http://www.didel.com/ **info@didel.com**

www.didel.com/diduino/Cours02.pdf

Cours Arduino/C 2ème partie

La première partie se trouve sous www.didel.com/diduino/Cours01.pdf
Cette 2e partie permet d'étudier les 4 structures de commandes importantes du C: `if, for, while, do .. while` et d'aller vers de programmes toujours plus intéressants, D'autres exemples de programmes utilisant ces structures se trouvent sous www.didel.com/diduino/Composants.pdf et naturellement dans les innombrables exemples sur le Web.
Mais avant, découvrons un gadget très utile.

2.1 Le terminal série

Le C peut s'apprendre sur un PC ou Mac sans utiliser une carte processeur Arduino ou autre. Mais on perd la dimension "temps réel" et toutes les applications avec des capteurs et actuateurs.
L'écran affiche "Hello World" comme premier exercice. Les données sont données au clavier et l'écran affiche les résultats. Le site du zéro a un excellent cours:
http://www.siteduzero.com/tutoriel-3-14189-apprenez-a-programmer-en-c.html#part_14188

Arduino permet aussi d'afficher sur l'écran, avec des notations légèrement différentes et des limitations. Via la ligne USB, des fonctions (ordres) permettent de balancer des caractères sur l'écran et lire le clavier. Très utiles pour la mise au point des programmes temps réel qui nous intéresse, mais le temps de communication avec le PC ou Mac fait perdre cet aspect "temps réel" (à la microseconde près) qui est la raison d'utiliser une carte Arduino.
Dans le set-up, il faut dire que l'on va communiquer avec l'écran et préciser la vitesse de transfert

```
void  setup () {
      Serial.begin (9600) ;
      pinmode …
}
```

Dans la boucle de programme, on prépare un texte, le nom d'une variable dont on veut afficher le contenu: `Serial.print ("la variable toto est égale à ") ;`
Un `Serial.println ( ) ;` fait passer à la ligne suivante
Quand le programme est chargé, on clique sur l'icône tout à droite et une fenêtre apparaît.

Testons d'abord le programme

```
//TestSerie.ino

void setup() {
  Serial.begin(9600);
  pinMode(Led, OUTPUT);
}
void loop() {
    Serial.print("J'allume   ");
    LedOn
    delay(1000);
    Serial.println("J'éteins");
    LedOff
    delay(1000);
}
```

Mettez delay(1); Le clignotement n'a pas la période souhaitée de 2 millisecondes; l'affichage ralentit le clignotement, et l'écran se remplit à vitesse max !

Comme autre exemple, comptons les clignotements.

```
// TestSerie2.ino
#define Led 13
#define LedOn digitalWrite(Led,HIGH);
#define LedOff digitalWrite(Led,LOW);
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}
int compte = 0 ;
```

```
void loop()
{
    Serial.print("J'allume   ");
    LedOn;
    delay(1000);
     compte++ ;
    Serial.print( "J'eteint pour la " );
    Serial.print( compte );
    Serial.println( "eme fois" );
    LedOff;
    delay(1000);
}
```

2.2 Commande if

La commande `if` a trois formes

`if (condition) {`
`  instructions ;`
`}`
	Si la condition est vraie, les instructions entre { } sont exécutées.
	Autrement le processeur passe plus loin.

`if (condition) {`
`  instructions ;`
`} else {`
`  instructions ;`
`}`
	Si la condition est vraie, les instructions entre { } sont exécutées.
	Autrement le processeur exécute le 2e groupe d'instructions

`if (condition) {`
`  instructions ;`
`} else if { (autre condition)`
`  instructions ;`
`} else { // tous les autres cas`
`  instructions ;`
`}`
	Si la condition est vraie, les instructions entre { } sont exécutées.
	Si la 2e condition est vraie, les instructions qui suivent sont exécutées
	On peut naturellement continuer les else if
	Pour les autres cas, si on veut faire qqchose, on ajoute un `else`

Comme application de cette 3e forme, qui n'a pas été rencontrée dans la première partie de ce cours, on pourrait câbler deux poussoirs, et décider quelle Led on allume.

2.3 Commande while (condition) { }

Le while (condition) permet de boucler on non selon une condition vrai/faux, ou une valeur >0/=0 . Le groupe d'instructions entre { } est exécuté jusqu'à ce que la condition devienne fausse. Donc `while (1){}` boucle indéfiniment !

Reprenons comme exemple le son continu désagréable à la longue. Avec un while, on peut le couper dès que l'on presse sur le poussoir.

```
//Son2.ino   Si on presse, le son s'arrête
// Le poussoir sur la pin9 est câblé vers le – avec une pull-up de 4k7 à 100k
// La haut-parleur sur la pin8 est câblé vers le + avec une résistance de 47-470 Ohm
#define Hp8 8
#define HpOn digitalWrite(Hp,HIGH);
#define HpOff digitalWrite(Hp,LOW);
#define Pous9 9
#define PousOn  !digitalRead(Pous9)
void setup () {
  pinMode (Hp8, OUTPUT);
  pinMode (Pous9, INPUT);
}
void loop () {
   while ( !PousOn ) {
        HpOn;
        delayMicroseconds (300) ;
        HpOff;
        delayMicroseconds (300) ;
    }
   while (1) {}  // On reste là . Reset ou nouveau chargement pour ré-écouter
}
```

Exercice: câbler un 2e poussoir, dont l'action permettra de se retrouver au début de la boucle.

Rappelons la différence: avec un `if` , si la condition est vraie, on exécute les instructions et on continue. Avec un `while`, on exécute les instructions en boucle tant que la condition est vraie, et on passe plus loin quand elle est fausse.

2.4 Exercice

On a câblé une Led et un poussoir. Un pèse, la Led s'allume, on repèse, elle s'éteint. A cause des rebonds de contact expliqués dans www.didel.com/diduino/Composants.pdf il faut attendre 5ms toutes les fois que l'on lit le poussoir.

2.5 Commande do ..while

La syntaxe du `do..while` est

```
do {
   instructions ; }
while (condition) ;
```

Tant que la condition est vraie, on exécute en boucle les instructions du bloc.
La différence avec le while est que la condition est testée après les instructions, qui se font donc au moins une fois, alors que avec le while, si la condition est fausse dès le départ, on n'exécute rien.
A noter le bon usage de mettre les accolades autrement.

Comme exemple, demandons via le Terminal de presser sur le poussoir et redemander tant que ce n'est pas fait. Avec un `do..while` le message apparaîtra même si l'utilisateur presse en anticipation.

```
//Merci.ino Test do while
const int Pous9=9;
void setup() {
    Serial.begin(9600);
    pinMode(Pous9, INPUT);
}
void loop () {
    do {
        Serial.println ("Presse svp ");
        delay(1000);  }
    while (digitalRead (Pous9) == HIGH) ;
    Serial.println ("Merci");
    delay (2000) ;
}
```

2.6 Commande for

Prenons un exemple simple. On veut compter de 1 à 10 et afficher sur le terminal.
Il faut définir un compteur et dire au programme

j'initialise le compteur à 1
j'affiche le compteur tant que la condition "mon compteur est inférieur ou égal à 10" est vraie
chaque fois j'augmente de 1

```
void loop () ; {
    for  (compteur=1 ; compteur <= 10 ; compteur ++ ) {
       Serial.print (compteur) ;
       Serial.print ("  ") ;
       delay (100) ;
    }
    delay (1000) ;
}
```

Chargez ce programme (TestFor.ino) .
Notons bien les 3 parties dans la parenthèse d'une boucle `for` :

```
for (initialization; condition; modification)
```

"modification" veut dire qu'il faut agir sur quelque chose (un calcul, une action extérieure) pour que la condition change et que l'on sorte de la boucle.

2.8 Sirène

La boucle `for` se prête bien pour faire une sirène comme en 1.17

```
void loop () {
    for (demiPeriode=DemiPerMax ; demiPeriode>=PeriodeMin; demiPeriode-=Increment) {
       HpToggle;
```

```
        delayMicroseconds (demiPeriode) ; }
    for (demiPeriode=DemiPerMin ; demiPeriode<=PeriodeMax ; demiPeriode+=Increment) {
        HpToggle;
        delayMicroseconds (demiPeriode) ; }
    }
}
```

A compéter et tester comme exercice
Vous voyez où rajouter un `while (Pous9 == HIGH)` pour couper le son? (solution SireneFor.ino)
La sirène évolue trop vite, comment la ralentir? Avec un boucle for qui joue n fois la même période.

2.9 Anticipons

Pour faire varier l'intensité de la Led, on peut s'inspirer de la sirène pour décider quand on allume et éteint, et faire cela assez rapidement pour que l'on ne voie pas le clignotement.
C'est plus simple avec la fonction `analogWrite (pin, %intensité);`, que l'on verra plus tard, qui agit sur la pins 3 et quelques autres (3, 5, 6, 9, 10, and 11).
On voit aussi dans cet exemple que la variable `i` est déclarée dans le `for`, pour la durée du `for`,

```
//LedVarie  delay(59; fixe la vitesse de variation
const int Led3=3;
void setup()
{
    pinMode (Led3, OUTPUT);
}
void loop ()
{
    for (int i=0 ; i<256 ; i++ )
    {
    analogWrite(Led3,i);
    delay (20) ;
    }
    for (int i=255 ; i>0 ; i-- )
    {
    analogWrite(Led3,i);
    delay (20) ;
    }
}
```

2.10 Règles d'écriture

Le C est "case sensitive" mais à part cela vous pouvez nommer les variables et constantes comme vous voulez. Mais si vous ne voulez pas vous faire traiter de mauvais programmeur au premier coup d'oeil jeté à votre programme, il faut suivre les règles suivantes

- les constantes ont un nom qui commence par une majuscule `ValeurMax`
- les variables ont un nom qui commence par une minuscule `demiPeriode`
- les noms des identificateurs sont significatifs (pas de `var, toto)`
- ne pas utiliser des noms qui se ressemblent trop
- éviter le risque de confusion entre 0 et o, 1 et l
- indenter les lignes pour faire ressortir la structure syntaxique du programme
- bien commenter l'objectif, l'environnement matériel, la structure de chaque bloc
- une ligne vide peut être un excellent commentaire
- une accolade fermante est seule sur une ligne, sauf pour le do..while

1.10 Assimiler

Il faut passer du temps, c'est à dire relire la doc, modifier des exemples, s'en inventer, avant d'être à l'aise avec ces commandes de base du C et respecter scrupuleusement la discipline de mise en page qui aide à la lisibilité, pour soi quand on écrit, pour les autres quand ils lisent.
Si vous demandez sous Google "Arduino while" vous tombez sur une page explicative qui vous aidera ou vous troublera. Nous n'avons encore vu qu'une partie des termes utilisés pour décrire la fonctionnalité du C.

Pour continuer, si vous avez le kit de composants Diduino, jouer avec ces composants et faire les programmes www.didel.com/diduino/Composants.pdf permet d'exercer le C avant de s'attaquer aux notions qui vous manquent encore et qui permettent d'aller vers des applications complexes.

Vous trouverez la suite de ce cours sous www.didel.com/diduino/Cours03.pdf dans quelques semaines.

Le tableau www.didel.com/coursera/ResumeDebutant.pdf est utile pour se rafraîchir la mémoire quand on hésite.