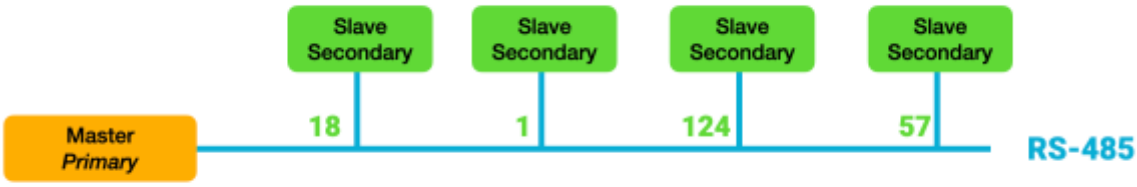
Modbus

[Mise à jour le 8/8/2022]

- **Source** : Mooc Fun “Programmer l'internet des objets”
- **Vidéos** sur YouTube
  - Les principes généraux de MODBUS

1. Historique

Modbus est apparu en 1979 et est toujours très populaire dans l’industrie.



2. Généralités

- **Vidéo** sur YouTube: Les principes généraux de MODBUS

À l’origine, Modbus était construit sur un bus série qui connectait différents équipements appelé secondaires ou slaves et un primaire ou master qui gère les communications. Chaque secondaire a un numéro unique ou adresse. Les adresses sont comprises entre 1 et 247. Le primaire n’a pas besoin d’une adresse puisque toutes les communications ont lieu avec lui. Le primaire envoie une requête au secondaire et le secondaire répond au primaire. Des communications directes entre deux secondaires ne sont pas possibles.

Un équipement Modbus gère l'information sous forme des registres :

- les **registres** qui peuvent prendre une valeur **binaire on** ou **off**. Si le master peut modifier l’état et, bien sûr, le lire, est appelé un ***coil***. Si la valeur binaire peut être uniquement lue, il s'agit d'un ***discrete input***.
- les registres sur **16 bits**. Ils sont utilisés pour représenter une valeur comme un courant électrique, une température, une vitesse de rotation… De même, si leur valeur ne peut être que lue par le primaire, il s'agit d'un ***input register*** sinon, si elle peut être également modifiée par le primaire, il est appelé ***holding register***.

Un équipement Modbus peut avoir jusqu'à 10000 registres de ces 4 catégories.

Modbus est un protocole **requête/réponse**. Le primaire envoie une requête à l'adresse d'un équipement pour lire ou écrire un de ses registres.

Une trame Modbus est une séquence de caractères commençant par un octet avec l'adresse du secondaire, suivi d'une commande (ou code de fonction) spécifique à chaque catégorie de registre :

- **1** pour lire un coil
- **2** pour lire un discrete input
- **3** pour lire un input register
- **4** pour lire un holding register
- **5** pour écrire un holding register et
- **6** pour écrire un input register

La suite de la trame contient les données, puis un CRC pour valider qu'il n'y a pas d'erreur de transmission dans la trame.

La partie donnée peut être différente dans la requête et la réponse.

Par exemple, pour lire un holding register, la requête contient l'adresse du premier registre à lire, et le nombre de registres à lire. La réponse contient le nombre d'octets transmis suivi de leur valeur.

Pour écrire sur un registre, les données de la trame seront l'adresse du registre et les données à écrire. La réponse est le même message.

Exemple en vidéo

- Sur Youtube