de relier les masses (GND / 0V) de tout le circuit, afin d'avoir une masse commune. Sinon les signaux risquent d'être mal interprétés et le circuit ne fonctionnera pas.

LE PROGRAMME

Le principe du programme est d'identifier, à une erreur tolérée près, la photorésistance qui reçoit le plus de lumière pour chaque paire (haut-bas et gauche-droite). Il est possible de modifier l'erreur afin d'obtenir un système plus ou moins précis et donc nerveux.

Si les LEDs de debug ne sont pas utilisées, il n'est pas nécessaire de les retirer du programme.

Le programme se décompose en plusieurs sous-fonctions :

- Asserv() constitue la fonction principale d'asservissement. Elle gère la commande des servomoteurs en fonction des valeurs renvoyées par les LDR.
- limitS() permet de limiter l'angle de commande envoyé aux servomoteurs (cf notes programmes pour les valeurs maximales tolérées). Cela permet de fixer informatiquement des butées angulaires.
- w_servo() permet d'envoyer une commande au servomoteur indiqué, en limitant automatiquement l'angle de commande.
- testLDR() permet de visualiser sur des LED les LDR qui reçoivent le plus de lumière, en incluant aussi l'erreur.
- lectureHB() et lectureGD() permettent de tracer sur deux courbes les valeurs renvoyées par les deux paires de LDR.