Memento Arduino

MONKEY MAKER
www.monkey-maker.com

Bases | Hardware | Fonctions

ARDUINO

Structure du code

```
//Inclusion des librairies
//Déclaration de vos variables
```
1 Déclarations

```
void setup() {
  // Code joué une seule fois au démarrage

}
```
2 Configuration Initialisation

```
void loop() {
  // Boucle de code principale se répétant
indéfiniment

}
```
3 Boucle Code Principal

Type de données

```
//Type de données
void          //Type vide
boolean       //0,1,false ou true
char          //'a' ou -128 à 127
unsigned char //0 à 255
String        //chaîne de caractères ASCII
byte          //0 à 255
int           //-32768 à 32767
unsigned int  //0 à 65535
word          //0 à 65535
long          //-2147483648 à 2147483647
unsigned long //0 à 4294967295
float         //-3.4028235E+38 à 3.4028235E+38
double        //identique à float
//Fonction de mesure de données
sizeof(variable); //Retourne la taille en octet
```

Constantes

```
//Constantes classiques
HIGH / LOW      //Niveau haut ou bas
INPUT / OUTPUT //Entrée ou sortie
true / false    //Vrai ou faux
123  //Nombre décimal classique
0127 //Nombre octal incrément de 0 à 7
B00110101 //Nombre binaire sur 8 bits
0x1F //Nombre héxadécimal incrément de 0 à F

//Fonction de forçage de type
7U //Force le chiffre en type unsigned avec U
6L //Force le chiffre en type long avec L
4UL//Force le chiffre en type unsignedlong avec UL
6.4//Force le chiffre en type float avec .
```

Opérateurs Binaires

```
Opérateurs binaires:
&  //Opérateur "et" entre deux nombres binaires
|  //Opérateur "ou" entre deux nombres binaires
^  //Opérateur "ou exclusif" entre deux nombres binaires
~  //Opérateur "non" entre deux nombres binaires
&= //Opérateur "et" avec réattribution
Exemple:
   binaire &= B00010100; //Est identique à l'opération
   binaire = binaire & B00010100;
|= //Opérateur "ou" avec réattribution
Exemple:
   binaire &= B00010100; //Est identique à l'opération s
   binaire = binaire & B00010100;
>> //Opérateur de décalage à droite d'un bit
Exemple:
   binaire = B00000111 >> 2;
   //Décale à droite de 2 bits = B00000001
<< //Opérateur de décalage à gauche d'un bit
Exemple:
   binaire = B00000111 << 3;
   //Décale à gauche de 3 bits = B00111000
```

Syntaxe

```
Les commentaires:
// double slash pour commenter sur une seule ligne
Exemple:
  int variable = 0; // commentaire
/* Slash étoile pour ouvrir un commentaire multiligne
et étoile slash pour fermer le commentaire */

Les déclarations/définitions:
#include <librairie_arduino.h> /*Pour inclure une
librairie native Arduino utiliser < et > */
#include  "librairie_racine_projet.h"  /*Pour  inclure
une librairie à la racine de votre projet utiliser "
et " */
#define CONSTANTE valeur; //pour définir une constante

Type_de_fonction nom_fonction(type1 paramètre1,...){
  code de la fonction}
/*Pour créer une fonction, déclarer le type (voir type
de données), donner lui un nom sans espace et indiquer
ou non des paramètres entre parenthèses */
Exemple création: void calcul(int var1,int var2){code}
Exemple appel de la fonction: calcul(2,3);
```

Qualificatifs des variables

```
static   /*Créer une variable visible depuis une fonction
         et préservée entre les appels de fonction*/
volatile //Inclus une utilisation via la RAM uniquement
const    //Force une variable en lecture seule
PROGMEM  //Inclus une variable dans la mémoire flash
```

Opérateurs Mathématiques

```
Opérateurs mathématiques:
=  //signe égal pour attribuer une valeur
+  //signe plus pour réaliser des additions
-  //signe moins pour réaliser des soustractions
*  //signe étoile pour réaliser des multiplications
/  //slash pour réaliser une division
%  //le signe pourcent permet de réaliser l'opération
modulo

Opérateurs mathématiques simplifiés:
+= //Pour incrémenter avec une valeur spécifique
Exemple:
   variable += 5; //Est identique à l'opération suivante
   variable = variable + 5;

-= //Pour décrémenter avec une valeur spécifique
Exemple:
   variable -= 6; //Est identique à l'opération suivante
   variable = variable - 6;

*= //Pour multiplier et réattribuer le résultat
Exemple:
   variable *= 7; //Est identique à l'opération suivante
   variable = variable * 7;

/= //Pour diviser et réattribuer le résultat
Exemple:
   variable /= 8; //Est identique à l'opération suivante
   variable = variable / 8;
```

Opérateurs Booléens

```
== //Double égal permet de tester l'égalité
!= //Exclamation égal permet de tester l'inégalité
<  //Inférieur permet de tester l'infériorité
>  //Supérieur permet de tester la supériorité
<= //Permet de tester l'infériorité ou l'égalité
>= //Permet de tester la supériorité ou l'égalité
&& //Applique la condition logique "et"
|| //Applique la condition logique "ou"
!  //Applique la condition logique "non"
```

Structures Contrôle

```
if(condition){code si condition vraie}
else{code si condition non vraie}

switch(variable){//Permet de tester plusieurs conditions
  case 1:
    code si variable = 1
    break;
  case 6:
    code si variable = 6
    break;
  default:
    break;
    code si variable différente de tous les cas listés
}

for(variable itérative; condition; itération){
  code joué pour chaque itération
  }
Exemple:
  int resultat = 2;
  for(int i=1; i<2; i++){ /*La boucle démarre avec i=1
  et continue si i<2 */
    resultat = resultat * i;
    }
    /* Equivaut à resultat * 2
     * Première itération i = 1 donc
       resultat = resultat * 1 = resultat
     * Deuxième itération i = 2 donc
       resultat = resultat * 2
     * Troisième itération la boucle s'arrête car i
       n'est plus < 2
     */

do {
  // Code à jouer dans la boucle
} while (condition);/*Boucle se répétant tant que
 * la condition est vraie
 */
while(condition){code joué en boucle} /*Boucle se
répétant tant que
 * la condition est vraie
 */

continue; /*Permet de passer à l'itération suivante
d'une boucle en
sautant le code restant avant itération */
Exemple:
int resultat = 0;
for (int i = 0; i <= 255; i ++) {
  if (i > 40 && i < 120) {  /* Si 40<i<120 passer à
  l'itération suivante */
    continue;
  }
  resultat = i; //Portion non jouée si 40<i<120
}

return variable; /*Utiliser pour renvoyer la valeur
d'une fonction
non vide */
Exemple:
int fonction(){ //Fonction de type nombre entier
  return 10;    //La fonction retourne en résultat 10
  }
```

Chaînes de caractères

```
//char sera utilisé pour déclarer un tableau de
caractères
char Str1[15];
char Str2[8]= {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8]= {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
/*String sera utilisé pour déclarer une chaîne de
caractères voir la rubrique Tableaux pour le détail */
String Str7 = "arduino";
```

Conversions

```
byte(variable) //Conversion en byte
char(variable) //Conversion en char
float(variable)//Conversion en float
int(variable)  //Conversion en int
long(variable) //Conversion en long
word(variable) //Conversion en word
String(variable) //Conversion en String
String(val, base)//Conversion en précisant la base
String(val, après_virgule)/*Conversion en précisant
le nombre de chiffre après la virgule*/
Exemples:
  String(13);      //Conversion du chiffre en String
  String(13,DEC);  /*Conversion précisant d'un décimal
  le résultat sera "13" */
  String(13,BIN);  /*Conversion précisant d'un binaire
  le résultat sera "1101"*/
  String(13,HEX);  /*Conversion précisant d'un
  héxadécimal  le résultat sera "d" */
  String(5.698, 2);/*Conversion avec 3 chiffres après
  la virgule le résultat sera "5.69" */

(unsigned long)variable
//Conversion en unsigned long

(unsigned int)variable
//Conversion en unsigned int
```

Tableaux

```
//Les tableaux de tous types
int entiers[6]; /*Déclaration d'un tableau vide
d'entiers pouvant contenir 6 chiffres */
int valeurs[] = {2, 4, 8, 3, 6}; /*Tableau à
longueur automatique suivant la valeur attribuée*/
int valeurs[6] = {2, 4, -8, 3, 2, 3};/*Tableau
pouvant contenir 6 entiers*/
char message[6] = {'h','e','l','l','o','\0'};/*Tableau
de caractère pouvant contenir 6 caractères. Attention
il faut prévoir une valeur supplémentaire pour le
caractère nulle en fin de tableau pour ce cas '\0'*/
char message[6] = {'h','e','l','l','o'};/*Identique
au précédent mais le caractère nulle est induit
automatiquement à la fin du tableau d'où la
dimension du tableau à 6 pour 5 caractères déclarés*/
char message[6] = "hello"; /*Identique au précédent
avec le caractère nulle induit automatiquement à la
fin du tableau d'où la dimension du tableau à 6 pour
5 caractères déclarés*/


//Récupérer une valeur dans un tableau
valeur = tableau[index] /*Pour récupérer une valeur
à un index spécifique du tableau */
Exemple:
  int valeurs[6] = {2, 4, -8, 3, 2, 3};
  int valeur = 0;
  valeur = valeurs[0]; /*Récupère la première valeur du
  tableau = 2. Attention l'index commence à 0, donc de 0
  à 5 pour un tableau de dimension 6 */


//Ecrire une valeur dans un tableau
tableau[index] = valeur /*Pour écrire une valeur
à un index spécifique du tableau */
Exemple:
  int valeurs[6] = {2, 4, -8, 3, 2, 3};
  int valeur = 0;
  valeurs[1] = valeur; /*Ecrit la valeur 0 à l'index
  numéro 1 du tableau, soit à la deuxième valeur.
  Ici remplace la valeur 4 par 0->{2, 0, -8, 3, 2, 3} */
```

Entrées/Sorties Digitales

```
pinMode(pin, [INPUT,OUTPUT]) /*Définition d'une entrée/sortie digitale */
Exemple:
  pinMode(2,INPUT); //La pin 2 sera une entrée

digitalWrite(pin,[HIGH,LOW,0,1]) /*Défintion d'une valeur en sortie d'une pin digital spécifiée (HIGH ou 1 pour niveau haut et LOW ou 0 pour niveau bas) */
Exemple:
  digitalWrite(2,HIGH);//= digitalWrite(2,1);
  //pin 2 mise à l'état haut

digitalRead(pin) /*Permet d'obtenir le niveau digital de la pin spécifiée */
Exemple:
  int resultat;
  resultat = digitalRead(2);
  //resultat = le niveau de la pin 2 (0 ou 1)
```

Entrées/Sorties Analogiques

```
analogReference([DEFAULT,INTERNAL,EXTERNAL])
/*Définir la tension de référence pour le convertisseur analogique numérique.
DEFAULT = 5V / INTERNAL = 1.1V / EXTERNAL = AREF (0 à 5V sur pin de référence)
Valable uniquement pour Arduino UNO et Nano voir la documentation en ligne pour les autres cartes */
Exemple:
  analogReference(DEFAULT);/*Tension de référence à 5V par défaut */

analogRead(ADCpin) /*Lecture analogique de la pin spécifiée. Résultat entre 0 et 1023 */
Exemple:
  int resultat;
  resultat = analogRead(2);
  /*resultat = le niveau de la pin 2
   soit une valeur entre 0 et 1023 */

analogWrite(pin, valeur) /*Ecriture analogique à la pin spécifiée sous forme de PWM pour les pins suivantes uniquement : 3, 5, 6, 9, 10, 11
Valable pour Arduino UNO, Nano et Mini */
Exemple:
  analogWrite(3,512);
  /*écriture sur la pin 3 d'une valeur PWM à
  512 soit environ 2.5V en moyenne sous forme
  d'un signal carré ayant un rapport cyclique
  de 0.5 (512 = 1024/2) et une fréquence de
  490Hz (980Hz sur pin 5 et 6) */
```

Pulsations / Fréquences

```
tone(pin, fréquence) / tone(pin, fréquence, durée)
/*Permet de générer une fréquence sur une pin sous forme d'un signal carré de rapport cyclique 1/2 fréquence de 31 à 65535Hz, la durée d'activation s'exprime en ms. Plusieurs fonction tone ne peuvent être appelées en même temps. L'utilisation de la fonction interfèrera avec l'usage d'un PWM sur les pin 9 et 11 (Arduino UNO, Nano)*/
noTone(pin) /*Permet de stopper la fonction tone si aucune durée n'était spécifiée (voir fonction tone() )*/
pulseIn(pin, valeur)/pulseIn(pin, valeur, tempsmort)
/*Permet d'utiliser une pin pour compter des impulsions bas vers haut ou haut vers bas suivant la valeur de référence spécifiée. Une valeur de temps mort peut être ajoutée pour forcer le comptage sans nouvelle impulsion */
Exemple:
  unsigned long compte;
  compte = pulseIn(7,LOW); /*compte = décompte
  d'impulsions sur un front descendant */
```

Interruptions

```
attachInterrupt(digitalPinToInterrupt(pin),fonction,mode)
/*Permet d'utiliser une pin comme interruption pour démarrer une fonction. digitalPinToInterrupt(pin) permet de se référer à la bonne référence d'interruption suivant la pin renseignée. Seules les pin 2 et 3 sont utilisables sur Arduino Uno ou Nano.
"fonction" est la fonction à jouer dans le code lors de l'interruption.
"mode" détermine le mode d'interruption suivant la liste suivante:
 LOW quand la pin est à l'état bas,
 CHANGE quand la pin change de valeur (0->1 ou 1->0)
 RISING quand la pin passe au niveau haut 0->1
 FALLING quand la pin passe au niveau bas 1->0
 */
Exemple:
 attachInterrupt(digitalPinToInterrupt(2),blink,CHANGE);
  /*Ici la fonction attribura la pin 2 pour l'interruption en démarrant la fonction "blink" lors de n'importe quel changement d'état (0->1 ou 1->0) puisque le mode est CHANGE*/

detachInterrupt(digitalPinToInterrupt(pin))
/*Permet d'arrêter l'interruption (voir fonction précédente)*/
```

Gestion du Temps

```
millis() /*Donne le nombre de millisecondes écoulées depuis le début du programme. Attention, utiliser un type unsigned long pour stocker la valeur */

micros() /*Donne le nombre de microsecondes écoulées depuis le début du programme. Attention, utiliser un type unsigned long pour stocker la valeur */

delay(valeur_ms) /*Permet de faire une pause dans le programme d'une durée en ms spécifiée */

delayMicroseconds(valeur_us) /*Permet de faire une pause dans le programme d'une durée en us spécifiée */
```

EEPROM

```
#include <EEPROM.h> /*Librairie à inclure en amont pour utiliser les fonctions suivantes */

EEPROM.read(adresse) /*Permet de lire un octet à l'adresse spécifiée. Arduino UNO et Nano = 1KB d'EEPROM disponible */
Exemple:
  int adresse = 0;
  byte valeur;
  valeur = EEPROM.read(adresse);
  /*valeur = l'octet stocké à l'adresse spécifiée */

EEPROM.write(adresse,valeur) /*Permet d'écrire un octet à l'adresse spécifiée */
Exemple:
  int adresse = 0;
  byte valeur = 0xFF;
  EEPROM.write(adresse, valeur);
  /*Ecrit l'octet 0xFF à l'adresse spécifiée */
```

	ATMega168	ATMega328	ATMega1280	ATMega2560
EEPROM	512 B	1 KB	4 KB	4 KB

Fonctions Mathématiques

```
min(valeur1, valeur2) /*Renvoi la plus petite des 2 valeurs */
Exemple:
  int resultat;
  resultat = min(25,50); //resultat = 25

max(valeur1, valeur2) /*Renvoi la plus grande des 2 valeurs */
Exemple:
  int resultat;
  resultat = max(25,50); //resultat = 50

abs(valeur) //Renvoi la valeur absolue d'un nombre
Exemple:
  int resultat;
  resultat = abs(-50); //resultat = 50

constrain(valeur, mini, Maxi) /*Permet d'encadrer une valeur entre un minimum et un maximum */
Exemple:
  int resultat = 25;
  resultat = constrain(resultat,20,50);
  //resultat = 25
  resultat = constrain(resultat,30,50);
  //resultat = 30

map(valeur, min1, max1, min2, max2);
/*Effectue une conversion linéaire de la valeur entre la gamme 1 (min1 et max1) et la gamme 2 (min2 et max2). Equivaut à: valeur((max2-min2)/(max1-min1)) */
Exemple:
  int resultat;
  resultat = map(2,0,10,0,100);
  //resultat = 20 = 2(100/10)

pow(base, exponent) /*Calcul de puissance d'une valeur */
Exemple:
  int resultat;
  resultat = pow(2,3); //resultat = 8 = 2^3

sqrt(valeur) //Calcul racine carrée d'une valeur
Exemple:
  int resultat;
  resultat = sqrt(4); //resultat = 2

sq(valeur) //Calcul du carré d'une valeur
Exemple:
  int resultat;
  resultat = sq(4); //resultat = 16

sin(valeur)/*Calcul du sinus d'une valeur en radian */

cos(valeur)/*Calcul du cosinus d'une valeur en radian */

tan(valeur)/*Calcul de la tangente d'une valeur en radian */

random(maxi) / random(mini,maxi) /*Permet de générer un nombre aléatoire en définissant des paramètres ou non. Gamme en fonction du type déclaré: int, long, float... */
Exemple:
  int resultat;
  resultat = random(); /*Génère un chiffre dans la gamme d'un entier int -> -32768 à 32767 */
  resultat = random(50);/*Génère un chiffre dans la gamme d'un entier avec un maximum à 50 int -> -32768 à 50 */
  resultat = random(50);/*Génère un chiffre dans la gamme d'un entier avec un maximum à 50 int -> -32768 à 50 */
```

Fonctions Chaînes de caractères

```
length() /*Retourne la longueur d'une chaîne de caractère */
Exemple:
  String str = "Arduino";
  int resultat = 0;
  resultat = str.length(); //Le résultat sera 7

+= //Permet de concaténer des chaînes de caractères
Exemple:
  String str1 = "Ardui";
  String str2 = "no";
  str1 += str2; //str1 = "Ardui" + "no" = "Arduino"
  char chr = '!';
  str1 += chr;  //str1 = "Arduino" + '!' = "Arduino!"
  /*Il est donc possible d'ajouter des caractères seuls à un String */

substring(de, jusquà) /*Permet d'extraire une portion de chaîne de caractères de l'index spécifié jusqu'à un autre index spécifié. La première lettre est à l'index 0 */
Exemple:
  String str1 = "Arduino";
  String resultat = "";
  resultat = str1.substring(2);  //resultat = "duino"
  resultat = str1.substring(2,4);//resultat = "dui"

toUpperCase() //Passe tout en majuscule
Exemple:
   String str1 = "Arduino";
   str1.toUpperCase(); //str1 = "ARDUINO"

toLowerCase() //Passe tout en minuscule
Exemple:
   String str1 = "ARDUINO";
   str1.toLowerCase(); //str1 = "arduino"

indexOf() /*Donne le 1er index de localisation d'un caractère */
Exemple:
  String str1 = "<HTML><HEAD><BODY>";
  int resultat = str1.indexOf('>'); /*resultat = 5 soit le premier index où le caractère > a été trouvé dans la chaîne */

lastIndexOf() /*Donne le dernier index de localisation d'un caractère */
Exemple:
  String str1 = "<HTML><HEAD><BODY>";
  int resultat = str1.lastIndexOf('<'); /*resultat = 12 soit le dernier index où le caractère < a été trouvé dans la chaîne */

replace() /*Permet de remplacer une chaîne existante par une autre */
Exemple:
  String str1 = "<html><head><body>";
  str1.replace("<", "</");
  // str1 = "</html></head></body>"
```

Liaison Série 1/2

```
Serial.begin(débit)
/*Permet de démarrer la liaison série native de l'arduino. Le débit s'exprime en bauds soit des bits par secondes suivant la liste des valeurs suivante: 300,1200,2400,4800,9600,14400,19200,28800,38400,57600, 115200.
La valeur renseignée dépend du débit de l'équipement raccordé à la liaison série. Cette fonction est à appeler durant le setup du programme (de préférence).*/

Serial.available()
/* Retourne le nombre d'octet reçus et en attente dans le buffer. Permet avec une condition de détecter les nouveaux octets reçu pour lecture.*/
Exemple:
  int octet_entrant = 0; //octet entrants
  void setup() {
    Serial.begin(9600); /*Démarrage de la liason série à 9600bps */
  }
  void loop() {
    if (Serial.available() > 0) { /*Si des octets sont présents */
      octet_entrant = Serial.read(); /*Lecture du 1er octet */
    }
  }

Serial.read() /*Permet d'extraire le 1er octet du buffer de la liaison série. Voir exemple précédent pour utilisation */

Serial.readBytes(buf, longueur) /*Permet de lire plusieurs octets présents dans le buffer de la liaison série */

Serial.readBytesUntil(caractère, buf, longueur) /*Permet de lire plusieurs octets jusqu'à la détection d'un caractère spécifié*/

Serial.readString() /*Permet de lire les octets directement sous forme d'une chaîne de caractères et non en octets brutes */

Serial.readStringUntil(caractère) /*Permet de lire les octets directement sous forme d'une chaîne de caractères et non en octets brutes jusqu'à un caractère spécifié */

Serial.peek() /*Permet de lire sans extraire le 1er octet du buffer de la liaison série */


Serial.write(val)
Serial.write(str)
Serial.write(buf, len)
/*Permet d'écrire une valeur, une chaîne de caractère ou un tableau d'octet sur la liaison série.
val est une valeur sous forme d'octet (ou byte)
str est une chaîne de caractères
buf est un tableau d'octets
len est la longueur du tableau d'octets */
Exemple:
  void setup() {
      Serial.begin(9600);
    }
    void loop() {
      Serial.write(45); //Envoi la valeur 45 sous forme d'octet
      int bytesSent = Serial.write("hello");/*Envoi la chaîne de caractère "hello" */
    }
```

Liaison Série 2/2

```
Serial.print(valeur, format) /*Permet d'écrire une valeur convertie en chaîne de caractères automatiquement sur la liaison série et suivant le format optionnel spécifié (BIN pour binaire,HEX pour hexadécimal, DEC pour décimal, OCT pour octal et une valeur décimale pour définir un nombre de chiffres après la virgule pour un nombre à virgule (float))*/
Exemple:
void setup() {
  Serial.begin(9600); //Démarrage de la liaison série
}
void loop() {
  Serial.print("hello");//Chaîne de caractère
  Serial.print(10,DEC); //10 en chaîne de caractères décimale
  Serial.print(10,HEX); //10 en héxadécimal = "A"
  Serial.print(10,OCT); //10 en octal = "12"
  Serial.print(1.234,2);//1.234 avec 2 chiffres significatifs "1.23"
}

Serial.println(valeur, format) /*Identique à la fonction précédente mais inclus un retour à la ligne à la fin de l'envoi '\n' */

Serial.end() /*Permet d'arrêter la liaison série */
Serial.flush()/*Permet de vider le buffer sortant */
```

Servomoteurs

```
#include <Servo.h> /*Librairie à inclure en amont pour utiliser les fonctions suivantes */
Servo.attach(pin) / Servo.attach(pin, min, max)
/*Permet d'attribuer une sortie de servomoteur sur une pin spécifiée en précisant les valeurs min et max des pas */
Exemple:
  #include <Servo.h>
  Servo monservo; /*Création de l'objet monservo de type Servo */
  void setup()
  {
  monservo.attach(9); /*Attribution de la pin 9 pour ce servo */
  }

Servo.detach(pin) /*Fonction inverse de la précédente permettant de supprimer l'attribution du servo à la spécifiée */

Servo.attached() /*Permet de vérifier si une pin est attribuée à la référence du servo. Retourne une valeur vrai ou fausse(true ou false) */

Servo.write(angle) /*Permet d'écrire un angle de pilotage pour le servomoteur attribué. L'angle s'exprime en degré de 0 à 180° */
Exemple:
  #include <Servo.h>
  Servo monservo; /*Création de l'objet monservo de type Servo*/
  void setup()
  {
    monservo.attach(9);  //Attribution pin 9
    monservo.write(90);  //Angle spécifié à 90°
  }

Servo.writeMicroseconds(uS) /*Permet d'écrire directement la valeur en microsecondes de l'impulsion à envoyer au servomoteur. Pour les servo standards envoyer entre 1000 et 2000us (0 et 180°)*/

Servo.read() /*Permet d'obtenir la valeur actuelle de l'angle appliqué au servomoteur */
```

Fonctions Binaires

```
lowByte() /*A utiliser pour extraire l'octet le plus à droite d'un binaire*/
Exemple:
   byte low;
   int nombre = 61610; /*61610 = B 11110000 10101010 sur 16 bits */
   low = lowByte(nombre); //low = 10101010

highByte() /*A utiliser pour extraire le second octet d'un binaire > 1 octet */
Exemple:
   byte high;
   int nombre = 61610; /*61610 = B 11110000 10101010 sur 16 bits */
   high = highByte(nombre); //high = 11110000

bitRead() /*Permet de lire un bit spécifique dans un octet */
Exemple:
   byte Byte = B00000001;
   bool resultat = 0;
   resultat = bitRead(Byte,0);
   //Résultat = 1 , 0 est le bit le plus à droite
   resultat = bitRead(Byte,7);
   //Résultat = 0 , 7 est le bit le plus à gauche

bitWrite() /*Permet d'écrire un bit spécifique dans un octet */
Exemple:
   byte Byte = B00000000;
   Byte = bitWrite(Byte,0,1); /*Byte = B00000001
   0 est le bit le plus à droite et 1 la valeur souhaitée */
   Byte = bitWrite(Byte,7,1); /*Byte = B10000001
   7 est le bit le plus à gauche et 1 la valeur souhaitée */

bitSet() /*Permet de mettre à 1 un bit spécifique dans un octet */
Exemple:
   byte Byte = B00000000;
   Byte = bitSet(Byte,0); /*Byte = B00000001
   0 est le bit le plus à droite */
   Byte = bitSet(Byte,7); /*Byte = B10000001
   7 est le bit le plus à gauche */

bitClear() //Permet de mettre à 0 un bit spécifique dans un octet
Exemple:
   byte Byte = B11111111;
   Byte = bitClear(Byte,0); /*Byte = B11111110
   0 est le bit le plus à droite */
   Byte = bitClear(Byte,7); /*Byte = B01111110
   7 est le bit le plus à gauche */

bit() //Permet de calculer la puissance de 2 relative à un bit
Exemple:
   int resultat = 0;
   resultat = bit(0); //Résultat = 1 soit 2^0
   resultat = bit(1); //Résultat = 2 soit 2^1
   resultat = bit(2); //Résultat = 4 soit 2^2 etc...
```

SoftwareSerial

```
#include<SoftwareSerial.h> /*Librairie à inclure en amont pour utiliser les fonctions suivantes */
SoftwareSerial nom_objet = SoftwareSerial(RXpin, TXpin);
/*SoftwareSerial est une classe, il faudra créer un objet de type SoftwareSerial comme ci-dessus en amont du code pour pouvoir l'utiliser */

SoftwareSerial.begin(9600) /*Permet de démarrer la liaison série suivant le débit spécifié (identique à la fonction Serial classique ) */
Exemple:
#include <SoftwareSerial.h>
// Création de l'objet de type SoftwareSerial
SoftwareSerial mySerial =  SoftwareSerial(10, 11);
void setup()  {
  //Définir pin 10 en entrée RX et 11 en sortie TX
  pinMode(10, INPUT);
  pinMode(11, OUTPUT);
  mySerial.begin(9600); //Démarrage de la liaison
}
void loop() {} //Boucle vide

SoftwareSerial.available()
/* Retourne le nombre d'octet reçus et en attente dans le buffer. Permet avec une condition de détecter les nouveaux octets reçu pour lecture. Fonctionne comme pour la fonction Serial classique */

SoftwareSerial.read() /*Permet d'extraire le 1er octet du buffer de la liaison série. Fonctionne comme pour la fonction Serial classique*/

SoftwareSerial.peek() /*Permet de lire sans extraire le 1er octet du buffer de la liaison série. Fonctionne comme pour la fonction Serial classique */

SoftwareSerial.write(val)
/*Permet d'écrire une valeur, une chaîne de caractère ou un tableau d'octet sur la liaison série. Fonctionne comme pour la fonction Serial classique*/

SoftwareSerial.print(valeur, format) /*Permet d'écrire une valeur convertie en chaîne de caractères automatiquement sur la liaison série software et suivant le format optionnel spécifié (BIN pour binaire,HEX pour hexadécimal, DEC pour décimal, OCT pour octal et une valeur décimale pour définir un nombre de chiffres après la virgule pour un nombre à virgule (float)).Fonctionne comme pour la fonction Serial classique*/

SoftwareSerial.println(valeur, format) /*Identique à la fonction précédente mais inclus un retour à la ligne à la fin de l'envoi '\n' */

SoftwareSerial.isListening() /*Permet de tester si le port série est actif ou non. Le résultat sera un booléen vrai ou faux (true ou false) */

SoftwareSerial.listen() /*Permet d'activer le port série si il avait été désactivé. Un seul port software ne peut être actif à la fois, attention donc si vous souhaitez en utiliser plusieurs*/

SoftwareSerial.end() /*Permet de fermer le port série software. Fonctionne comme pour la fonction Serial classique*/
```

La plupart des fonctions classiques Serial sont compatibles avec la fonction SoftwareSerial, pensez à regarder la documentation de la fonction Serial classique!

Memento Arduino

MONKEY MAKER
www.monkey-maker.com

Bases | Hardware | Fonctions

ARDUINO

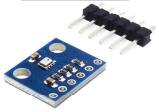
Wire

```
#include<Wire.h> /*Librairie à inclure en amont pour utiliser les fonctions suivantes */

Wire.begin(adresse)/*Démarrage du bus en mode master si aucune adresse n'est spécifiée (Master, sinon en mode Slave (esclave)*/

Wire.requestFrom(adresse, quantité)
Wire.requestFrom(adresse, quantité, stop)
/*Requête du maître à l'adresse indiquée et pour la quantité d'octets spécifiés. stop en option pour libérer le bus après réception (true ou false) */

Wire.beginTransmission(adresse)  /*Démarrage d'une transmission à l'adresse définie */
Wire.endTransmission() /*Termine la transmission avec le périphérique vers lequel une transmission avait démarrée */

Wire.write(valeur)/.write(str)/.write(data,taille)
/*Permet d'écrire un octet, une chaîne de caractère ou un paquet d'octets suivant la taille spécifiée */

Wire.available()  /*Indique si des données sont disponibles pour lecture après une requête sur un périphérique esclave. Retourne le nombre d'octets disponibles en lecture */

Wire.read() /*Permet d'extraire le 1er octet sur le bus de communication */

Wire.setClock(fréquence_bus)  /*Permet de définir la fréquence de communication en Hz du bus entre 100000 (mode standard) et 400000 (mode rapide)*/

Wire.onRequest(fonction)  /*Permet de jouer une fonction à la réception d'une requête pour le cas d'un périphérique esclave */

Wire.onReceive(fonction)/*Permet de jouer une fonction à la réception de données d'un périphérique maître pour le cas d'un périphérique esclave */
```

Modules

MPU6050

Accéléromètre 3 axes numérique

BME280

Capteur pression, température, hygrométrie numérique

DHT11

Capteur température, hygrométrie numérique

DS18B20

Capteur température numérique

VL53L0X

Capteur de distance infrarouge numérique

ModuleIR

Détecteur infrarouge analogique

ACS712

Capteur courant analogique

INA219

Capteur courant numérique

ADS1115

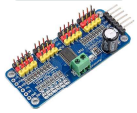
Convertisseur analogique numérique

MCP23017

Multiplexeur numérique

Relais

Commande relais

PCA9685

Commande 16 PWM (servo ou LED)

AT24C

Module EEPROM externe

MicroSD

Extension micro SD

RTC

Horloge temps réel

HD44780

Ecran LCD

WS2812(B)

LEDs RGB adressables

HC-05/06

Modules bluetooth

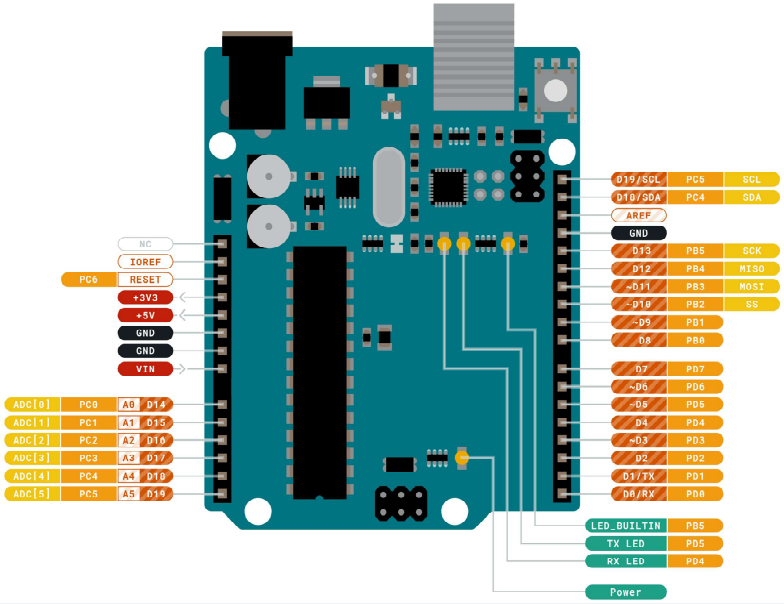
ESP8266

Modules WIFI

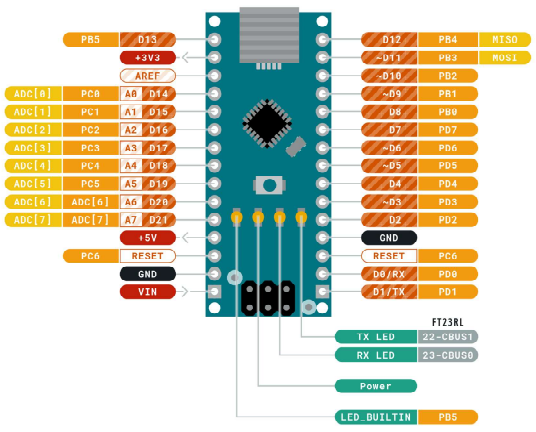
HC-012

Module radio 433MHz 1km

UNO REV3



NANO REV3



Ground | Internal Pin | Digital Pin | Microcontroller's Port
Power | SWD Pin | Analog Pin
LED | Other Pin | Default