Laboratoire 6 : I2C (circuit inter-intégré)

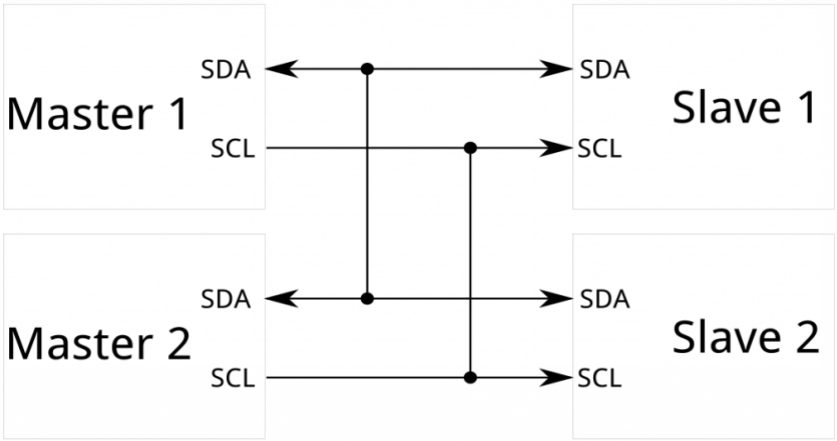
Ce laboratoire couvre le concept d'I2C. Pour plus d'informations sur ce sujet, consultez la fiche technique ATmega328P [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf] et Inter-Integrated Circuit .

1. I2C (circuit inter-intégré)

Le protocole I2C (ou IIC - Inter-Integrated Circuit / TWI - Two-Wire Interface) est un protocole de communication série synchrone multi-maître - multi-esclave développé par Phillips en 1982. Un bus I2C utilise les signaux suivants :

- SDA - ligne de données
- SCL - le signal d'horloge

Le signal d'horloge est généré par le maître, mais la ligne de données est contrôlée soit par le maître, soit par l'esclave, selon la phase du protocole, avec la restriction qu'un seul appareil peut transmettre à la fois. Pour cette raison, le protocole I2C est semi-duplex.



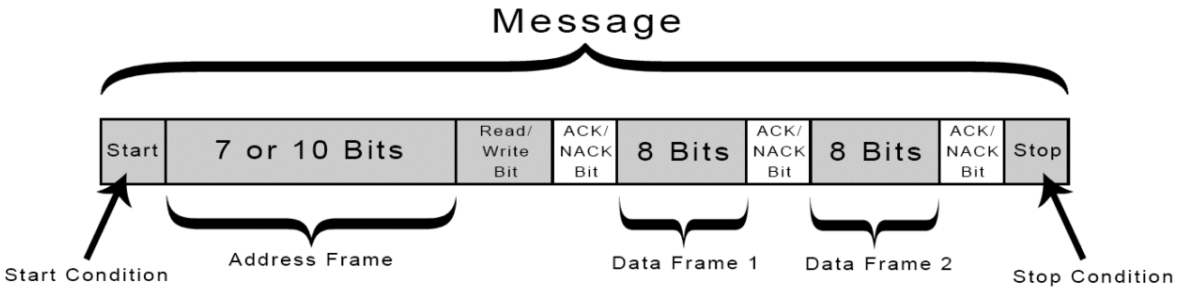
I2C est également connu sous le nom de TWI (interface à deux fils)

2. Mode de fonctionnement

Par rapport à SPI où le maître active, via le signal *Slave Select* , l'appareil avec lequel il souhaite communiquer, I2C ne nécessite pas un tel signal supplémentaire. Le protocole I2C introduit la notion d' *Adresse Esclave* . L'adresse d'un périphérique esclave est un nombre de 7 bits (le plus courant), de 8 bits ou de 10 bits. La communication entre un maître et un esclave se fait par messages et est toujours initiée par le maître. Ces messages peuvent être divisés en deux types de trames :

- une trame d'adresse
- une ou plusieurs trames de données

Ces trames ne sont échangées qu'après que le maître a envoyé *la condition de démarrage* . La fin d'un message est identifiée par *la condition d'arrêt* .



1. Condition de démarrage

Avant que le maître n'envoie sur la ligne de données l'adresse de l'esclave avec lequel il veut communiquer, il doit générer une condition de démarrage. La condition de démarrage amène tous les dispositifs esclaves à "écouter" la ligne de données car une adresse suivra. Pour générer cette condition, le maître laisse la ligne SCL HIGH et met la ligne SDA LOW.

2. Cadre d'adresse

Après que le maître a généré la condition de démarrage, il envoie sur la ligne de données (SDA) l'adresse de l'équipement esclave avec lequel il souhaite communiquer. L'adresse est (la plupart du temps) un nombre de 7 bits (bits A6-A0). Le bit 0 indique si le maître lance une opération de lecture (le bit 0 est 1) ou une opération d'écriture (le bit 0 est 0).

L'esclave reconnaissant son adresse envoie un ACK au maître en mettant la ligne SDA LOW au neuvième cycle d'horloge. L'état par défaut des lignes SDA/SCL est HIGH en raison des résistances pull-up. Le Maître/Esclave ne fait que "tirer" les lignes vers BAS.

Le maître identifie s'il a reçu ACK (SDA défini LOW) ou NACK (SDA est resté HIGH pendant le neuvième cycle d'horloge).

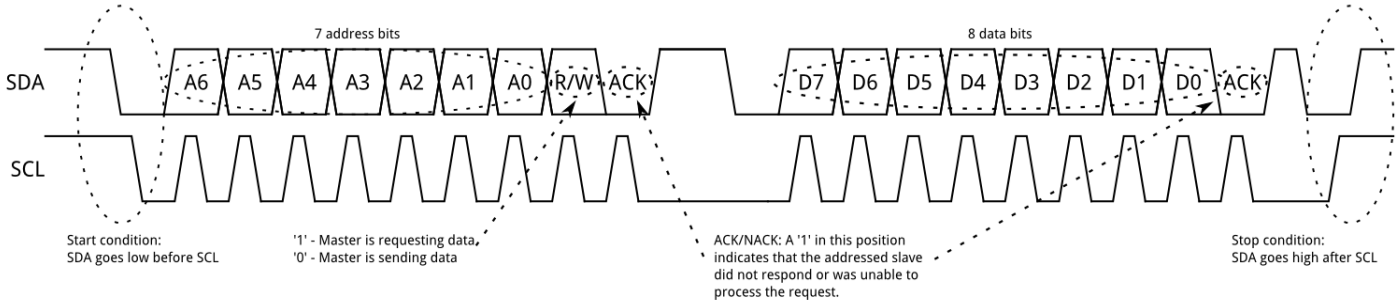
3. Trames de données

Si le maître a reçu l'ACK (s'il y a un esclave sur le bus avec cette adresse), il peut continuer la transmission de données (opération d'écriture) ou la réception de données (opération de lecture). Le nombre de trames de données est arbitraire, n'importe quel nombre peut être échangé. Chaque trame envoyée/reçue est ACK'd ou NACK'd. Selon l'opération (lecture ou écriture), l'ACK/NACK est envoyé soit par le maître, soit par l'esclave.

- Si le maître a lancé une opération d'écriture, chaque trame envoyée est acquittée (ACK'd) par l'esclave.
- Si le maître a initié une opération de lecture, chaque trame reçue est acquittée (ACK'd) par le maître. Lorsque le maître veut arrêter la transaction après qu'un certain nombre de trames ont été reçues, au lieu d'envoyer ACK, il envoie NACK. Ainsi, l'esclave cessera de transmettre.

4. Condition d'arrêt

Une fois que toutes les trames de données ont été échangées, le maître génère la condition d'arrêt. Cela se fait en relâchant la ligne SDA (passant de LOW à HIGH) **après** avoir relâché la ligne SCL (passant de LOW à HIGH).



Le bus est le même pour tous les capteurs et chaque esclave est référencé par une adresse. Ainsi, si nous avons plusieurs esclaves connectés au bus, le maître choisit avec lequel d'entre eux il veut initier la communication, en précisant l'adresse indiquée dans la fiche technique du capteur. Le plus souvent, des adresses à 7 bits sont utilisées et le bit le moins significatif de l'octet d'adresse indique si une opération d'écriture ou de lecture suit. Après avoir identifié l'appareil avec lequel il communique, la transmission des données peut commencer.

3. Enregistrez la configuration I2C

Arduino peut fonctionner à la fois en mode maître I2C et esclave I2C.

Registre de débit binaire TWI

Bit	7	6	5	4	3	2	1	0	
(0xB8)	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- TWBR sélectionne le diviseur de fréquence qui génère la fréquence d'horloge SCL dans les modes maîtres.

Registre de contrôle TWI

Bit	7	6	5	4	3	2	1	0	
(0xBC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Le TWCR est utilisé pour activer l'interface I2C, initier l'accès au maître en appliquant la condition START, générer l'accusé de réception du récepteur, générer la condition d'arrêt et la contrôler lors de l'écriture dans le registre de données TWDR. Il indique également les collisions d'écriture.
- TWINT : Indicateur d'interruption TWI
  - 1 - Clair, défini par le logiciel, le reste est défini par le matériel.
- TWEA : TWI Activer le bit d'accusé de réception
  - 1 - ACK est envoyé
  - 0 - NACK, déconnexion virtuelle de l'appareil de l'interface
- TWEN : bit d'activation TWI
  - 1 - Permet de communiquer
  - 0 - Arrêt de la transmission quel que soit le statut

Registre d'état TWI

Bit	7	6	5	4	3	2	1	0	
(0xB9)	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- TWSR décrit l'état et le prescaler de l'interface.

Registre de données TWI

Bit	7	6	5	4	3	2	1	0	
(0xBB)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- TWDR contient en émission le prochain octet à émettre et en réception le dernier octet reçu.

4. Utilisation du module I2C sur l'AVR

Le processus de configuration du module I2C sur les microcontrôleurs AVR est relativement simple (seule la fréquence d'horloge doit être réglée) :

Configuration I2C :

```
void setup ( )
{
  // Initialise I2C à une horloge de 100 KHz
  // Registre d'état TWI : initialise le prescaler à 1
  TWSR =  ( 0b00  << TWPS0 ) ;
  // Registre de débit binaire TWI : définir le débit binaire
  // SCL_Freq = CPU_Freq / (16 + 2*TWBR * TWSR_Prescaler)
  // donc : TWBR = (SCL_Freq / CPU_Freq - 16) / (TWSR_Prescaler * 2)
  TWBR =  72 ;  // (16000000/100000 - 16) / (1 * 2)
}
```

Mais les étapes pour effectuer une transaction complète sur I2C sont complexes :

- configuration TWCRpour passer la condition de démarrage ;
- attendre la fin de l'opération (par occupé en attente sur TWSR) ;
- écriture de l'adresse du dispositif esclave dans le registre TWDR+ attente ;
- vérification de la condition d'acquittement (également via TWSR);
- transmettre ou recevoir zéro ou plusieurs octets via le registre de données TWDR(les étapes intermédiaires d'attente de la fin des opérations doivent également être incluses);
- enfin, en passant la condition d'arrêt à travers un autre drapeau de TWCR.

5. Utilisation de TWI via la bibliothèque Arduino

Bien sûr, Arduino dispose de fonctions de haut niveau pour les opérations sur le bus I2C du microcontrôleur, par exemple :

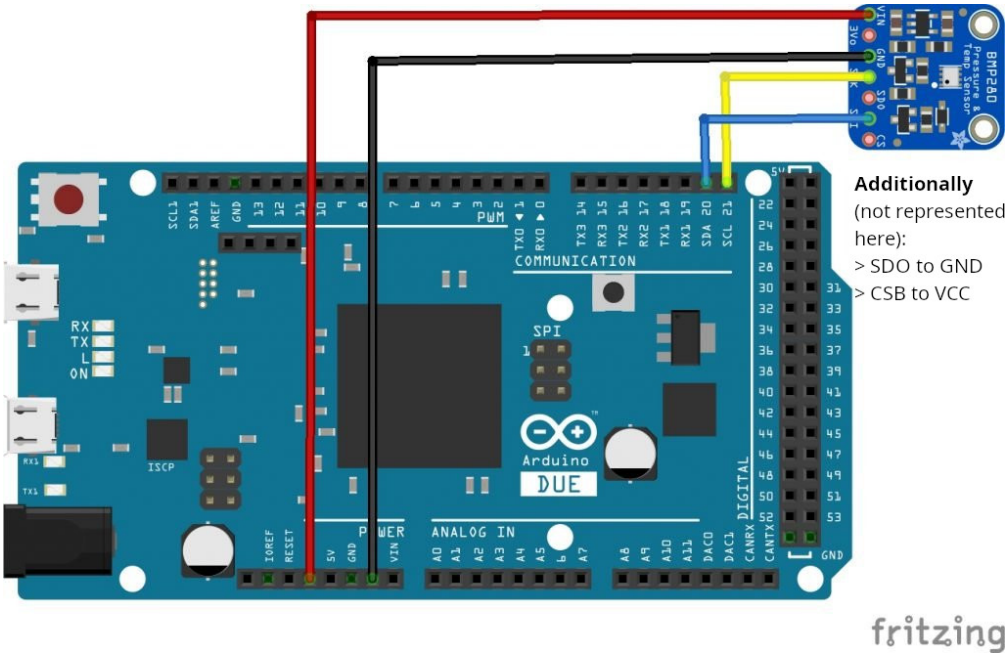
```
void setup ( )
{
  Fil. setClock ( 100000 ) ;  //
  Fil Hz. commencer ( ) ;
}

boucle vide ( )
{
  Fil. débutTransmission ( 0x52 ) ;   // adresse esclave
  Wire. écrire ( 0x88 ) ;
  Fil. finTransmission ( ) ;
  retard ( 5OOO ) ;
}
```

On ne fait pas ça ici !

6. Exercices

Pour ce laboratoire, nous utiliserons un Arduino Uno avec le rôle de maître, auquel un capteur de température + pression BMP280 a été connecté (cliquez pour la fiche technique

[https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf]) à partir duquel nous lirons les



données.

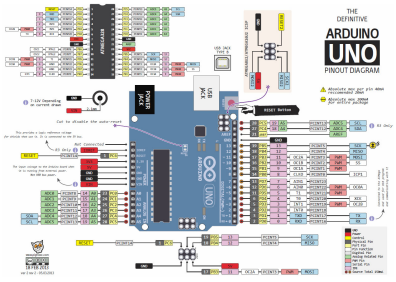
Pour cela, vous devrez télécharger le squelette de code trouvé ici : lab06-i2c.zip .

Tâche 1 : Finaliser et tester le code pour effectuer un balayage linéaire des appareils I2C connectés. Vous devriez trouver l'appareil sur `0x76`.

Tâche 2 : Définir les fonctions de lecture/écriture des registres de l'appareil et les utiliser pour lire + convertir la température en degrés Celsius.

7. Ressources

- Fiche technique Atmega 328p
- Squelette de code I2C Lab 6
- Brochage Arduino UNO



- Responsables : Florin Stancu [mailto:florin.stancu@upb.ro] , Daniel Rosner [mailto:daniel.rosner@upb.ro] , Cristi Trancă [mailto:dumitru.tranca@upb.ro]

RÉSOUDRE

8. Liens utiles

- Arduino I2C [https://www.arduino.cc/en/Reference/Wire]
- Fiche technique ATmega328p [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf]
- Fiche technique Capteur BMP280 [https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf]