Objectifs :

Comprendre le bus I2C et mettre en place une communication entre deux matériels.

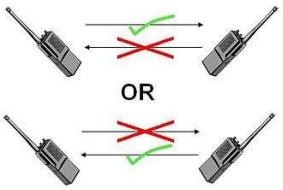
PRESENTATION DU BUS I2C (Inter-Integrated Circuit)

Conçu par Philips pour les applications de domotique et d'électronique domestique,
il permet de relier facilement un microcontrôleur et différents circuits, notamment ceux d'une télévision moderne : récepteur de la télécommande, réglages des amplificateurs basses fréquences, tuner, horloge, gestion de la prise péritel, etc.

Ce bus porte parfois le nom de TWI (Two Wire Interface) chez certains constructeurs.

Le bus I2C est un bus de série synchrone bidirectionnel half duplex.
La communication est de type maître esclave donc pas de perte de données.
Le débit de la communication est :

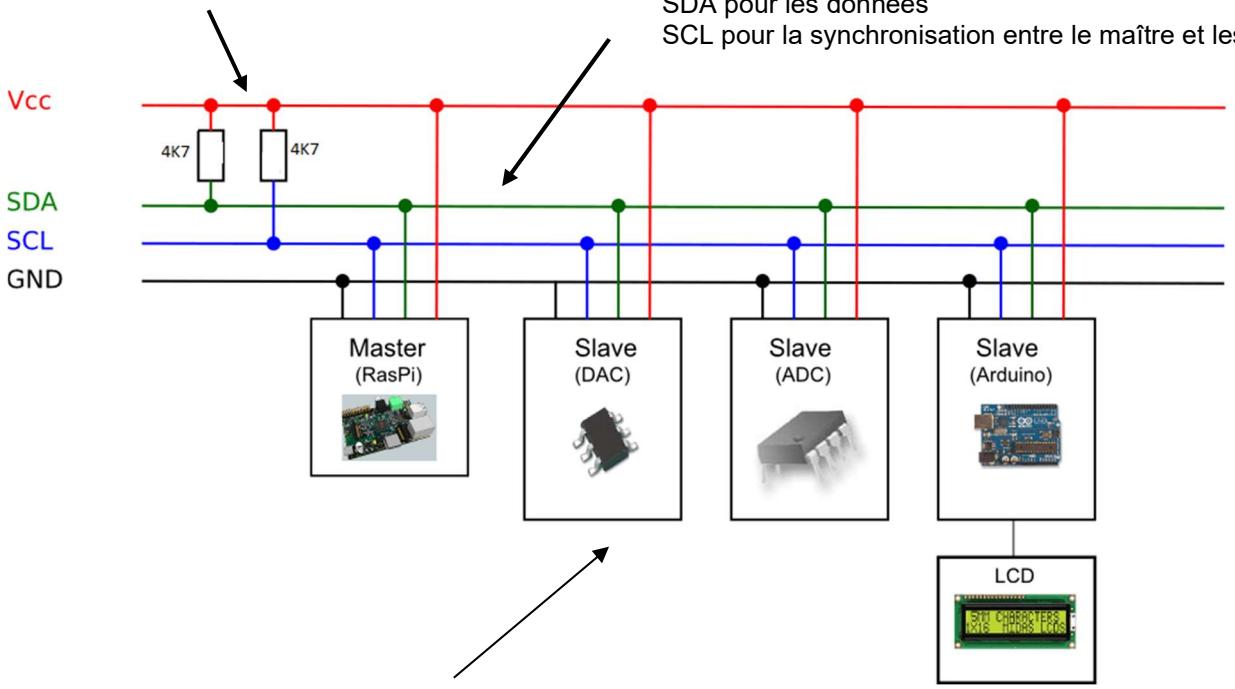
100K bits/s : mode standard
5M bits/s : mode ultra fast



1Présentation d'une architecture possible entre plusieurs types de circuits:

Ces deux résistances sont obligatoires pour assurer l'état repos du bus.

Pour communiquer sur le bus I2C, il faut au minimum trois fils :
La masse (GND)
SDA pour les données
SCL pour la synchronisation entre le maître et les esclaves



Dans le schéma ci-dessus, il y a un maître : RaspberryPI et trois esclaves (DAC, ADC carte Arduino)
Chaque esclave est identifié par une adresse unique donnée par le constructeur du circuit.

Dans une communication maître-esclave des principes sont :

Le maître gère la communication donc un esclave ne peut pas transmettre des données sans que le maître ne l'autorise à parler.

2 Présentation des lignes SDA et SCL :

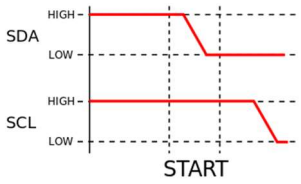
SDA (Serial Data Line) :
Ligne de données bidirectionnelle.
Cette ligne est générée par le maître pendant une phase d'écriture du maître ou par l'esclave pendant une phase de lecture du maître.

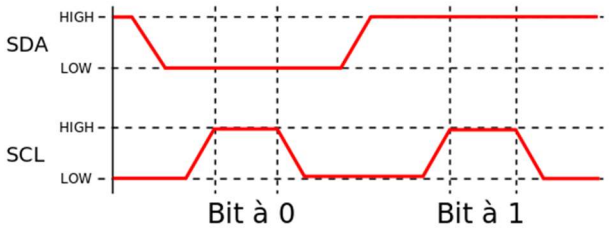
SCL (Serial Clock Line) :
Ligne d'horloge de synchronisation bidirectionnelle.
Dans une communication I2C, il y a toujours un seul maître avec un ou plusieurs esclaves.
Le Maître génère le signal d'horloge.

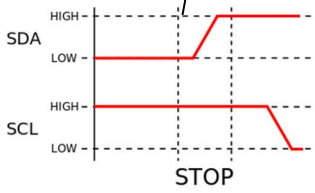
3 Présentation de la structure d'une trame I2C:

Au repos, c'est-à-dire sans transmission : SDA=SCL=1

Condition de départ START	Adresse du composant I2C sur 7bits MSB en premier	Lecture ou écriture. sur 1 bit. 1 : lecture (Read) 0 : écriture(Write)	Acquittement fait par l'esclave (1 bit). ACQ = 0 NACK=1	Octets de données. de 0 à x	Condition de d'arrêt STOP







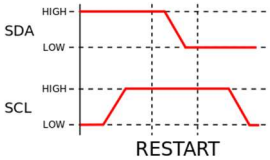
Remarques:

1 :
On peut câbler 2^7 esclaves car on a 7 bits pour les adresses.

2 :
Pour la condition de START, on part de l'état de repos (SDA=SCL=1)
et la ligne SDA doit passer à 0 donc un front de descente.

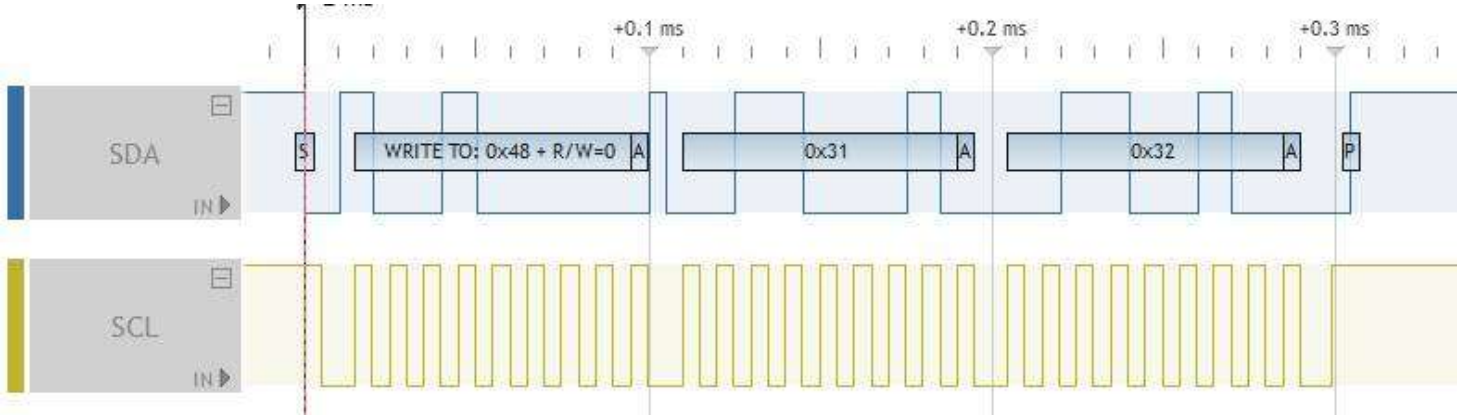
3 :
Pour la condition de STOP, il faut que SCL=1 et un front de montée sur la SDA.

4 :
Normalement, on a une condition de START, l'adresse et les données et pour finir la condition de STOP. Mais parfois, il y a certains circuits impose une nouvelle condition de START sans attendre la condition de STOP. Cette nouvelle condition de START s'appelle un RESTART :



5 :
Pour la lecture ou l'écriture d'un zéro :
SDA = 0 et une impulsion (passage à l'état haut puis à l'état bas) sur la ligneSCL.
6 :
Pour la lecture ou l'écriture d'un un :
SDA = 1 et une impulsion (passage à l'état haut puis à l'état bas) sur la ligneSCL.

Exemple de chronogramme avec présence du circuit esclave

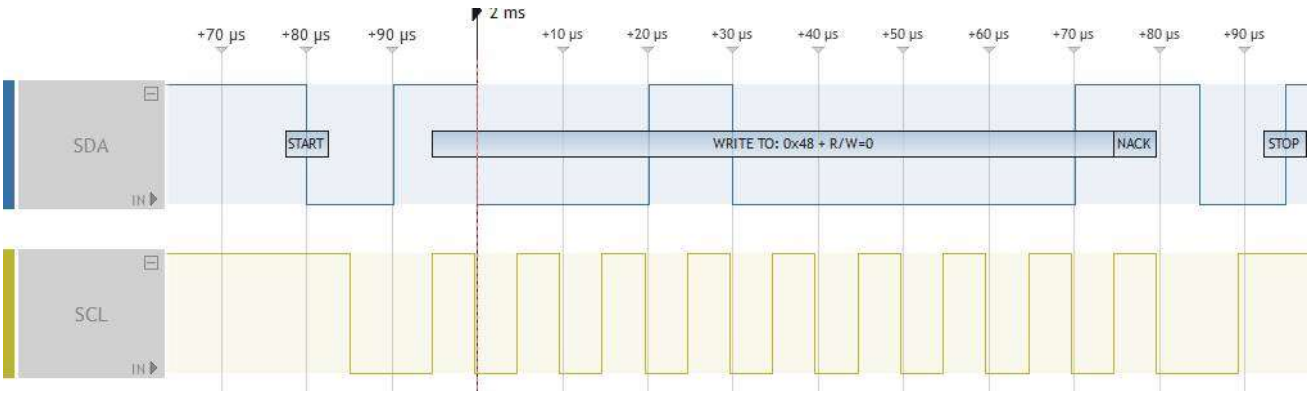


Analyse du chronogramme :

Condition de START car SCL=1 et un front de descente sur SDA
Dans ce chronogramme, le maître écrit car R/W = 0 trois octets :
Adresse codée sur 7 bits : 0010100
R/W sur 1 bit : 0
1 bit pour l'acquittement de l'adresse : 0 (ACQ)

1 octet de données 0x31 acquitté par l'esclave : ACQ
1 octet de données 0x32 acquitté par l'esclave : ACQ
Condition de STOP car SCL=1 et un front de montée sur SDA

Exemple de chronogramme avec absence du circuit esclave ou erreur de câblage :



On remarque la présence d'un NACQ donc l'adresse aucun esclave n'a acquitté cette adresse donc fin de la transmission.

TRAVAIL A FAIRE :

Tester les deux exemples ci-dessous. Conclusion.

EXEMPLE N°1: Transmission d'un maître vers un esclave des données 1 et 2

Avec arduino1.6

Maitre.ino

```
#include <Wire.h>

void setup()
{
  Wire.begin();                                  // mise en place liaison I2C
}

byte x = 0;

void loop()
{
  Wire.beginTransmission(72);                    // I2C start pour le composant adresse 72
  Wire.write("12");                              // On ecrit 2 caracteres
  Wire.endTransmission();                        // I2C stop
  delay(500);                                    // On attend 0,5s
}
```

Le maître ne possède pas d'adresse.

esclave.ino

```
#include <Wire.h>
byte c;                          // variable pour la lecture d'un caractere
void setup()
{
  Wire.begin(72);                // adresse I2C de l'esclave du le bus
  Wire.onReceive(recept);        // Mise en place d'une fonction de reception
  Serial.begin(9600);            // Mise en place de la liaison serie RS232( TERMINAL)
}

void loop()
{
  delay(100);
}

void recept(int nb)
{
  while(Wire.available()>0)      // Il y a des caracteres à lire
  {
    c = Wire.read();             // lecture d'un caractere
    Serial.write(c);             // affichage sur le terminal
  }
}
```

EXEMPLE N°2: Le maître demande des informations à un esclave

Avec arduino1.6

Maitre.ino

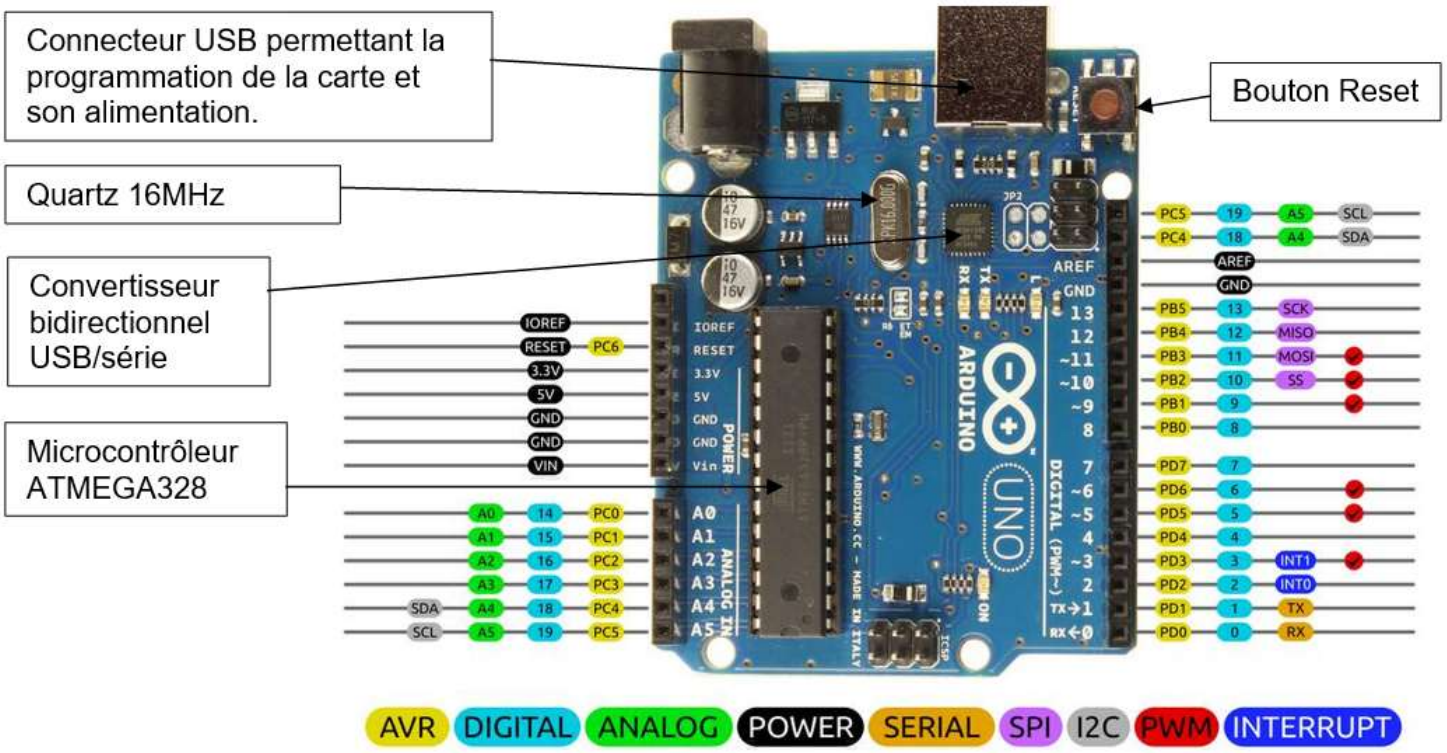
```
#include <Wire.h>                          //bibli i2c
byte val =0;                               //declar variable
byte led=13;                               //led sur sortie 13
byte tab[4];                               // tableau pour stockage des donnees reçues
byte i=0;                                  // indice pour le tableau
void setup()
{
   pinMode(led, OUTPUT);                   //led en sortie
   Wire.begin();                           //initialisation  i2c
   Serial.begin(9600);                     //init sortie serie asynchrone
}

void loop()
{

  Wire.requestFrom(0x48,3);                //demande de lecture de 3 octets  01001000=72
  i=0;
  while(Wire.available()>0){               // lire tant qu'il y a des caracteres
   tab[i] = Wire.read();                   // lecture des caracteres et stockage dans un tableau
   Serial.println(tab[i],DEC);             // print the character
   i++;
}

  digitalWrite(led, HIGH);                 // allume la Led sur la sortie 13
  delay(500);                              // attendre 0,5s
  digitalWrite(led, LOW);                  // eteint la Led
  delay(500);                              // Attendre 0,5s
}
```

esclave.ino

```
#include <Wire.h>                                    //bibli i2c

int led=13;                                          //led sur sortie 13
byte tab[3]={0b10000000,0b00001100,0x25};    // Tableau des données a écrire

void setup()
{
   pinMode(led, OUTPUT);                             //led en sortie
   Wire.begin(0b1001000);                           //adresse  i2c    1001000=72
   Wire.onRequest(demande);

}

void loop()
{

  digitalWrite(led, HIGH);                          // Led allumée
  delay(100);                                       // attendre 100ms
  digitalWrite(led, LOW);                          // Led éteinte
  delay(100);                                      // attendre 100ms
}

void demande()
{

 Wire.write(tab,3);
}
```

ETUDE DE LA BIBLIOTHEQUE I2C pour Arduino

1 Aspect matériel:



La communication I2C utilise les broches :

PC4 pour SDA
PC5 pour SCL

2 Aspect Logiciel :

1 Il faut faire un #include<Wire.h>
2 Les différentes fonctions possibles avec Arduino1.6

Les plus utiles sont :

Wire.beginTransmission() ; // en maître seulement
Wire.endtransmission() ;

Pour écrire sur la bus I2C :

Wire.write(string) ; // voir la doc pour les autres options possibles

Pour lire sur le bus I2C :
Wire.read() ; associé avec Wire.available() ;

Voir documentation pour plus de détails.

Functions

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- onReceive()
- onRequest()