<https://sti2d.ecolelamache.org/iii_communications_informatiques__3_exemple_n2_le_bus_i2c.html>

***Qu’est-ce que le bus I2C et comment fonctionne-t-il sur Arduino ?***

***La solution la plus simple pour établir une communication entre votre board***[***Arduino***](https://www.arduino-france.com/tutoriels/quest-ce-que-arduino/)***et les différents périphériques de la carte, ou des périphériques externes tels capteurs de température, d’humidité, accéléromètres ou gyroscopes par exemple, est l’utilisation du protocole I2C, compatible avec des milliers de composants existant et intégré à toutes les versions de carte Arduino.***

*Pour la mise en place aisée de vos applications domotiques ou robotique sur mesure, trouvez tout d’abord les****bases du fonctionnement du bus de communication I2C****et l’algorithme permettant la communication d’une multitude de périphériques sur une seule ligne.*

***Puis les aspects hardware et software à prendre en compte pour l’utilisation concrète d’I2C avec votre board Arduino sont abordés, dont le câblage des différents éléments de votre système et les principales fonctions à utiliser dans votre code.***

*Communication avec les périphériques Arduino*

*Vous désirez pouvoir établir une communication entre votre bord Arduino et les différents périphériques de la carte, voir des périphériques externes disponibles dans le commerce, tels des capteurs de température, d’humidité, des accéléromètres ou des gyroscopes, ou divers types d’actionneurs ?*

*Pour cela la solution la plus simple est****l’utilisation du bus I2C, directement intégré à votre carte*** *; il existe en effet des milliers de composants disposant d’une interface I2C, et toutes les cartes Arduino sont basées sur des microcontrôleurs disposant de cette fonction.*

*Le protocole I2C*

***I2C, pour Inter Integrated Circuit , est un protocole de communication crée à la base par Philips en 1982 pour standardiser l’échange de données entre différents circuits intégrés d’un même système. Elle est aujourd’hui assurée par le fabricant NXP, spin-off de l’entreprise.***

*Ce protocole est****basé sur un bus de communication sériel****, ce qui signifie qu’un seul câble est utilisé pour le transfert de données.*

*En pratique,****le bus I2C est constitué de deux câbles****, un donc pour les données, nommé SDA (Serial Data) et l’autre faisant office d’horloge pour déterminer la fréquence de la communication, nommé SCL (Serial Clock).*

*Dans la mesure où tous les périphériques sont connectés sur le même bus, comment donc faire pour qu’un seul des capteurs ne communique ses données en même temps et éviter une saturation de la ligne ?*

*Ceci est garanti par le protocole à proprement parler.*

*Tout d’abord, il n’y a qu’****un maître****(le microcontrôleur de votre board Arduino),****et une multitude d’esclaves****(vos périphériques), chacun identifiés par une adresse unique. Seul le maître peut initier une communication.*

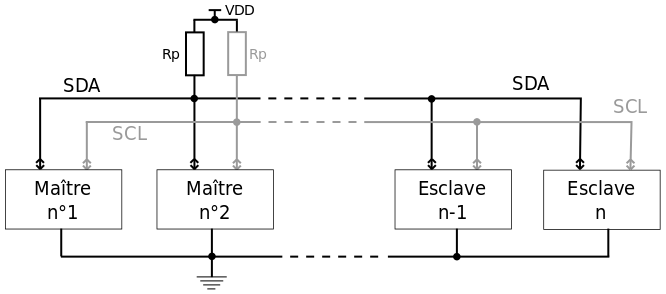
*Le maître envoie tout d’abord l’adresse du périphérique dont il désire recevoir les données. L’esclave envoie un premier signal de confirmation (acknowledgement) pour signifier qu’il a bien reçu la demande.*

*Puis le maître envoie l’adresse d’un registre interne du périphérique. Par exemple, pour un accéléromètre, on a trois registres stockant respectivement les données de l’accélération selon X, Y et Z. Un deuxième signal de confirmation est envoyé par le périphérique.*

*Enfin c’est le périphérique qui émet cette fois le message, en transférant la valeur du registre qui a été sollicité. Il termine avec une dernière confirmation, après quoi le maître envoie un signal spécifique pour mettre fin à la communication.*

*Le processus est similaire pour l’écriture sur un périphérique par le microcontrôleur, pour piloter un actionneur ou sauvegarder des données par exemple. Dans ce cas une commande spécifique est envoyée lors de l’adressage du registre et c’est le maître qui transmet les données.*

*À noter que les deux lignes SDA et SCL sont liées à une résistance de pull-up. L’état 0 est donc l’état dominant. D’un point de vue hardware, vous n’aurez pas à implémenter ces résistances qui sont en place par défaut.*

**

*Utilisation d’I2C avec Arduino*

*La première chose à faire est d’identifier l’adresse des périphériques liés à votre carte, et l’adresse de leur divers registres internes. Vous trouverez ces informations dans la fiche technique de votre bord. Ensuite, dans le cas d’une connexion de périphériques externes, indépendants ou montés sur d’autres cartes Arduino, il faut identifier les ports attribués aux lignes SDA et SCL de chaque carte ou composants et les relier entre eux.*

*N’oubliez bien entendu pas de connecter chaque composant à l’alimentation (5V / GND ou 3V3 / GND).*

*Il s’agira ensuite principalement de coder les échanges dans votre soft. Vous trouverez ci-dessous les principales commandes :*

***Wire. Begin();***

*Initialisation de la communication I2C, à effectuer au début de votre programme.*

***Wire.beginTransmission(Adresse de votre périphérique);***

*Cette commande est à appeller à chaque fois pour initialiser une transmission avec le périphérique dont l’adresse est donnée en argument.*

***Wire.write(Adresse du registre sollicité);***

*Cette commande permet d’accéder à un registre du périphérique, et est à appeler avec d’écrire ou de lire des données.*

***Wire.write(Byte à écrire);***

*Écriture des données désirées dans le registre sollicité au préalable.*

***Wire.requestFrom(Adresse de votre périphérique, n);***

*Lecture de n bits du périphérique adressé, dans le registre sollicité au préalable.*

***Wire.endTransmission();***

*Fin de la transmission. Cette commande doit être appelée à chaque fois, et avant l’envoi de toute nouvelle requête par I2C.*

<http://hlembke.de/arduinoablage/crate.php?20150411i2c>

<https://www.arduino-france.com/tutoriels/quest-ce-que-le-bus-i2c-et-comment-fonctionne-t-il-sur-arduino/>

*k.aboudou*