Chapitre IV : La mémoire EEPROM

I.1 Introduction

Lorsque vous stockez une valeur dans une variable, la carte Arduino ne conserve cette valeur que tant qu'elle est sous tension. Dès que vous éteignez la carte, toutes les données en mémoire sont perdues (elles sont volatiles). Voyons dans ce chapitre quelles techniques sont disponibles pour rendre certaines données permanentes.

I.2 L'écriture et la lecture vers l'EEPROM

Le circuit ATMega328 qui anime la carte Arduino Uno est doté de 1 Ko de mémoire EEPROM effaçable électriquement (Electrically Erasable Read-Only Memory). Les valeurs stockées dans cette mémoire EEPROM restent accessibles pendant des années.
Malgré son nom, cette mémoire n'est pas en lecture seule. Vous pouvez tout à fait y écrire des données.
Les commandes d'écriture et de lecture vers la mémoire EEPROM dans Arduino sont aussi malaisées que celles pour la directive PROGMEM. En particulier, la lecture et l'écriture en EEPROM doivent se faire un octet à la fois.

Exemple 1

Le programme suivant stocke la valeur lue a partir l'entrée analogique dans de l' EEPROM.

```
/*************************************************************************/
      #include <EEPROM.h>
      //the current address in the EEPROM
      int addr = 0;

      void setup(){}

      void loop(){
      // need to divide by 4 because analog inputs range from 0 to 1023 and
      // each byte of the EEPROM can only hold a value from 0 to 255.
      int val = analogRead(0) / 4;
      EEPROM.write(addr, val);
      // advance to the next address.  there are 1024 bytes in
      // the EEPROM, so go back to 0 when we hit 1024.
       addr = addr + 1;
      if (addr == 1024)
       addr = 0;

      delay(100);
      }
/*************************************************************************/
```

La fonction nommée EEPROM.write() attend deux arguments dont le premier est l'adresse mémoire EEPROM de stockage qui doit se situer entre 0 et 1023. Le second argument contient la donnée à écrire à cette adresse.

Exemple 2

Le programme suivant lire les valeurs stockées dans l'EEPROM et les envoyer vers la laison série.

```
/*************************************************************************/
      #include <EEPROM.h>
      // start reading from the first byte (address 0) of the EEPROM
      int address = 0;
      byte value;
      void setup()
      {
        Serial.begin(9600);
      }
      void loop()
      {
        // read a byte from the current address of the EEPROM
       value = EEPROM.read(address);
      Serial.print(address);
      Serial.print("\t");
      Serial.print(value);
      Serial.println();
      address = address + 1;
      if (address == 1024)
      address = 0;
       delay(500);
      }
/*************************************************************************/
```

I.3 Effacement du contenu de la EEPROM

À partir du moment où vous utilisez la mémoire EEPROM, vous devez vous souvenir que même en téléchargeant un nouveau programme, vous n'effacez pas le contenu actuel de la EEPROM. Vous risquez donc rapidement de laisser des valeurs s'accumuler inutilement de vos projets précédents.

Le programme suivant efface la totalité du contenu de la mémoire EEPROM en y écrivant des 0 :

```
/*************************************************************************/
      #include <EEPROM.h>
      void setup()
      {
      Serial.begin(9600);
      Serial.println("Effacement EEPROM");
      for (int i = 0; i <= 1023; i++)
      {
      EEPROM.write(i, 0);
      }
      Serial.println("EEPROM vidée");
```

```
}
void loop()
{
}
```

Le nombre maximal d'écritures dans chaque adresse de la mémoire EEPROM est d'environ 100 000, après quoi les données peuvent devenir non fiables. De plus, les accès à cette mémoire sont assez lents, puisqu'il faut environ 3 millisecondes pour écrire un octet.

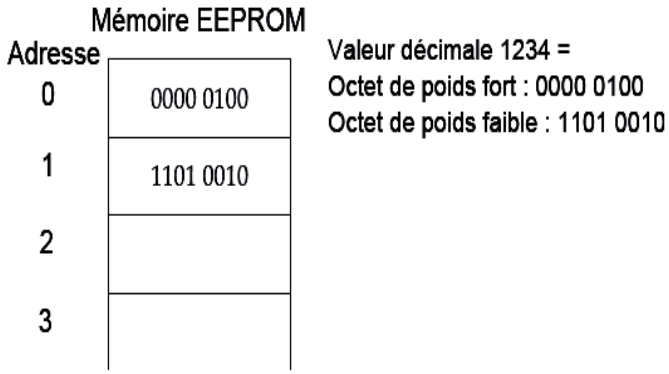
```
/*************************************************************************/
```

I.4 Stockage d'un int en EEPROM

Voici comment procéder pour stocker une valeur int (2 octets) dans les adresses 0 et 1 de la mémoire EEPROM :

```
int x = 1234;
EEPROM.write(0, highByte(x));
EEPROM.write(1, lowByte(x));
```

Nous utilisons les deux fonctions prédéfinies highByte() et lowByte() pour récupérer les deux octets de la valeur int. La Figure suivante montre comment ce int est stocké dans la mémoire EEPROM.



Pour relire la valeur int depuis la mémoire EEPROM, il faut bien sûr relire les deux octets puis reconstruire la valeur int :

```
byte octetH = EEPROM.read(0);
byte octetB= EEPROM.read(1);
int x = (octetH << 8) + octetB;
```

L'opérateur << est un opérateur de décalage de bits qui déplace les 8 bits de poids fort dans la moitié