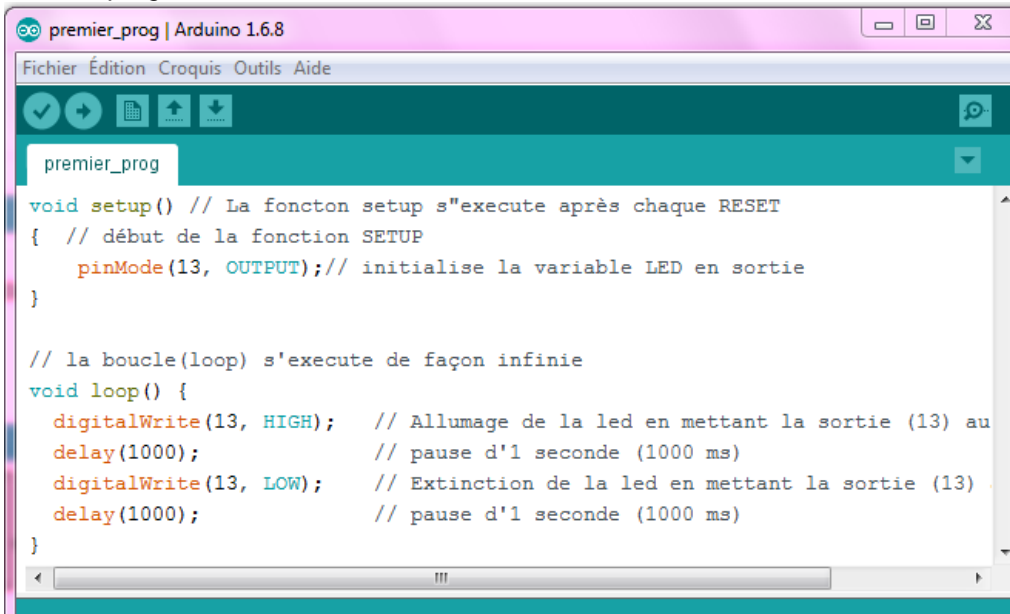


Premier programme

Ecrire le programme ci-dessous.



```
premier_prog | Arduino 1.6.8
Fichier Édition Croquis Outils Aide
premier_prog
void setup() // La fonction setup s'exécute après chaque RESET
{ // début de la fonction SETUP
  pinMode(13, OUTPUT); // initialise la variable LED en sortie
}

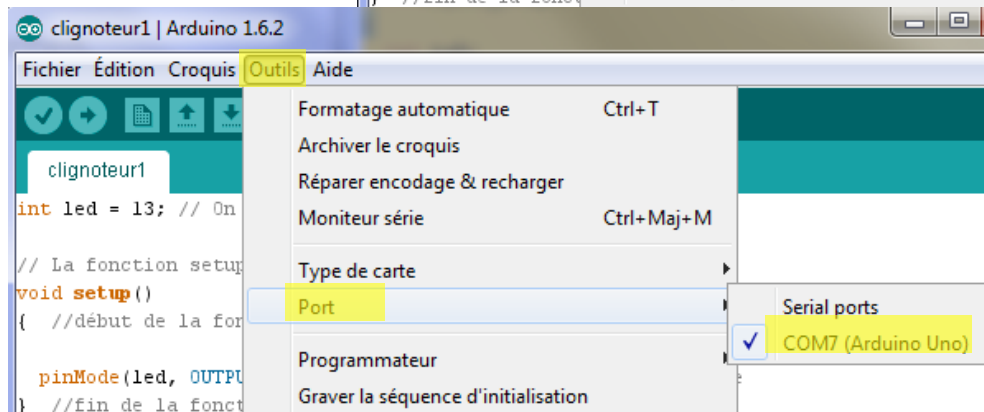
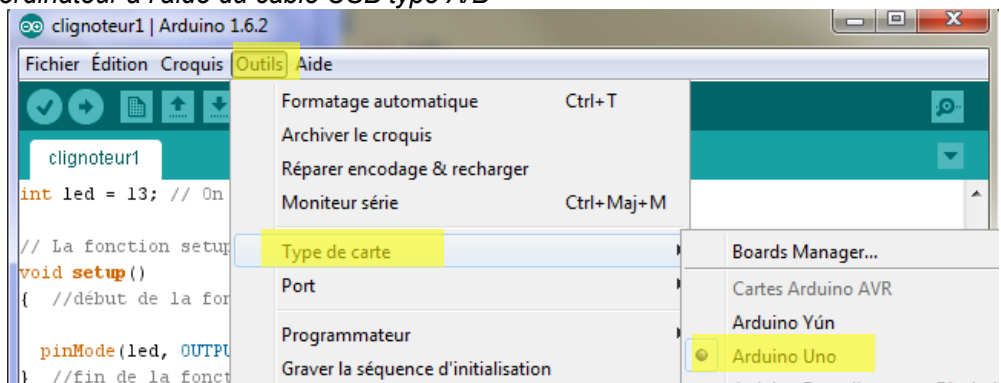
// la boucle(loop) s'exécute de façon infinie
void loop() {
  digitalWrite(13, HIGH); // Allumage de la led en mettant la sortie (13) au
  delay(1000); // pause d'1 seconde (1000 ms)
  digitalWrite(13, LOW); // Extinction de la led en mettant la sortie (13)
  delay(1000); // pause d'1 seconde (1000 ms)
}
```

Compiler¹ le
programme



Brancher la carte arduino à l'ordinateur à l'aide du câble USB type A/B

Vérifier que le type de
carte associée est bien
carte UNO



Vérifier que la carte est
reconnue par l'ordinateur
et se situe sur bon port.

La valeur de COM peut naturellement différer de COM7

Télécharger le programme on peut aussi dire téléverser



Expliquer le fonctionnement du programme en décrivant son action.

¹ **Compilation**, en informatique : travail réalisé par un compilateur qui consiste à transformer un code source lisible par un humain en un fichier binaire exécutable par une machine.

Génération d'un signal SOS

Nous allons créer pour l'Arduino un programme permettant de générer un signal **MORSE lumineux d'un SOS**.

Rappelons que le SOS correspond à 3 signaux courts suivi de 3 longs puis à nouveau de 3 courts.

Nous choisissons donc une durée :

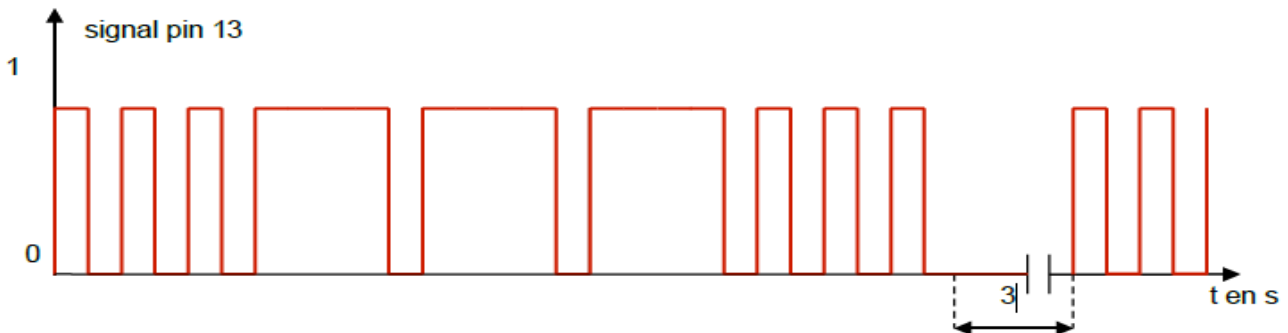
- d'allumage et d'extinction de 250ms pour les signaux courts,
- d'allumage de 1000ms et d'extinction de 250ms pour les signaux longs



De plus :

- Les appels de détresse seront espacés de 3 secondes.
- La DEL sera affectée à la broche 13 de la carte (la broche 13 comporte déjà une DEL).

Chronogramme de la broche 13



Premier programme

En reprenant la structure du programme précédent, écrire le programme, et faites valider son fonctionnement. La structure du programme reste linéaire après des allumages et des extinctions on utilise des delays.

Ce programme est assez mauvais, car il n'y a en fait que 2 opérations à réaliser : le clignotement court et le clignotement long.

Afin d'améliorer le programme, il est possible **d'insérer des boucles** qui nous permettront de compter 3 allumages courts, puis 3 allumages longs, puis à nouveau 3 allumages courts. Pour cela nous devons créer une variable de comptage noter "i".

Les différents types de variables susceptibles d'être utilisées sont les suivantes :

| Type de variable | Type de nombre | Valeur du nombre | Nombre de bits | Nombre d'octet |
|------------------|--------------------|---|----------------|----------------|
| int | Entier | De -32768 à + 32767 | 16 | 2 |
| long | Entier | De - 2147483648 à - 2147483647 | 32 | 4 |
| char | Entier | De -128 à 127 | 8 | 1 |
| float | Décimal | De $-3,4028235 \times 10^{+38}$ à $3,4028235 \times 10^{+38}$ | 32 | 4 |
| double | Décimal | De $-3,4028235 \times 10^{+38}$ à $3,4028235 \times 10^{+38}$ | 32 (sur uno) | 4 |
| unsigned char | Entier non négatif | De 0 à 255 | 8 | 1 |

| | | | | |
|---------------|--------------------|-------------------|----|---|
| unsigned long | Entier non négatif | 0 à 4 294 967 295 | 32 | 4 |
| unsigned int | Entier non négatif | 0 à 65535 | 16 | 2 |
| byte | Entier non négatif | 0 à 255 | 8 | 1 |
| word | Entier non négatif | 0 à 65535 | 16 | 2 |
| boolean | Entier non négatif | 0 à 1 | 1 | 1 |

Choisir le type de la variable "i" la plus appropriée à notre application (on rappelle que i ne dépassera pas la valeur 3)

La variable i doit nous permettre de réaliser une boucle du type :

| | |
|---|--|
| Algorithme | Syntaxe ARDUINO (et langage C) |
| TANT QUE $i < 3$ FAIRE "clignotement rapide"... | While (condition sur "i") { instructions;} |

Le tableau ci-dessous liste les opérateurs communs

| Opérateur | Dénomination | effet | Syntaxe |
|-----------|------------------------|--|-----------------------------|
| == | égalité | si variable1 = variable2 résultat vrai sinon résultat faux | ((variable1)==(variable2)) |
| != | différent de | si variable 1 \neq variable 2 résultat vrai sinon résultat faux | ((variable1)!= (variable2)) |
| < | plus petit que | si variable 1 < variable 2 résultat vrai sinon résultat faux | ((variable1)<(variable2)) |
| <= | plus petit ou égal | si variable 1 \leq variable 2 résultat vrai sinon résultat faux | ((variable1)<=(variable2)) |
| > | Plus grand que | si variable 1 > variable 2 résultat vrai sinon résultat faux | ((variable1)>(variable2)) |
| >= | Plus grand que ou égal | si variable 1 \geq variable 2 résultat vrai sinon résultat faux | ((variable1)>=(variable2)) |

Pour incrémenter la variable i il suffit d'écrire l'instruction :

$i=i+1$; ou : $i++$;

Modifier le programme en insérant les boucles de type While pour les différents types de clignotement de la LED.

Faire valider votre programme.

Utilisation de fonctions

Notre programme, même amélioré n'est pas encore un programme propre au sens de la programmation. Pour qu'il le devienne, il faut réaliser des fonctions (des sous programmes) réutilisables par le programme principal.

Il existe deux types de fonctions, **les fonctions de type "void"** qui ne renvoient pas de valeur au programme principal après leurs exécutions et **les fonctions "typées"** qui elles, retournent une valeur vers le programme principal après leurs exécutions.

Création de la fonction S

Juste après la fonction `setup()`, créer la fonction "S" telle que:

```
void S( ) {  
  byte i=0;  
  while (i<3){  
    digitalWrite(led,HIGH); // écrit un niveau haut sur la variable LED, la LED s'allume  
    delay(250); // attente pendant 250ms  
    digitalWrite(led, LOW); // écrit un niveau bas sur la variable led, la led s'éteint  
    delay(250); //attente pendant 250ms  
    i = i+1;  
  }  
}
```

Création de la fonction O

De la même manière que la fonction "S", créer la fonction "O"

Création de la fonction « DETRESSE »

Créer la fonction "DETRESSE" de telle manière que le programme principal ne contienne que les lignes suivantes :

```
void loop( ) {  
  DETRESSE( );  
  delay(3000);  
}
```

Votre programme principal ne contient plus que 2 lignes, c'est une programmation « propre ».

Création d'une fonction typée

Créer le code ci-dessous en utilisant au maximum le code précédemment écrit

Remarque : le nombre issu de l'exécution de la fonction soustraction est un entier.

La fonction typée "soustraction" renvoie une valeur, De quel type est-elle ?

Que se passe t-il tant que $x > y$?

Que se passe t-il lorsque $x = y$?

Que se passe t-il tant que $x < y$?

Représenter le programme principal sous forme d'algorithme.

Inverser x et y afin d'obtenir `soustraction(y,x)` à la place de `soustraction(x,y)`. Compiler et tester. Que constater vous par rapport au fonctionnement initial ?

Remplacer `nombre1` par x et `nombre2` par y , le programme fonctionne t-il toujours ? Les variables x et y sont elles globales, ou locales, c'est à dire utilisable partout ou seulement dans le programme principal?

```
int led = 13; // On affecte la variable led à la broche 13

// La fonction setup s'exécute après chaque reset

void setup()
{
  pinMode(led, OUTPUT); // initialise la variable led en sortie
}

void S()
{
  byte i=0;
  while (i<3)
  {
    digitalWrite(led,HIGH); // écrit un niveau haut sur la variable led, la led s'allume
    delay(250); // attente pendant 250ms
    digitalWrite(led, LOW); // écrit un niveau bas sur la variable led, la led s'éteint
    delay(250); // attente pendant 250ms
    i = i+1;
  }
}

void O()
{
  byte i=0;
  while (i<3)
  {
    digitalWrite(led,HIGH); // écrit un niveau haut sur la variable led, la led s'allume
    delay(1000); // attente pendant 500ms
    digitalWrite(led, LOW); // écrit un niveau bas sur la variable led, la led s'éteint
    delay(250); // attente pendant 500ms
    i = i+1;
  }
}

int soustraction (int nombre1,int nombre2)
{
  int total=0;
  total=nombre1-nombre2;
  return total;
}

int x = 10;
int y = 3;
int resultat = 0;

void loop()
{
  resultat = soustraction(x,y);
  if (resultat >0) { S() ; delay (2000);}
  else if (resultat < 0) { O() ; delay (2000);}
  else {digitalWrite (led,HIGH);delay (3000);digitalWrite (led,LOW);delay(1000);}
  y=y+1;
}
```