Utilisation d'une EEPROM

Une mémoire morte est une mémoire utilisée pour enregistrer des informations qui ne doivent pas être perdues lorsque l'appareil qui les contient n'est plus alimenté en électricité. La caractéristique de la mémoire PROM est qu'elle ne peut être programmée qu'une fois. Une fois programmée, elle devient une mémoire morte (ROM ou *Read Only Memory*) au sens littéral du terme. À la différence d'une mémoire PROM qui ne peut être programmée qu'une seule fois, une mémoire EEPROM peut être effacée et reprogrammée plusieurs fois (de 100 000 à 1 000 000 de fois) et peut être lue à l'infini. Alors qu'il faut la retirer de l'appareil et la soumettre à un rayonnement ultra-violet pour effacer la mémoire EPROM (aussi appelée UVEPROM), la mémoire EEPROM peut être effacée par un simple courant électrique sans qu'il ne soit nécessaire de la retirer de l'appareil qui la contient.

EEPROM interne à la carte Arduino Uno

La carte Arduino contient une mémoire morte de 1 kio. 1024 octet c'est peu mais on peut déjà avoir beaucoup d'applications (ça correspond à un millier de caractères, soit plus de six SMS complets).
Chacun des 1024 octets a son adresse. Pour écrire un octet sur l'eeprom, on indique l'adresse de l'octet de 0 à 1023 et sa valeur de 0 à 255 : on importe la librairie pour l'eeprom interne en ajoutant *#include <EEPROM.h>* en début de programme puis on utilise *EEPROM.write(*adresse,valeur*);* autant que l'on veut. Pour lire on utilise, de la même librairie l'instruction *EEPROM.read(*adresse*);*.

Programme pour écrire un octet :	Programme pour lire un octet :
EEPROMunoEcriture1Octet	EEPROMunoLecture1Octet

#include <EEPROM.h>
 	
void setup()
{ EEPROM.write(0b0011100,0x41); }
 	
void loop()
{ }
	#include <EEPROM.h>
	int c;
	
	void setup(){
	Serial.begin(9600);
	delay(5000);
	c = EEPROM.read(28);
	Serial.print(c);
	Serial.print(" correspond au code ascii : ");
	Serial.print((char)c);
	}
	
	void loop()
	{ }

Les valeurs et les adresses peuvent indifféremment être écrites en décimal, binaire ou hexadécimal : par exemple, 28 = **0b**0011100 = **0x**1C .

Exercice : écrire 109 (mise en décimal) à l'adresse 512 (mise en hexadécimal). Lire cette donnée en écrivant l'adresse en binaire. Quel est le code ascii correspondant ? Comment s'écrit 109 en binaire et hexadécimal, expliquer pourquoi.

Évaluation de l'enseignant à demander pour cet exercice.

(*American Standard Code for Information Interchange* « Code américain normalisé pour l'échange d'information » est la norme de codage de caractères en informatique la plus connue, la plus ancienne et la plus largement compatible. ASCII contient les caractères nécessaires pour écrire en anglais.)

Code en base				Caractère	Signification
10	8	16	2		
0	0	00	0000000	NUL	*Null* (nul)
1	01	01	0000001	SOH	*Start of Header* (début d'en-tête)
2	02	02	0000010	STX	*Start of Text* (début du texte)
3	03	03	0000011	ETX	*End of Text* (fin du texte)
4	04	04	0000100	EOT	*End of Transmission* (fin de transmission)
5	05	05	0000101	ENQ	*Enquiry (End of Line)* (demande, fin de ligne)
6	06	06	0000110	ACK	*Acknowledge* (accusé de réception)
7	07	07	0000111	BEL	Bell (caractère d'appel)
8	010	08	0001000	BS	Backspace (espacement arrière)
9	011	09	0001001	HT	Horizontal Tab (tabulation horizontale)
10	012	0A	0001010	LF	Line Feed (saut de ligne)
11	013	0B	0001011	VT	Vertical Tab (tabulation verticale)
12	014	0C	0001100	FF	Form Feed (saut de page)
13	015	0D	0001101	CR	Carriage Return (retour chariot)
14	016	0E	0001110	SO	Shift Out (fin d'extension)
15	017	0F	0001111	SI	Shift In (démarrage d'extension)
16	020	10	0010000	DLE	Data Link Escape
17	021	11	0010001	DC1	Device Control 1 à 4 (DC1 et DC3 sont généralement utilisés pour coder XON et XOFF dans un canal de communication duplex)
18	022	12	0010010	DC2	
19	023	13	0010011	DC3	
20	024	14	0010100	DC4	
21	025	15	0010101	NAK	Negative Acknowledge (accusé de réception négatif)
22	026	16	0010110	SYN	Synchronous Idle
23	027	17	0010111	ETB	End of Transmission Block (fin du bloc de transmission)
24	030	18	0011000	CAN	Cancel (annulation)
25	031	19	0011001	EM	End of Medium (fin de support)
26	032	1A	0011010	SUB	Substitute (substitution)
27	033	1B	0011011	ESC	Escape (échappement)
28	034	1C	0011100	FS	File Separator (séparateur de fichier)
29	035	1D	0011101	GS	Group Separator (séparateur de groupe)
30	036	1E	0011110	RS	Record Separator (séparateur d'enregistrement)
31	037	1F	0011111	US	Unit Separator (séparateur d'unité)
32	040	20	0100000	SP	Espace (Space en anglais)
33	041	21	0100001	!	Point d'exclamation
34	042	22	0100010	"	Guillemet droit
35	043	23	0100011	#	Croisillon et parfois Dièse
36	044	24	0100100	$	Dollar (symbole)
37	045	25	0100101	%	Pourcent
38	046	26	0100110	&	Esperluette
39	047	27	0100111	'	Apostrophe (guillemet fermant simple ou accent aigu)
40	050	28	0101000	(	Parenthèse ouvrante
41	051	29	0101001	)	Parenthèse fermante
42	052	2A	0101010	*	Astérisque
43	053	2B	0101011	+	Plus

44	054	2C	0101100	,	Virgule
45	055	2D	0101101	-	Moins
46	056	2E	0101110	.	Point
47	057	2F	0101111	/	Barre oblique (Slash en anglais)
48	060	30	0110000	0	Le chiffre zéro
49	061	31	0110001	1	Le chiffre un
50	062	32	0110010	2	Le chiffre deux
51	063	33	0110011	3	Le chiffre trois
52	064	34	0110100	4	Le chiffre quatre
53	065	35	0110101	5	Le chiffre cinq
54	066	36	0110110	6	Le chiffre six
55	067	37	0110111	7	Le chiffre sept
56	070	38	0111000	8	Le chiffre huit
57	071	39	0111001	9	Le chiffre neuf
58	072	3A	0111010	:	Deux-points
59	073	3B	0111011	;	Point-virgule
60	074	3C	0111100	<	Inférieur
61	075	3D	0111101	=	Égal
62	076	3E	0111110	>	Supérieur
63	077	3F	0111111	?	Point d'interrogation
64	0100	40	1000000	@	Arrobe
65	0101	41	1000001	A	
66	0102	42	1000010	B	
67	0103	43	1000011	C	
68	0104	44	1000100	D	
69	0105	45	1000101	E	
70	0106	46	1000110	F	
71	0107	47	1000111	G	
72	0110	48	1001000	H	
73	0111	49	1001001	I	
74	0112	4A	1001010	J	
75	0113	4B	1001011	K	
76	0114	4C	1001100	L	
77	0115	4D	1001101	M	
78	0116	4E	1001110	N	
79	0117	4F	1001111	O	
80	0120	50	1010000	P	
81	0121	51	1010001	Q	
82	0122	52	1010010	R	
83	0123	53	1010011	S	
84	0124	54	1010100	T	
85	0125	55	1010101	U	
86	0126	56	1010110	V	
87	0127	57	1010111	W	
88	0130	58	1011000	X	
89	0131	59	1011001	Y	

90	0132	5A	1011010	Z	
91	0133	5B	1011011	[	Crochet ouvrant
92	0134	5C	1011100	\	Barre oblique inversée (backslash en anglais) ; également nommée Antislash
93	0135	5D	1011101	]	Crochet fermant
94	0136	5E	1011110	^	Accent circonflexe
95	0137	5F	1011111	_	Tiret bas ou souligné (underscore en anglais)
96	0140	60	1100000	`	Accent grave
97	0141	61	1100001	a	
98	0142	62	1100010	b	
99	0143	63	1100011	c	
100	0144	64	1100100	d	
101	0145	65	1100101	e	
102	0146	66	1100110	f	
103	0147	67	1100111	g	
104	0150	68	1101000	h	
105	0151	69	1101001	i	
106	0152	6A	1101010	j	
107	0153	6B	1101011	k	
108	0154	6C	1101100	l	
109	0155	6D	1101101	m	
110	0156	6E	1101110	n	
111	0157	6F	1101111	o	
112	0160	70	1110000	p	
113	0161	71	1110001	q	
114	0162	72	1110010	r	
115	0163	73	1110011	s	
116	0164	74	1110100	t	
117	0165	75	1110101	u	
118	0166	76	1110110	v	
119	0167	77	1110111	w	
120	0170	78	1111000	x	
121	0171	79	1111001	y	
122	0172	7A	1111010	z	
123	0173	7B	1111011	{	Accolade ouvrante
124	0174	7C	1111100	|	Barre verticale
125	0175	7D	1111101	}	Accolade fermante
126	0176	7E	1111110	~	Tilde
127	0177	7F	1111111	DEL	*Delete* (effacement)

Nous allons ensuite, pour la pression, simuler la montée du ballon sonde dans l'atmosphère avec la cloche à vide.
Attention : montez le capteur de pression sur la plaque multiconnexion blanche nue.
Montez le capteur de pression avec la lecture analogique sur A0 de la carte Arduino.
Pour le programme vous vous inspirerez de celui utilisez pour la journée sidaction , vous prendrez une mesure de pression toutes les secondes :

BallonFilinEcriture

```
 #include <EEPROM.h>
int a;     // adresse sur l'eeprom
int P;     // presion sur dix bits
int B0aB7; // les huit premiers bits
int B8;    // le neuvième
int B9;    // le dixième
int B8etB9;

void setup()
{
   pinMode(13, OUTPUT);//indicateur
   digitalWrite(13, HIGH);
   delay(8000); //pause initiale
for(a=0; a<1024; a=a+2) {
   digitalWrite(13, LOW);
  P=analogRead(0);//valeur 0 à 1023
  B9=int(P/512);         // 0 ou 1
  B8=int((P-512*B9)/256);// 0 ou 1
  B0aB7=P-512*B9-256*B8; //un octet
  B8etB9=B8+2*B9;
  EEPROM.write(a,B0aB7);
  EEPROM.write(a+1,B8etB9);
   delay(2000);
   digitalWrite(13, HIGH);
   delay(2000);
                      }
}

void loop()
{
   digitalWrite(13, HIGH);
   delay(100);
   digitalWrite(13, LOW);
   delay(100);
}
```

Faire un test avec la pompe à vide manuelle (en dépression !).
L'enseignant placera votre montage avec celui de trois autres groupes sous la cloche à vide.
On notera la pression du pressiomètre avant dépression et au minimum, à l'arrêt de la pompe, pour étalonner le capteur.
Lire les données et les analyser sur un tableur (la courbe sera imprimée et analysée avec les constantes de temps).

BallonFilinLecture

```
 #include <EEPROM.h>
int a; int P;
int B0aB7;
int B8; int B9;
int B8etB9;

void setup()
{
  pinMode(13, OUTPUT);
```

```
 Serial.begin(9600);
 for(a=0; a<1024; a=a+2) {
 B0aB7 = EEPROM.read(a);
 B8etB9 = EEPROM.read(a+1);
 B9=int(B8etB9/2);// 0 ou 1
 B8=B8etB9-2*B9;  // 0 ou 1
 P=B0aB7+256*B8+512*B9;
 Serial.print(P);
 Serial.println(";");
 delay(500);
  }
}
void loop()
{
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
}
```

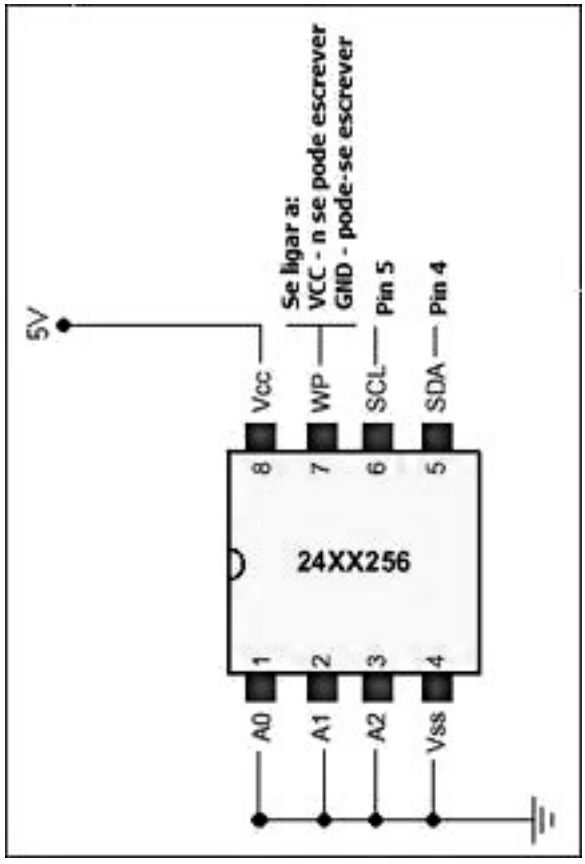
Évaluation de l'enseignant à demander pour cet exercice.

EEPROM externe 24LC256 de 32 kio

La carte Arduino va communiquer via un bus I2C avec l'eeprom.
I²C (pour *Inter Integrated Circuit*) est un bus composé de trois fils :

- un signal de donnée (SDA) ;
- un signal d'horloge (SCL) ;
- un signal de référence (masse).

Le bus I²C fut développé par Philips pour les applications de domotique et d'électronique domestique au début des années 1980, notamment pour permettre de relier facilement à un microprocesseur les différents circuits d'une télévision moderne.



Imaginez combien de fils seraient reliés entre la carte et l'eeprom si on n'utilisait pas un protocole de communication mais une connexion directe avec la mémoire morte : 32 kio = 32 x 8 x 1024 bits, plus la masse, ça fait 262 145 fils au lieu de 3 ici... De toute façon nous n'avons que 13 entrées/sorties numériques sur la Uno !

La patte 5 de l'eeprom 24LC256 est reliée au pin A4 de la carte UNO. Et la patte 6 au pin A5.

La patte 7 est reliée à la masse.

Ici on écrit avec la fonction *eeprom_ecrire_octet(* composant*,* adresse*,* valeur*);* où nous avons une entrée en plus, en premier, qui désigne le numéro du composant concerné. L'adresse pour une mémoire de 32 kio va de 0 à 32 767, et la valeur toujours un octet de 0 à 255.

Le numéro, adresse du composant, est déterminé par le branchement de A0, A1 et A2 de l'eeprom 24LC256. Tous les trois sont ici à zéro et cette

adresse vaut 0x50 = 80. Si nous ajoutons une eeprom nous pourrons la mettre en 0x51 avec A0 à 5V. Nous pouvons au total mettre 8 eeprom de ce type soit 256 kio.

Pour lire la fonction est *i2c_eeprom_lire_octet(* composant, adresse*);* .

EEPROM24LC256Ecriture1Octet

```
#include <Wire.h> //I2C library

void i2c_eeprom_ecrire_octet( int deviceaddress, unsigned int eeaddress, byte data )
{ int rdata = data;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8)); // MSB
  Wire.send((int)(eeaddress & 0xFF)); // LSB
  Wire.send(rdata);
  Wire.endTransmission();
}
void setup() {
  Wire.begin(); //initialise la connection
  Serial.begin(9600);
  delay(3000);
  i2c_eeprom_ecrire_octet(0x50, 0x3A, 0b1100001);
  delay(10);
  Serial.println("Memoire ecrite !");
          }
void loop()
{   }
```

EEPROM24LC256Lecture1Octet

```
#include <Wire.h>
byte i2c_eeprom_lire_octet( int deviceaddress, unsigned int eeaddress )
{
  byte rdata = 0xFF;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8)); // MSB
  Wire.send((int)(eeaddress & 0xFF)); // LSB
  Wire.endTransmission();
  Wire.requestFrom(deviceaddress,1);
  if (Wire.available()) rdata = Wire.receive();
  return rdata;
}
void setup() {
  Wire.begin();
  Serial.begin(9600);
```

```
  delay(3000);
  byte b = i2c_eeprom_lire_octet(80, 0b0111010);
  Serial.print((char)b);
  Serial.println(" ! ");
          }
 void loop()
 {  }
```

Exercice : même exercice qu'avec l'eeprom de l'Arduino pour le code ascii qui correspond à un point d'interrogation.

Évaluation de l'enseignant à demander pour cet exercice.

Utiliser l'eeprom pour la photo-résistance en pont diviseur. Comme au premier TP Arduino nous plaçons la photo-résistance en série avec une résistance de 10 kΩ. Ici nous enregistrons les données sur 2 kio tous les deux centièmes de seconde. La lecture est très rapide au centième de seconde avec un port série à 57600 bauds.
Les variations de luminosité sont ainsi enregistrées avec une grande précision.

EEPROM24LC256EcritureA0

```
 #include <Wire.h> //I2C library

int a;     // adresse sur l'eeprom
int P;     // presion sur dix bits
int B0aB7; // les huit premiers bits
int B8;    // le neuvième
int B9;    // le dixième
int B8etB9;

void i2c_eeprom_ecrire_octet( int deviceaddress, unsigned int eeaddress, byte data )
 {
  int rdata = data;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8)); // MSB
  Wire.send((int)(eeaddress & 0xFF)); // LSB
  Wire.send(rdata);
  Wire.endTransmission();
 }

 void setup()
 {
  pinMode(13, OUTPUT);//indicateur
  digitalWrite(13, HIGH);
  delay(5000); //pause initiale
  digitalWrite(13, LOW);
  Wire.begin();
 // i2c_eeprom_ecrire_octet(0x50, 0x3A, 0b1100001);
 for(a=0; a<2048; a=a+2) {
  digitalWrite(13, LOW);
 P=analogRead(0);//valeur 0 à 1023
 B9=int(P/512);         // 0 ou 1
 B8=int((P-512*B9)/256);// 0 ou 1
 B0aB7=P-512*B9-256*B8; //un octet
 B8etB9=B8+2*B9;
 i2c_eeprom_ecrire_octet(0x50, a,byte(B0aB7));
  delay(10);
 i2c_eeprom_ecrire_octet(0x50, a+1,byte(B8etB9));
  delay(10);           }
```

```
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

EEPROM24LC256LectureA0

```
 #include <Wire.h>

int a; int P;
int B0aB7; byte B0B7;
int B8; int B9;
int B8etB9; byte B8B9;

 byte i2c_eeprom_lire_octet( int deviceaddress, unsigned int eeaddress )
 {
  byte rdata = 0xFF;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8)); // MSB
  Wire.send((int)(eeaddress & 0xFF)); // LSB
  Wire.endTransmission();
  Wire.requestFrom(deviceaddress,1);
  if (Wire.available()) rdata = Wire.receive();
  return rdata;
 }

 void setup()
 {
  pinMode(13, OUTPUT);
  Wire.begin();
  Serial.begin(57600);
  delay(8000);
 //  byte b = i2c_eeprom_lire_octet(80, 0b0111010);
  // Serial.print((char)b);
  // Serial.println(" ! ");

for(a=0; a<2048; a=a+2) {
 B0B7 = i2c_eeprom_lire_octet(80, a);
 B0aB7 = int(B0B7);
 B8B9 = i2c_eeprom_lire_octet(80,a+1);
 B8etB9 = int(B8B9);
  B9=int(B8etB9/2);// 0 ou 1
  B8=B8etB9-2*B9;  // 0 ou 1
  P=B0aB7+256*B8+512*B9;
  Serial.print(a);
  Serial.print(";;");
  Serial.print(P);
  Serial.println(";");
  delay(10);
                    }
 }

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Évaluation de l'enseignant à demander pour cet exercice.

Liste du matériel :

par binôme :

- Ordi avec le logiciel Arduino
- Kit Arduino
- connecteur pile 9V > Alimentation carte Arduino
- pile 9V
- Plaque multiconnexion blanche nue pour le capteur de pression
- capteur de pression MPX4115A avec embout pour la pression extérieure
- eeprom 24LC256
- pompe à vide manuelle
- petits fils intra plaque multiconnexion

pour tous :

- pompe à vide avec supports pour 4 montages et la place pour le pressiomètre
- réveil et ampoule pour présentation de la cloche
- pressiomètre
- imprimante / feuilles A4