Le code Morse international, ou l'alphabet Morse international, permet de coder et de transmettre un texte à l'aide d'impulsions courtes et longues.

Il peut être utilisé pour moduler du son, de la lumière ou même des gestes. Nous connaissons tous le message SOS envoyé par le Titanic (… — …) le point (dit) représentant une impulsion courte et le trait une impulsion longue (dash).

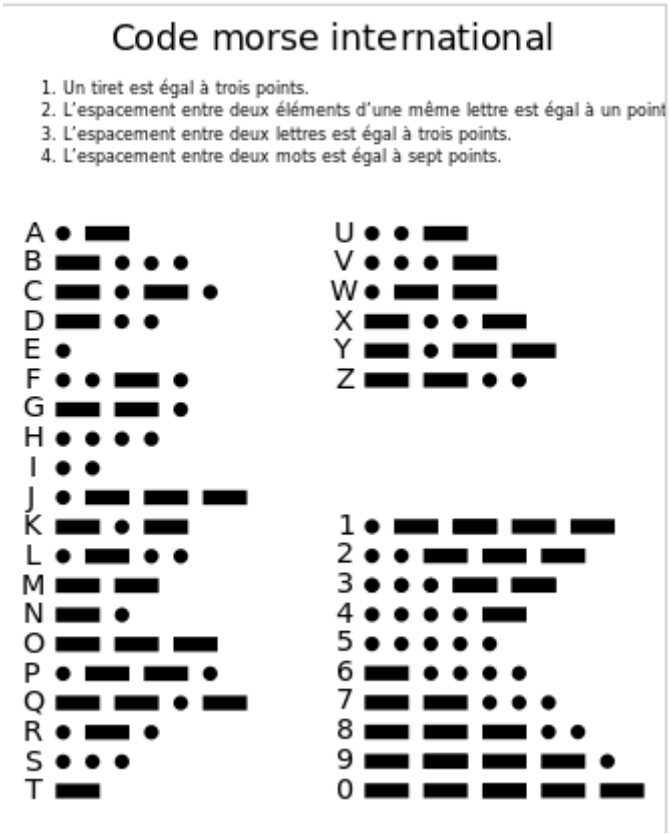
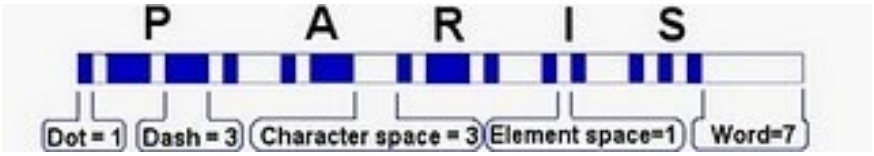


figure 1 : code Morse international

Timing du code Morse [1]:

The basic element of Morse code is the dot and all other elements can be defined in terms of multiples of the dot length. The other elements are dash (= dot length x 3), pause between elements (= dot length), pause between characters (= dot length x 3) and pause between words (= dot length x 7). An example is the word PARIS that has length of 50 dot lengths:



La vitesse en définie en mots/minute (wpm) et on a la relation :

dot length (ms) = durée du point (ms) = 1200/vitesse(wpm)

1 http://sv8gxc.blogspot.com/2010/09/morse-code-101-in-wpm-bw-snr.html

Travail proposé :

En réutilisant la classe Led du TP n°2, nous vous proposons de définir et d'écrire une classe Morse permettant de transmettre un texte sous la forme d'une chaîne de caractères (classe Arduino String[2]).

Spécification de la classe Morse :

1. Elle utilisera un objet Led préalablement instancié pour émettre le signal lumineux.
2. La vitesse en mots/minute pourra être modifiée par l'utilisateur dans le constructeur[3].
3. Elle contiendra l'alphabet Morse, bien entendu – deux codages possible sont expliqués en annexe.
4. Une méthode **setSpeed()** permettra de modifier la vitesse de transmission entre 0.5 et 20 wpm.
5. Une méthode privée permet d'envoyer un point puis d'attendre la durée d'un point.
6. Une méthode privée permet d'envoyer un trait puis d'attendre la durée d'un point.
7. Une méthode, **send()**, permet d'envoyer un caractère (type char) du code (ne fait rien si le caractère n'est pas dans le code).
8. Une méthode, **send()**, permettra d'envoyer une chaîne de caractères (type String) fournie en argument.
9. Deux assesseurs **getSpeed()** et **getDotLenght()** permettent d'obtenir la vitesse et la durée du point.

2 https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/
3 Une valeur par défaut de 3.5 wpm permet de "lire" facilement le code à l'œil.

ANNEXE CODAGE DE L'ALPHABET MORSE

Deux solutions peuvent être envisagées pour coder l'alphabet Morse :

1 - un tableau de chaînes de caractères char ** contenant l'alphabet sous une forme ASCII

```
const char *morse[] = {".-.-.-","--..--","..--..", "-..-.",".-", "-...", "-.-.",
"-..",".","..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", "-.", "---",
".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--..",
".----","..---","...--","....-",".....","-....","--...","---..","----.","-----"}
;
```

Cette solution, très simple à comprendre, présente tout de même l'inconvénient de son occupation mémoire. (le nombre de caractères total ajouté des '\0' de terminaison des chaînes).

2 - Un codage plus judicieux où chaque code est représenté sur un octet unique, voir tableau ci-dessous. Cette solution consiste à utiliser les bits de l'octet, de droite à gauche, pour représenter les '-' et les '.'

ex : A = 6 → 00000110 → En partant de la gauche, on cherche le premier « 1 ». Après ce premier « 1 » il reste donc deux bits (deux signes pour coder A).

On part maintenant de la droite : 0→ « . » puis 1 → « - », donc A= « .- »

Sym.	**déc.**	**binaire**	**morse**				
.	106	01101010	.-.-.-	Q	27	00011011	--.-
,	115	01110011	--..--	R	10	00001010	.-.
?	76	01001100	..--..	S	8	00001000	...
/	41	00101001	-..-.	T	3	00000011	-
A	6	00000110	.-	U	12	00001100	..-
B	17	00010001	-...	V	24	00011000	...-
C	21	00010101	-.-.	W	14	00001110	.--
D	9	00001001	-..	X	25	00011001	-..-
E	2	00000010	.	Y	29	00011101	-.--
F	20	00010100	..-.	Z	19	00010011	--..
G	11	00001011	--.	1	62	00111110	.----
H	16	00010000	....	2	60	00111100	..---
I	4	00000100	..	3	56	00111000	...--
J	30	00011110	.---	4	48	00110000	....-
K	13	00001101	-.-	5	32	00100000	.....
L	18	00010010	.-..	6	33	00100001	-....
M	7	00000111	--	7	35	00100011	--...
N	5	00000101	-.	8	39	00100111	---..
O	15	00001111	---	9	47	00101111	----.
P	22	00010110	.--.	0	63	00111111	-----

```
const uint8_t morse[]={106, 115, 76, 41, 6, 17, 21, 9, 2, 20, 11, 16, 4, 30, 13,
18, 7, 5, 15, 22, 27, 10, 8, 3, 12, 24, 14, 25, 29, 19, 62, 60, 56, 48, 32, 33,
35, 39, 47, 63} ;
```

Le code c des tableaux ci-dessous peut être copier à partir de à : http://grimaldi.univ-tln.fr/files/code-morse.c