Création d'une librairie pour un capteur I2C BH1750

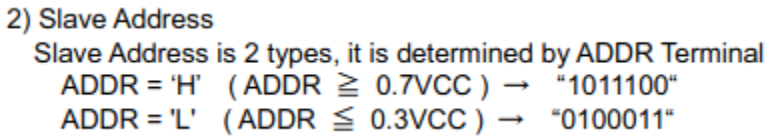
On commence par lancer I2C scanner, afin de vérifier le câblage de notre capteur.

```
Scanning...
I2C device found at address 0x23  !
I2C device found at address 0x48  !
done
```

On a donc deux capteurs, le capteur de luminosité, et le capteur de temperature.

1.1 Étude de la documentation constructeur

On va chercher dans la documentation afin de retrouver l'adresse I2C du capteur de luminosité.



2) Slave Address
Slave Address is 2 types, it is determined by ADDR Terminal
ADDR = 'H' (ADDR ≧ 0.7VCC) → "1011100"
ADDR = 'L' (ADDR ≦ 0.3VCC) → "0100011"

Cela nous donne :

ADDR = 'H' —> 0x5C

ADDR = 'L' —>0x23

On retrouve bien notre capteur sur l'adresse 0x23 !

Le capteur communique en simplex, on commence par un le start de la librairie wire (begin.Transmission()), suivi du mode utilise, soit une transmission continu (ADDR = 'L' —> 0x23), soit une transmission unique (ADDR = 'H' —->0x5C). On aura ensuite un bit pour savoir si la donnée envoyée est à but d'instruction ou de lecture.

On attendra ensuite acknowledge de l'esclave ainsi que son renvoie d'information quand la demande est une lecture de données. Dans le cas contraire, cela sera la ligne de code d'instruction (voir en dessous) qui lui sera envoyé.

On attendra ensuite l'acknowledge de l'autre parti pour finir par la fin de la librairie wire (end.Transmission()) dans le cas d'une instruction. Dans l'autre cas, On recevra le low byte restant pour finir sur un acknowledge du maître et la fin de la librairie wire.

① Send "Continuously H-resolution mode " instruction

ST	0100011	0	Ack	00010000	Ack	SP

② Wait to complete 1st H-resolution mode measurement.(max. 180ms.)

③ Read measurement result.

ST	0100011	1	Ack	High Byte [15:8]	Ack

Low Byte [7:0]	$\overline{\text{Ack}}$	SP

- **Démarrage du circuit :**

  Dans la doc technique, on trouve un tableau avec toute les fonctions

Instruction	Opecode	Comments
Power Down	0000_0000	No active state.
Power On	0000_0001	Waiting for measurement command.

- **Reset du circuit :**

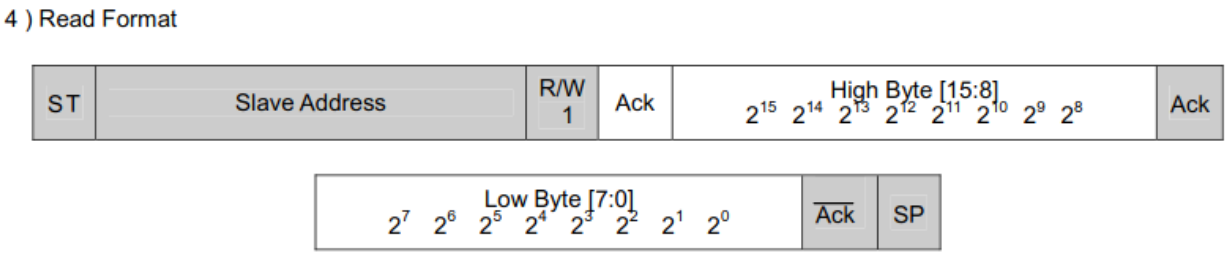
Instruction	Opecode	Comments
Reset	0000_0111	Reset Data register value. Reset command is not acceptable in Power Down mode.

- **Circuit en mode continu haute résolution**

Continuously H-Resolution Mode	0001_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms.

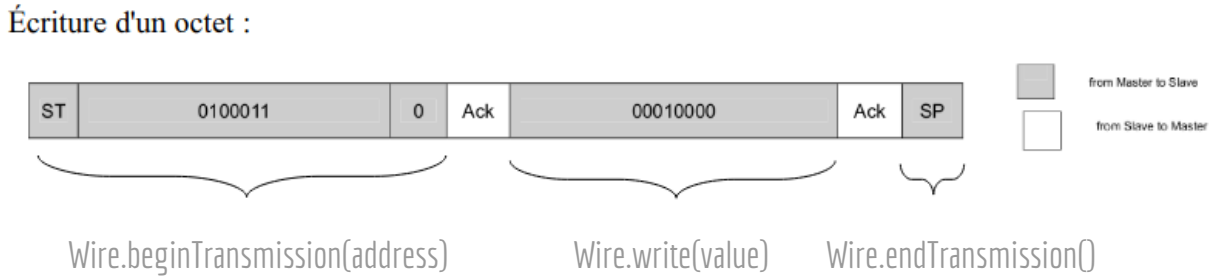
- **Lecture d’une mesure**

  Il suffit de mettre un 1, sur le bit de Read/Write :

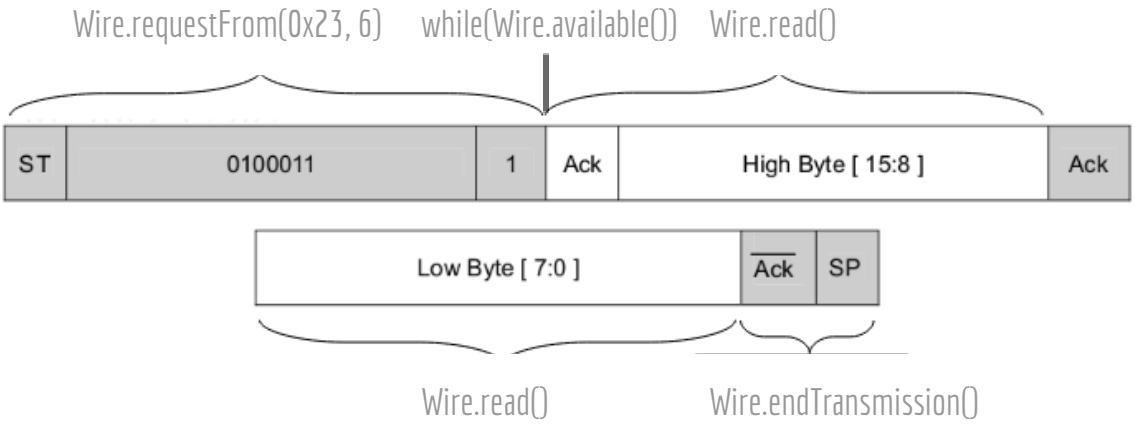


1.2 Étude de la librairie Wire

Pour l'écriture d’un octet, nous allons avoir besoin de fonction :



Pour la lecture de deux octets, on va avoir besoin :



La fonction Wire.available() nous renvoie le nombre de byte qui sont disponible à la récupération avec la fonction Wire.read().

1.3 Une première lecture de la luminosité

On commence par initialiser le capteur avec le paramètre 0x01 (pour allumer le capteur), 0x07 (pour reset le capteur), et 0x10 (pour configurer le capteur en mode HR).

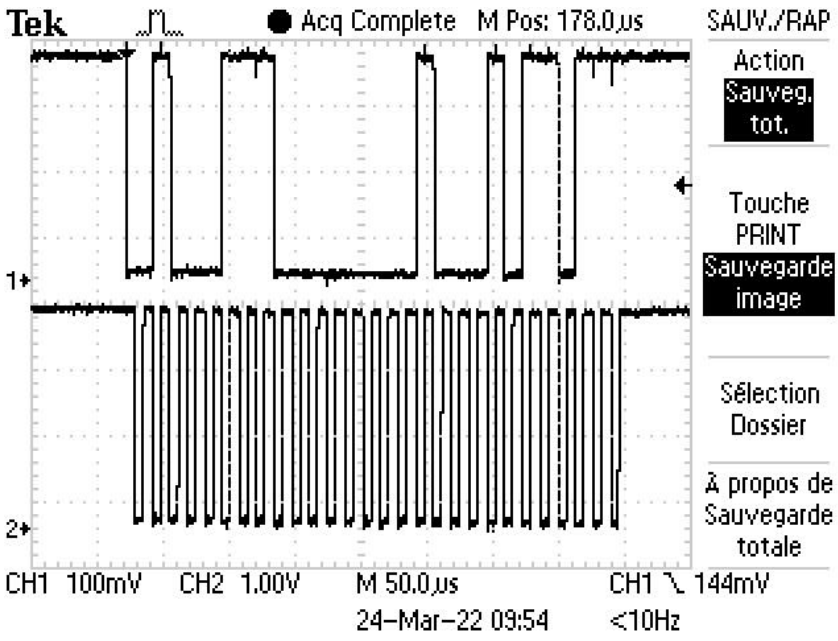
```
void setup()
{
  Serial.begin(9600);

  Wire.begin();
  i2cWrite8(capteur,0x01);//ON
  i2cWrite8(capteur,0x07);//RESET
  i2cWrite8(capteur,0x10);//MODE HR

  delay(300);
}

void i2cWrite8( uint8_t address , uint8_t data ){
  Wire.beginTransmission(address);
  Wire.write(data);
  Wire.endTransmission();
}
```

On oublie pas le delay en fin de setup, afin de laisser le temps au première valeur d'arriver et d'être mise en attente.

Pour la partie de la récupération de la trame grâce à la bibliothèque Wire, on va récupérer dans un premier temps les bits de poid fort, et ensuite les bites de poid faible. On assemble les 2 et on arrive à récupérer la valeur que l'on veut.

```
void loop()
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Serial.println(i2cRead16(capteur));
  }
}

uint16_t i2cRead16( uint8_t address ) {
  int lux = 0;
  Wire.requestFrom(address, 2);
  while (!Wire.available());
  lux = Wire.read();
  while (!Wire.available());
  lux = (lux << 8) + Wire.read();
  return lux;
}
```

On peut également vérifier que l'on reçoit bien la trame grâce à un oscilloscope :



Trame SDA

Trame SCL

1.4 Conception d'une classe C++ pour accéder au circuit BH1750

On va partir de la bibliothèque fournie sur moodle, on va donc dans un premiers temps aller regarder le fichier '.h'.

```
#ifndef BH1750_h
#define BH1750_h

#include "Arduino.h"

#include "Wire.h"

#define ADDR 0x23
#define POWER_ON 0x01
#define RESET 0x07
#define CNT_HR 0x10
#define BH1750_TIME_OUT 200 // temps max d'attente de la réponse (ms)
#define BH1750_ERROR 0xFFFF // valeur retournée si le circuit ne répond pas


class BH1750 {
  public:
    BH1750();  // constructeur
    void begin(void); // initialisation du circuit
    void setAddress( uint8_t address = 0x23) ; // fixe adresse du circuit
    void setMode(uint8_t mode ); // change le mode de fonctionnement
    void reset(void); // reset du circuit
    uint16_t getRawLight(void); // lit le résultat brut d'une mesure

  private:
    void i2cWrite8( uint8_t data ) ; // écrit un octet
    uint16_t i2cRead16( ) ; // lit 16 bits
    uint8_t address ;
    uint8_t mode ;
};
#endif
```

Le but de ce fichier, est de définir les variables que l'on va utiliser, mais aussi de déclarer les fonctions que l'on va utiliser (ou non). On voit également la définition de la classe BH1750, avec ces composants publics et privés.

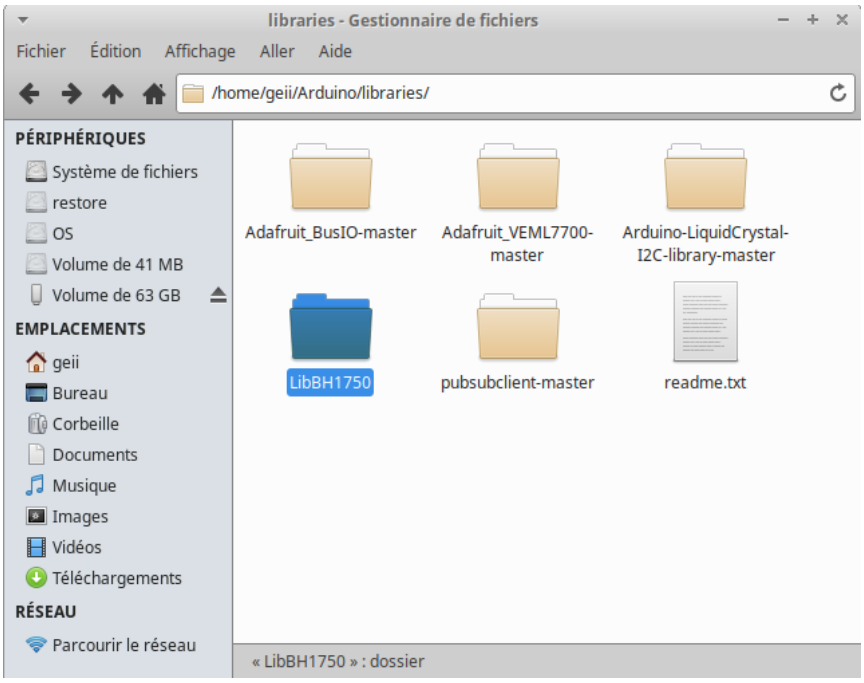
On va maintenant regarder la partie la plus intéressante, le fichier '.cpp'. C'est l'endroit où on va trouver nos fonctions qu'on pourra utiliser dans le programme principale.

```
#include "bh1750.h"

BH1750::BH1750() {

}

void BH1750::begin(void) {
  i2cWrite8(POWER_ON);
  i2cWrite8(RESET);//RESET
  i2cWrite8(CNT_HR);//MODE HR
  delay(300);
}

uint16_t BH1750::getRawLight(void) {
  uint16_t RawLight = i2cRead16();
  return RawLight;
}


void BH1750::i2cWrite8( uint8_t data ) {
  Wire.beginTransmission(ADDR);
  Wire.write(data);
  Wire.endTransmission();
}

uint16_t BH1750::i2cRead16(void) {
  int lux = 0;
  Wire.requestFrom(ADDR, 2);
  while (!Wire.available());
  lux = Wire.read();
  while (!Wire.available());
  lux = (lux << 8) + Wire.read();
  return lux;
}
```

Ici, on a juste repris notre programme de base, qu'on a séparé en plusieurs fonctions distinctes. On renomme juste les variables pour qu'elle soit correspondante au '.h' . On a aussi notre fonction '.getRawLight' qui va permettre de passer la valeur de la fonction '.i2cRead16' en public, et permettre de récupérer la valeur dans le programme principale.

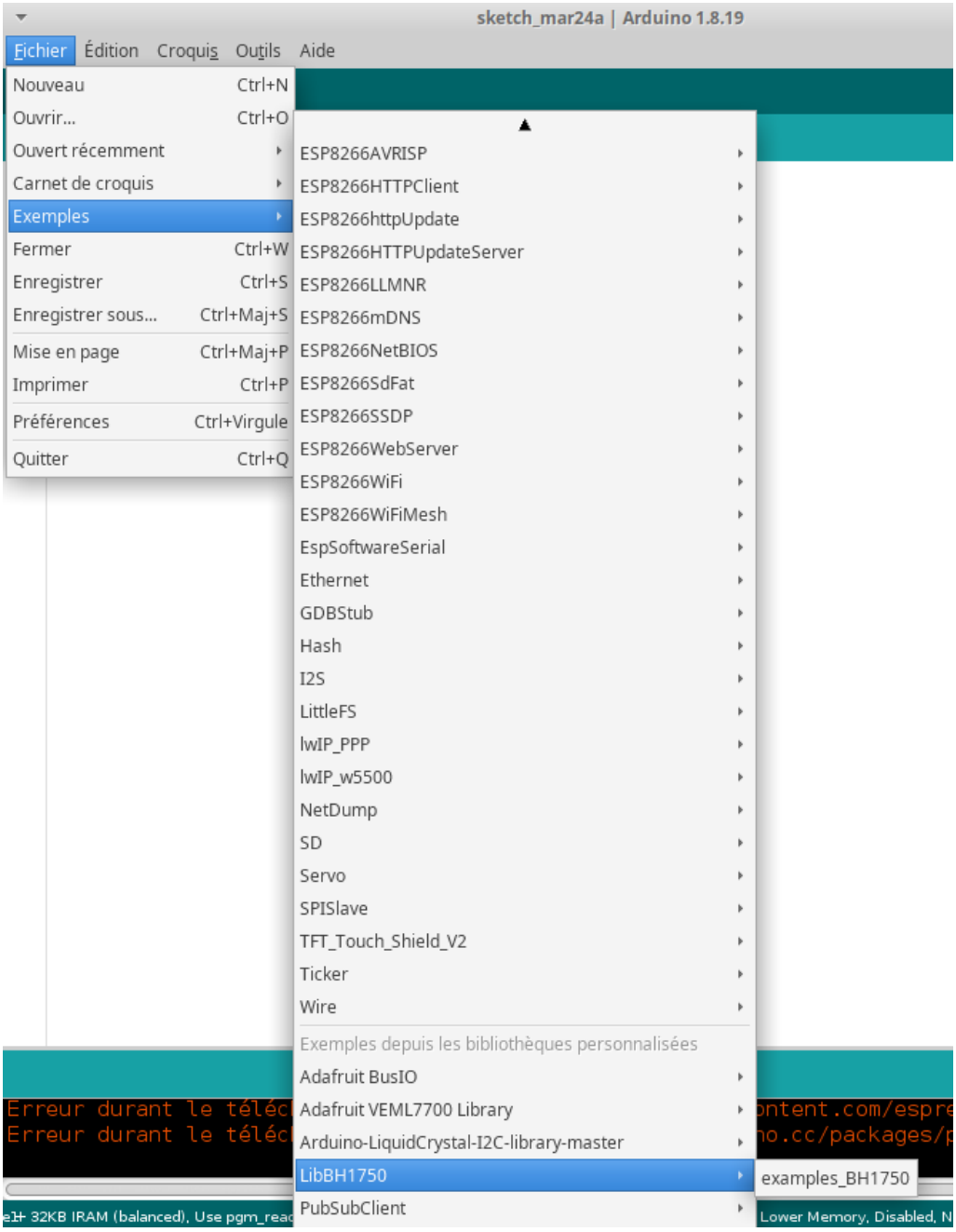
1.5 Intégration dans le système de libraires Arduino

Maintenant que notre bibliothèque marche bien en local, on va essayer de l'intégrer à l'IDE arduino, afin de vérifier que tout puisse bien marcher. On va donc dans le fichier root d'Arduino, on ouvre son fichier librairie, et on crée un fichier au nom de **LibBH1750**.



Afin de compléter notre librairie, il faut qu'on puisse trouver un exemple de comment marche notre bibliothèque. On va donc mettre notre fichier arduino qui nous a permis de tester notre programme.

On redémarre arduino afin de pouvoir refaire faire un scan de ces bibliothèques.



Notre bibliothèque avec son adresse a bien été prise en compte.

Afin de peaufiner un petit peu, on peut aussi rajouter un fichier “library.properties” afin de donner les informations générales de la bibliothèque.

/home/geii/Arduino/libraries/LibBH1750/library.properties - Mousepad

Fichier Édition Rechercher Affichage Document Aide

```
name=LibBH1750
version=0.0.1
author=Mereci
maintainer=Mereci (no)
sentence=Arduino library for the BH1750 light sensor
paragraph=Arduino library for the BH1750 light sensor
category=Sensors
url=https://www.youtube.com/watch?v=dQw4w9WgXcQ
architectures=*
```

On crée également un fichier ‘keywords.txt’ qui va permettre de mettre en couleur nos fonctions.

/home/geii/Arduino/libraries/LibBH1750/keywords.txt - Mousepad

Fichier Édition Rechercher Affichage Document Aide

```
########################################
# Syntax Coloring Map For LibBH1750
########################################

########################################
# Datatypes (KEYWORD1)
########################################

BH1750  KEYWORD1

########################################
# Methods and Functions (KEYWORD2)
########################################

getRawLight     KEYWORD2
i2cWrite8       KEYWORD2
i2cRead16       KEYWORD2

########################################
# Constants (LITERAL1)
########################################
```

Insérez votre texte ici Insérez votre texte ici

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Insérez votre texte ici

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.