Structure

```
void setup()      // Initialisation
{
                  // Bloc d'instruction
}
void loop()       // Boucle Principale
{
                  // Bloc d'instruction
}
```

Commentaires

```
// Commentaire simple sur une ligne
/*
  Bloc de commentaire sur plusieurs lignes
*/
```

Définition et Inclusion

```
#define ulong uint32_t // Macro
#include <Arduino.h>  // Biblio système
#include ''Test.h''   // Biblio personnel
```

Opérateurs

```
= assignement          + addition
- soustraction         * multiplication
/ division             % modulo
== égalité             != différent
<  plus petit que      >  plus grand que
<= plus petit ou égale
>= plus grand ou égale && et logique
|| ou logique          !  négation
^ ou exclusif          ~ complément
<< rotation gauche     >> décalage droite
```

Opérateurs Composés

```
++x x++  Incrément     --x x-- décrément
+=  addition           -= soustraction
*=  multiplication     /= division
%=  modulo             >>= décalage D
<<= rotation G         ^= ou exclusif
&=  et bit à bit       |=  ou bit à bit
```

Constantes

```
HIGH (1)  LOW (0)  INPUT (0)  OUTPUT (1)
true (1)  false (0)
```

Base

```
143          // Décimal (base 10)
0175         // Octal   (base 8)
0b00101011   // Binaire (base 2)
0xF5         // Hexa    (base 16)
```

Forçage des Types

```
8U           // Force non signé
10L          // Force Long
12UL         // Force Long non signé
10.0         // Force Flottant
2.4e5        // Décalage Virgule
```

Pointeur

```
& (adresse de la variable pointé)
*  (accès à la variable pointé)
```

Types Variable

```
void     // sans type (Fonction)
bool     // Booléen 0 ou 1              (1 octet)
char     // -128 à +127                 (1 octet)
byte     // 0 à 255                     (1 octet)
int      // -32768 à +32767             (2 octets)
word     // 0 à 65535                   (2 octets)
long     // -2147483648 à 2147383647 (4 o)
ulong    // 0 à 4294967295              (4 octets)
float    // -3.4028235e38 à 3,402835e38
double   // = float                     (4 octets)
int y = sizeof(x)  // Taille en octet de x
NULL     // ASCII 0, valeur Pointeur 0
```

Chaîne

```
char S1[15];     // Chaîne de 15 caractères
char S2[5] = {'D','e','b','u','t'};
char S3[] = ''Debut'';  // Assignation
S3[2] = 'f';            // Assignation index
String Val;             // Autre type Chaîne
Val = ''Debut'';        // Assignation
```

Tableau

```
Int Val[6];      // Tableau de 6 valeurs
word val[] = {2, 3, 4, 5};
Val[2] = 12;     // Assignement index
byte[4][5];      // Tableau à 2 dimensions
```

Structure

```
struct modele  // Utilise Types différents
{
byte un;       // Premier membre
bool deux;     // Deuxième membre
float trois;   // Troisième membre...
};             // Doit finir par ';'
```

Conversion de type

```
char(x) (char)   // Conversion Char
byte(x) (byte)   // Conversion Byte
int(x)  (int)    // Conversion Entier Signé
word(x) (word)   // Conversion Entier !Signé
long(x) (long)   // Conversion Long
float(x) (float) // Conversion Flottant
```

Qualifieurs

```
static   // Persistant entre deux utilisations
volatile // Utilise la RAM (interruption)
const    // Constante en lecture seule
PROGMEM  // Utilise la mémoire Flash
```

Bit et Octet

```
lowByte();   highByte();   bitRead(x, bitn)
bitWrite(x, bitn, bit);  bitSet(x, bitn);
bitClear(x, bitn); bit(bitn); // 0-LSB 7-MSB
```

Ordre de Priorité

```
Parenthèses  ! ++ -- - (signe) sizeof
* / %  + - <  <=  >  >=  == !=
&(adresse de)  &&  ||  =  (+= ...)  ,
```

Entrée/Sortie Digital

```
pinMode(port, [INPUT/OUTPUT]); // Mode
digitalWrite(port, val); // Écrit sur le port
int x = digitalRead(port); // Lire le port
// Écrire HIGH Entrée = résistance Pull-Up
```

Entrée/Sortie Analogique

```
analogReference([DEFAULT, INTERNAL,
EXTERNAL]) ; // Définit la référence +5v
int x = analogRead(port); // Lire Analogie
analogWrite(port, val);  // Écrit en PWM
```

Entrée/Sortie Avancé

```
tone(port, Hz);  // Bip à la fréquence Hz
tone(port, Hz, temps); // Bip Hz pendant ms
noTone(port);    // Arrêt du Bip
ulong x = pulseIn(port, [HIGH/LOW]); //
Mesure d'impulsion sur front haut ou bas
```

Temps

```
ulong x = millis();    // Temps en ms (50 j)
ulong y = micros() ; // Temps en µs (70 min)
delay(ms);  // Délais en ms ( ! interruption)
delayMicroseconds(µs); // Délais en µs
```

Pseudo Aléatoire

```
randomSeed(val);        // Fixe le premier nbr
long x = random(max); // Valeur aléatoire
long y = random(min, max) ; // Valeur...
```

Mathématique

```
min(x, y); // Mini          max(x,y); // Maxi
abs(x);                    // Valeur Absolue
constrain(x, min, max); // Contenue entre
map(val, deL, deH, aL, aH);
pow(base, exposant);    // Élévation
sqrt(x);                // Racine carrée
sin(rad);               // Sinus
cos(rad);               // Cosinus
tan(rad);               // Tangente
PI                      // 3.1415926
```

Test Conditionnel

```
if (x<5) { } else { }     // Test conditionnel
x = (y > 4) ? -1 : +1;    // Test en ligne
switch (x)                // Choix multiples
{
 case 1:                  // Choix 1
   break;
 case 2:                  // Choix 2
   break;
 default:                 // Choix par défaut
}
```

Boucle

```
for (i = 0; i < 10; i++); // Boucle Définit
for (;;);                 // Boucle infini
do                        // Boucle test après
{ // bloc } while <condition>;
while <condition>         // Boucle test avant
{ // bloc };
continue;        // Revient au début Boucle
break;           // Quitte la Boucle
```

Fonction

```
void Fonction() // Fonction sans retour
{
 // Bloc d'instruction
}
byte Fonction() // Fonction avec retour
{
 return x;       // Valeur de retour
}
```

Interruption

attachInterrupt(num, fonction, mode);
Uno = Int0 port 2, Int1 port 3
Mega = Int2 port 21, Int3 port 20, Int4 port 19, et Int5 port 18
Mode = LOW, CHANGE, RISING et FALLING
detachInterrupt(interrupt);
interrupts(); // Active les Interruptions
noInterrupts(); // Désactive Interruptions

String (Chaîne de caractères)

```
S = String(x);   // Conversion en Chaîne
S[2] = 'c';              // Accès à l'élément
S3 = S.concat(S2);       // Concaténation
S3 = "De" + "but";       // Concaténation
val = S.length();        // Longueur de S
val = S.equals(S2);      // Test Égalité
val = S.equalsIgnoreCase(S2) ;// Egalité
val = S.compareTo(S2);// Comp. Longueur
S2 = S.substring(pos);  // Extraction pos...
S2 = S.substring(deb, fin); // Extraction
S2 = S.toLowerCase();  // Minusclue
S2 = S.toUpperCase();  // Majuscule
S2 = S.trim();   // Enlève Espace deb et fin
S2 = S.replace("abc", "def"); // Remplace
S2 = S.replaceFirst("abc", "def"); // Premier
S2 = S.replaceAll("abc", "def");   // Tout
val = S.startsWith("abc");// Commence par
val = S.endsWith("def");  // Finie par
val = S.indexOf('c');      // Position de 'c'
val = S.lastIndexOf('c', pos); // Après pos
S2 = S.charAt(pos);      // Extrait caract.
S[] = S.getBytes();      // Conv Tableau Byte
S[] = S.getCharts();     // Conv Tableau Char
S[] = S.toCharArray();   // Conv Tableau Char
S2 = S.split(pos);       // Découpe
val = (S1 == S2);        // Comparaison Ptr
```

EEPROM (#include <EEPROM.h>)

```
Byte x = EEPROM.read(adr); // Lecture byte
EEPROM.write(adr, val);       // Écriture byte
```


Communication Série

```
Serial.begin(baud);      // Vitesse
Serial.end();            // Fin d'utilisation
int x = Serial.available(); // Caractère Dispo
int y = Serial.read();   // Lecture d'un octet
Serial.flush();          // vide le buffer
Serial.peek();           // lire sans décalage
Serial.print();          // Imprime
Serial.println(); // Imprime avec saut ligne
Serial.write();   // Écrire un octet
```

Série Logiciel («#include <SoftwareSerial.h>);

```
SoftwareSerial(portRx, portTx);
Serial1.begin(Baud); // Vitesse ≥ 9600 baud
char x = Serial1.read(); // Lire char
Serial1.print();    // Affichage
Serial1.println(); // Affichage avec saut ligne
```

Communication I2C (#include <Wire.h>)

```
.begin();          // Initialisation Master
.begin(port);      // Init Esclave
.beginTransmission(adr);// Début
.endTransmission();        // Fin
.requestFrom(adr, cpt);  // Demande data
.send(val);                // Envoie Donnée
.send(char * val);         // Par pointeur
.send(byte * val, taille); // Par pointeur
byte x = .available();     // Nbr Byte dispo
byte y = .receive();       // Bit suivant
.onReceive(focntion);      // Sur Reception
.onRequest(fonction);      // Sur Requête
```

Communication SPI (#include <SPI.h>)

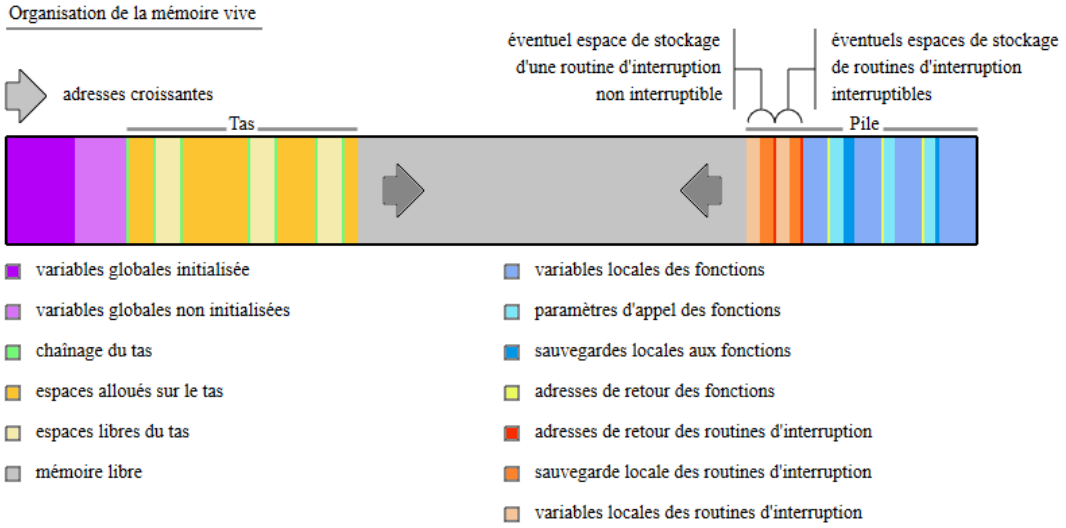
```
pinMode(ss, OUTPUT);  // Sélection en out
.begin();             // Initialise SPI
.setBitOrder(MSBFIRST/LSBFIRST);// Mode
digitalWrite(ss, LOW);   // Activer SPI
.transfert(val);         // Envoie la valeur
digitalWrite(ss, HIGH);  // Désactiver SPI
shiftOut(portD, portClk, [MSBFIRST/
LSBFIRST], val); // Envoie data SPI
byte x = shiftIn(portD, portClk, MSBFIRST/
LSBFIRST);       // Recevoir data SPI
```

Port et Communication

Types	Port Uno	Port Mega
Série	0 RX 1 TX	0 RX1 1 TX1 19 RX2 19 TX2 17 RX3 16 TX3 15 RX4 14 TX4
Int.	2 INT0 1 INT1	2, 3, 21, 20, 19, 18 IRQ0 à IRQ5
PWM	5, 6 Timer 0 9, 10 Timer 1 3, 11 Timer 2	0 à 13
SPI	10 SS 11 MOSI 12 MISO 13 SCK	53 SS 51 MOSI 50 MISO 52 SCK
I2C	A4 SDA A5 SCL	20 SDA 21 SCL

Servo (#include <Servo.h>)

```
Servo.attach(port, [min µs, max µs]);
Servo.write(angle);      // 0 à 180°
Servo.writeMicroseconds(µs); // Temps état
Haut (0 à 5000)
val = Servo.read();      // 0 à 180°
bool z = Servo.attached(); // Actif ?
Servo.detach();          // Détache
```



Moteur Pas à Pas (#include <Stepper.h>)

```
Stepper mpp(Res, S1, S2, S3, S4);// Init
// Res = nombre de pas du moteur 48...
mpp.setSpeed(NbTpM); // Nombre de Tour
par Minute
mpp.step(NbPas);         // Nbr de pas
// Si positif = sens normal, si négatif inverse
mpp.run(Mode);           // Fonctionnement
// Mode = FORWARD, BACKWARD, RELEASE
```

Lcd (#include <LiquidCrystal.h>)

```
LiquidCrystal lcd(Rs,En,d4,d5,d6,d7); // Init
lcd.begin(16,2); // Lcd 2 lignes de 16 caract.
lcd.write(S);       // String ou Entier
lcd.noDisplay(); // N'affiche rien sur le Lcd
lcd.display();      // Affiche le Lcd
lcd.clear();        // Efface l'écran
lcd.setCursor();  // Affiche le curseur
lcd.setCursor(col,lig); // Positionne curseur
lcd.noCursor();   // Curseur non affiché
lcd.home();         // Place curseur aux début
lcd.blink();        // Fait clignoter le curseur
lcd.noBlink();      // Arrêt clignotement
lcd.leftToRight();// Texte déplacer à gauche
lcd.autoscroll();  // Déplacement automatiq.
lcd.createChar(n, tab); // Création caract.
// tab est un ensemble de 8 lignes de 5 bits
// n entre 0 à 7 pour les 8 carat. possibles.
```