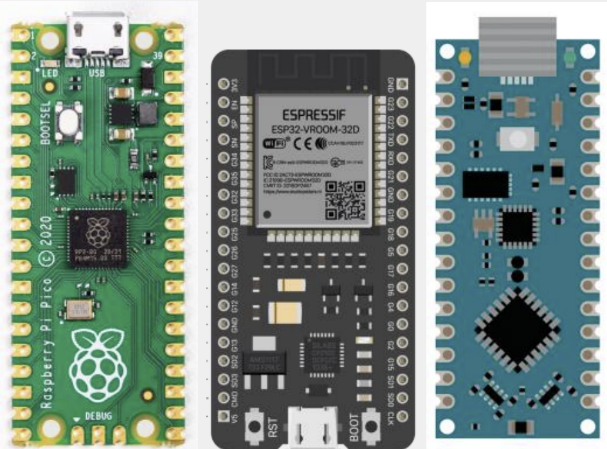
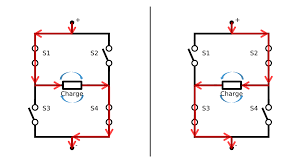
**Club Robotik : Formation Altium**

I - Composants principaux

Avant de faire des circuits électriques, il faut savoir quoi brancher dessus et donc connaître les composants existants. Ici on voit les plus importants actionneurs, capteurs, et contrôleurs du club :



Servo moteur Microcontrôleur



Moteur CC Ponts en H

Ce sont des exemples, il y en a d’autres (moteur pas à pas, optocoupleurs, level shifters, etc…). On en couvrira certains avec leurs utilités, conditions d’utilisations, points forts, points faibles, et autres informations utiles. Ce recueil a pour but d’être un guide d’introduction – n’hésitez donc pas à le compléter si vous utilisez d’autres composants, des pompes ou des électroaimants par exemple.

Servo moteur : Un petit moteur électrique permettant d’aller à une position angulaire précise. Permet de maintenir en place un composant avant de le relacher par exemple.

Comprant un asservissement précis à l’ordre du degrès près.

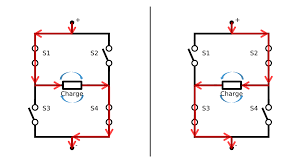
Se connecte avec 3 cables (Rouge - VCC, Noir/Marron - GND, PWM).

Le signal PWM permet d’envoyer la position voulue (Course entre 0° et 180°) au servo à l’aide d’un code C++. Ca position est également renvoyé par le PWM au microcontroleur. Cela implique de le connecter à une pin du microcontroleur permettant le IN et le OUT.

On peut également interposer un optocoupleur (ou photocoupleur) entre le microcontrôleur et le servo sur la connexion PWM. Il s’agit d’un composant permettant d’isoler le microcontrôleur des pics de tensions, des courants de retours, ainsi que des bruits électriques. En gros, c’est une protection pour le microcontrôleur.

P.S : l’optocoupleur n’est pas super utile pour un servo mais ça ne fait pas de mal

Moteur CC (ou DC) & Ponts en H : Vos cours de prépa sur les moteurs à courant continu. On alimente dans un sens ce qui fait tourner des aimants et crée un couple.

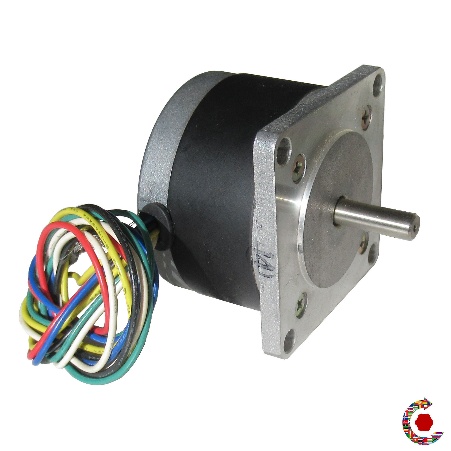
Les ponts en H sont de simples circuits électriques composés de 4 interrupteurs permettant de changer le sens du courant – ils permettent ainsi de contrôler le sens de rotation de ces moteurs.

Ils offrent une rotation continue avec une course >360° (contrairement au servomoteurs) dont le couple, et donc la vitesse, dépendent de la tension.

Il possède deux connexions – Le GND et le VCC i.e. la masse et l’alimentation.

Au club, on utilise le composant L293D qui est un double pont en H – il permet de contrôler deux moteurs à la fois - utile pour des robots à 2 roues ;).

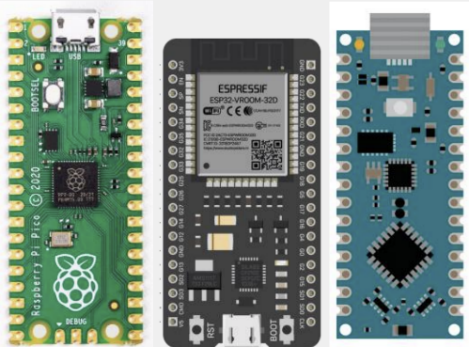
On peut également interposer un optocoupleur entre le microcontrôleur et la partie puissance (pont en H + moteur).

Moteur pas à pas (Stepper) : Moteur électrique qui divise une rotation complète en pas discrets. Chaque impulsion fait donc tourner le stepper d’un pas (ex : 1,8° par pas -> 200 pas/tour). Le moteur tourne par “sauts” successifs plutôt que de façon continue et à une course >360°.

Il est donc plus simple de contrôler la position d’un stepper que d’un moteur DC en revanche le couple est souvent limité à haute vitesse comparé au moteur DC.

Il possède 4 connexions – 2 pour chaque bobine que l’on relie à un driver. En effet, il nécessite un driver pour être contrôlé (DRV8825 au club par exemple). C’est un composant qu’on intègre sur la carte électronique entre le microcontrôleur et le stepper.

Il est également conseillé d’interposer un optocoupleur entre le microcontrôleur et le stepper.

Microcontrôleur (Pôle info pour plus de détails) : On utilise principalement les 3 de la photo au club à savoir – dans l’ordre – les Raspberry Pi Pico, les Esp32, et les Teensy.

Ces composants sont des mini-ordinateurs autonomes qu’on peut programmer afin de contrôler des actionneurs ou lire des données capteurs à l’aide de leurs nombreux pins.

! L’important à savoir est que tous les pins ne sont pas égaux, certains ont des fonctions particulières (GND, VCC, etc…), d’autres permettent uniquement d’envoyer ou de renvoyer des données, d’autres les deux, etc. Il faut toujours se référer à la « pin layout » de chaque microcontrôleur avant de créer une connexion !

Les Teensy :

Un microcontrôleur très puissant et très rapide surtout. Utile pour les projets en temps réel. On utilise les 4.0 au club, fonctionne en 3,3 V et se programme en C++ (ou Arduino)

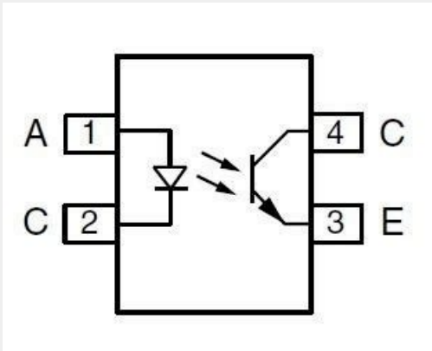
Les Esp32 :

Un microcontrôleur un peu moins puissant mais possédant le Wifi / Bluetooth et permettant de faire du multitâche. On utilise les Esp32 DevKit v1, fonctionne en 3,3 V et se programme en C++ (ou MicroPython ou Arduino).

Le DevKit correspond à la carte de développement autour (avec le port usb, des régulateurs de tensions, boutons EN & BOOT, des leds, etc) – l’esp32 correspond à la puce (WROOM32, H2, S3 …) qu’il est impossible de manipuler seule.

Les Raspberry Pi Pico :

Un microcontrôleur encore un peu moins puissant que les esp32 mais avec un prix très faible. Fonctionne en 3,3 V et se programme en C++ (ou MicroPython)



Les optocoupleurs (ou photocoupleurs) : Vu que j’en ai parlé plus haut, les voilà.

Composant électrique permettant d’isoler deux circuits puis éviter que le bruit ne se propage, qu’il y ait des courants de retour, ou encore des pics de tension.

Très utile avec les moteurs comme précisé plus haut

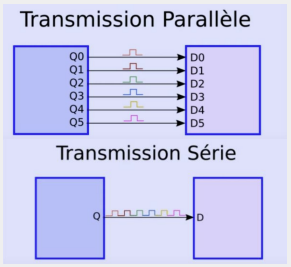
Concrètement, une Led infrarouge s’allume lorsque le courant passe. Cette onde est captée par le phototransistor ce qui induit un courant dans le circuit de sortie sans aucune connexion entre les deux.

En conclusion, ils existent plein de composants – électriques comme mécaniques – différents, chacun avec sa propre utilité. Le mieux est de se renseigner et d’utiliser les datasheets des composants pour choisir le bon et ensuite le brancher correctement !

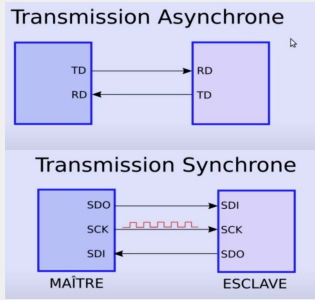
II - Protocoles de communications

Après savoir quels composants existent, il est utile de savoir comment ils communiquent entre eux : C’est le principe de ces protocoles de communication.

On verra ici les principaux à savoir : l’UART – l’I²C – le SPI – l’USB que vous connaissez déjà.

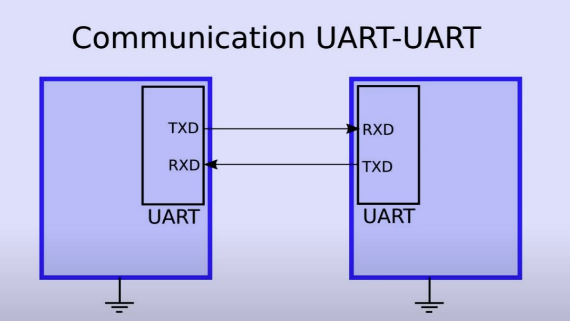
Tour d’abord, du vocabulaire : série / parallèle et synchrone / asynchrone.

Dans une communication série, chaque bit est envoyé l’un après l’autre ce qui rend la communication plus lente mais avec moins de fils. Contrairement au parallèle qui envoie plusieurs bits en même temps sur des fils différents

Dans une communication synchrone, un fil supplémentaire correspond à l’horloge permettant de synchroniser les deux composants. Celle du maitre est utilisé.

Dans une communication asynchrone, chaque signal contient des bits de synchronisation avant chaque paquet afin que les composants soient malgré tout synchronisés.

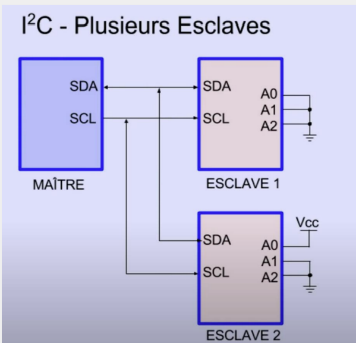
Maintenant, les protocoles :

L’UART (Universal Asynchronous Receiver/Transmitter) : Protocole de communication série asynchrone utilisé pour transmettre des données entre deux appareils via seulement deux fils : un pour émettre (TXD) et un pour recevoir (RXD).

Très simple mais ne permet que deux appareils contrairement à l’I²C ou SPI.

Est très lent (max 100 kbits/s) mais utilisé pour programmer des contrôleurs par exemple.

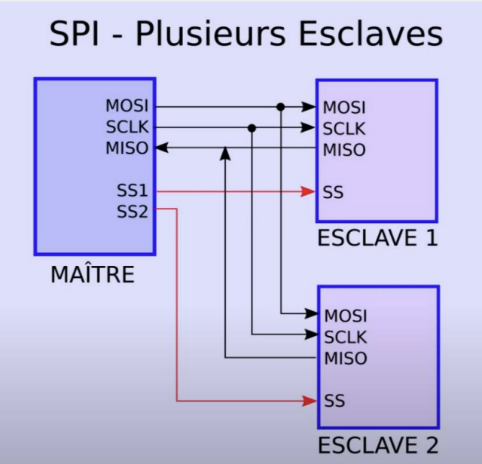
! On connecte bien le TXD de l’un au RXD de l’autre et vice versa !

I²C (Inter-Integrated Circuit) : Il s’agit d’un bus de communication série synchrone permettant à plusieurs composants de communiquer via 2 fils.

Un maître communique avec plusieurs esclaves à l’aide d’une adresse sur 7 bits modifiable.

Plus rapide (400 Kbits/s) que l’UART mais plus complexe et plus lent que SPI. Fonctionne sur des faibles distances.

! Important : Mettre des résistances de pull up entre 4 et 10 kΩ sur les lignes sda (serial data ligne – les données transmises en série) et scl (serial clock line – l’horloge) !

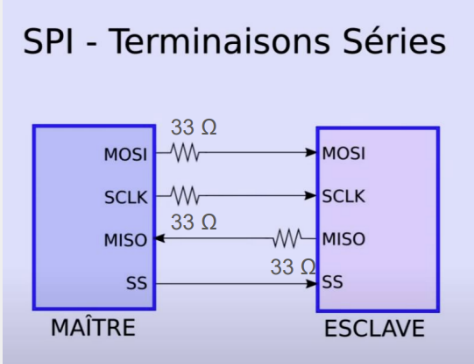


SPI (Serial Peripheral Interface) : protocole de communication série synchrone conçu pour des échanges entre un maître et plusieurs esclaves.

Fonctionne sur 4 fils: MOSI (Master Out Slave In) – MISO (Master In Slave out) – SCK (Serial Clock Select) – SS (Slave Select). SS nécessite une résistance de pull up.

Beaucoup plus rapide (centaines de MHz) mais nécessite plus de pins.

* À chaque front d’horloge, 1 bit est échangé entre maître et esclave (plein-duplex).
* Les données sont échangées bit par bit en parallèle sur MOSI et MISO.
* Le SS (chip select) est utilisé pour activer un esclave à la fois (à LOW).

Si la distance est trop grande ou la vitesse est trop élevé (USB), il faut faire de l’adaptation d’impédance – 50 Ω en général.

Permet d’éviter les ondes réfléchies et donc les signaux stationnaires (cours de prépa).

PWM (Pulse Width Modulation) : Technique de modulation qui permet de simuler une tension analogique avec un signal numérique.

Il génère un signal carré (0 ou 1) à fréquence fixe, mais fait varier le rapport cyclique (durée de l'état haut). Très simple à générer sur un Esp32 et donc très utilisé pour des servomoteurs, des leds, des moteurs, etc…