LYCÉE de l'ÉLORN
le choix de votre réussite

Le bus de liaison synchrone I²C

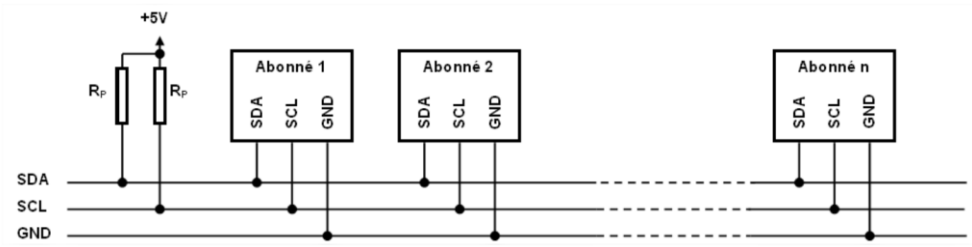
1. Historique

Le bus I2C (Inter Integrated Circuit Bus) est une liaison série synchrone développée par Philips pour les applications de domotique et d'électronique domestique. Il est présent sur les systèmes embarqués comme le Raspberry ou l'Arduino.

2. Caractéristiques

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils :

- un signal de donnée (SDA),
- un signal d'horloge (SCL) destiné à valider la présence des bits de données,
- un signal de référence électrique (Masse).

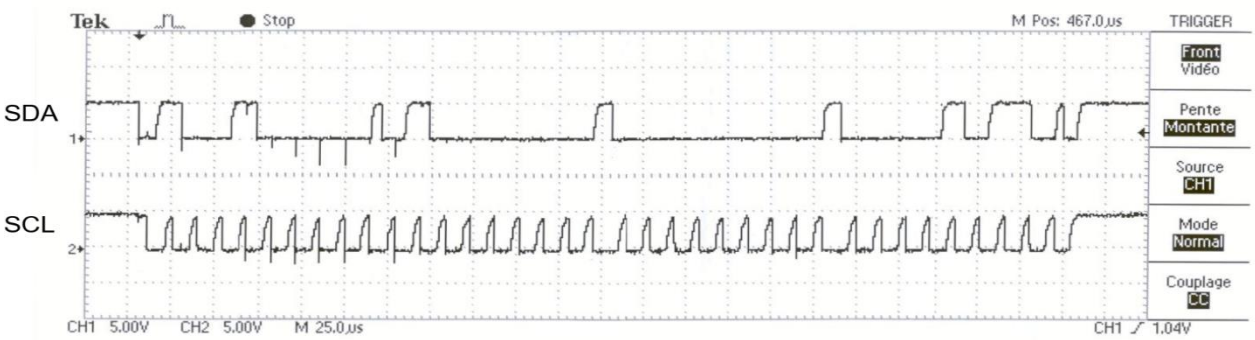


Les données sont transmises en série à une vitesse de 100Kbits/s en mode standard et jusqu'à 400Kbits/s en mode rapide.

3. Principe

Tous les abonnées peuvent définir un niveau électrique sur le bus sans qu'il y ait un risque de destruction de composant. Dès que le bus est libre, le premier abonné qui prend la parole devient le « maître » d'un échange de données.

Exemple de chronogramme :

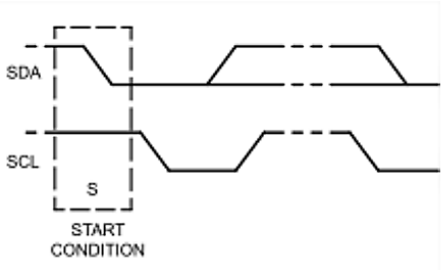


4. Le protocole de communication I²C

La communication sur le bus I²C ne peut se faire qu'entre 2 abonnés. Lorsqu'un abonné prend le contrôle du bus, il devient le maître de la communication. Il génère le signal d'horloge SCL et communique avec un esclave. Selon le sens de la communication, il sera l'émetteur ou le récepteur.

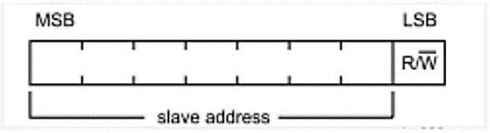
a) La condition de départ

Un abonné prend le contrôle du bus I²C en émettant une condition de départ : Niveau haut sur SCL et front descendant sur SDA
Cet abonné devient le maître.



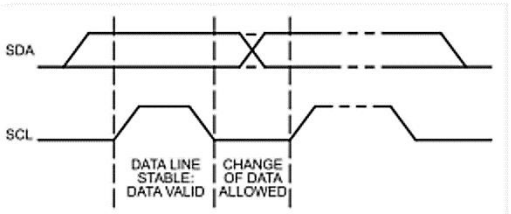
b) Transmission de l'adresse

Après avoir pris le contrôle, le maître transmet un octet contenant l'adresse de l'esclave (sur 7 bits) ainsi que l'opération effectuée (écriture ou lecture). « 1 » pour lecture, « 0 » pour écriture.

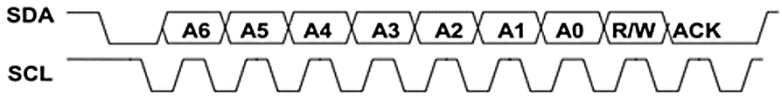


Philips Semiconductors

La validation d'un bit se fait lorsque le signal SCL est au niveau haut.

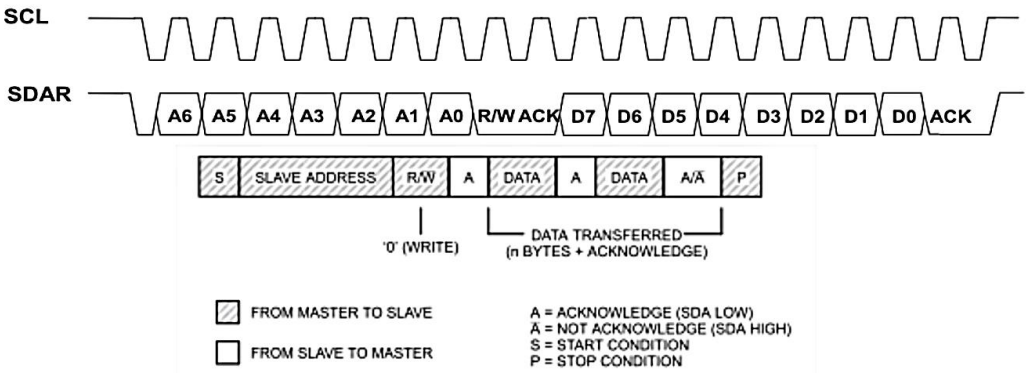


Lorsque l'esclave a détecté son adresse, il émet un bit d'acquittement (ACK) au niveau logique bas.

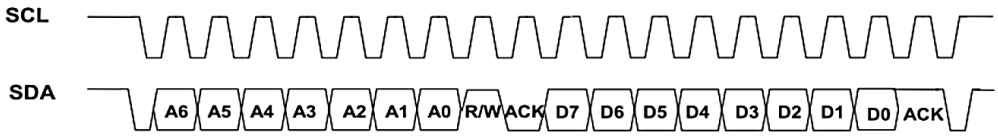


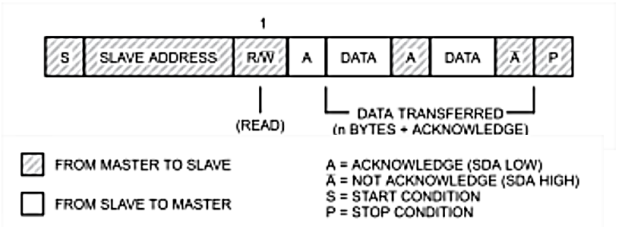
c) Transmission des données

Deux situation sont envisageables :
1er cas : Le maître envoie des données à l'esclave. A la fin de la transmission de chaque octet, l'esclave émet un acquittement.



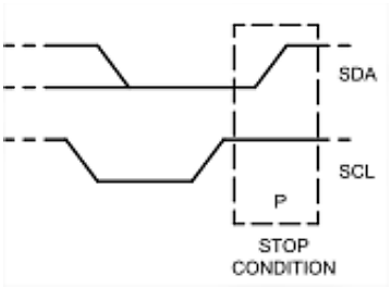
2nd cas : L'esclave envoie des données au maître. A la fin de la transmission d'un octet, le maître émet un acquittement (niveau « 0 ») s'il veut recevoir encore un octet ou bien un non acquittement (niveau « 1 ») s'il a terminé de recevoir.





d) Fin de la communication

Pour terminer la communication, le maître émet une condition d'arrêt : Un niveau haut sur SCL et un front montant sur SDA



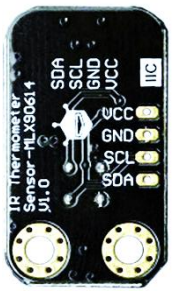
5. Application : Câblage d'un capteur de température sur une carte Arduino UNO

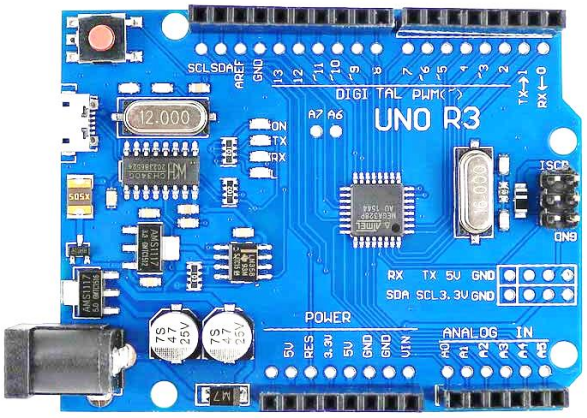
Nous allons brancher sur une carte Arduino un capteur de température infrarouge avec une communication par bus I^2C.

Caractéristiques du capteur *de température* :

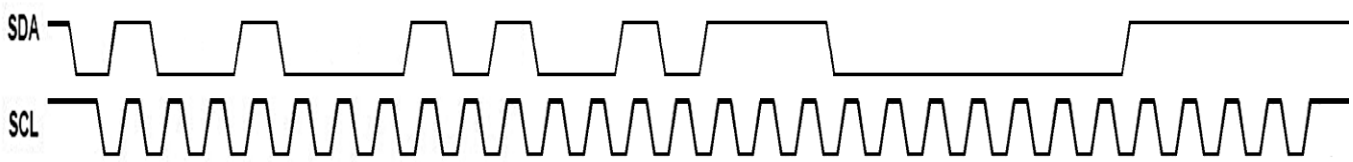
- Le capteur fonctionne sur 12 bits,
- Plage de mesure : -40°C à 300°C (résolution de 0,01 °C)
- Le capteur envoi d'abord les bits de poids faible (B7 à B0) puis ceux de poids plus forts (B11 à B8) de la température exprimée en centième de degré Celsius.

a) Réaliser le câblage du composant sur l'arduino





On relève la trame I^2C suivante :



b) Décodage de la trame :

- Entourez sur la trame le bit de START.
- Relevez l'adresse du capteur. La mettre en hexadécimal.
- Entourez sur la trame le bit de R/W. Quel est son état logique et que cela signifie-t-il ?
- Entourez sur la trame les bits d'acquittement (ACK).
- Entourez sur la trame les bits de données transmis par le capteur.
- Entourez sur la trame le bit de non-acquittement (NACK)
- Entourez sur la trame le bit de STOP.

c) Analyse des données :

- Donnez la valeur des 12 bits de mesure que le capteur a envoyé (lue sur la trame).
- En déduire la température mesurée par le capteur.

d) Passage en mode veille (Sleep mode) :

Un mode basse consommation peut être imposé au capteur en lui envoyant la commande $FFE8 (0xFFE8).

Générer le chronogramme de la ligne SDA pour que cette commande soit transmise au composant ayant l'adresse $B4 (0xB4)