La nature de l'information

Table des matières

Les progrès technologiques ont dopé la communication, par la rotative et le chemin de fer au XIXe siècle, puis les ondes hertziennes, le satellite et l'Internet. L'information est immatérielle. Elle peut être consignée directement ou pas sur un support matériel qui prend alors la valeur de document. L'information toutefois est indépendante du support : elle existe indépendamment de lui.

1. La domotique

La domotique est l'ensemble des techniques de l'électronique, de physique du bâtiment, d'automatisme, de l'informatique et des télécommunications utilisées dans les bâtiments pour répondre aux besoins de confort (gestion d'énergie, optimisation de l'éclairage et du chauffage), de sécurité (alarme) et de communication (commandes à distance, signaux visuels ou sonores).

figure 1.

1.1. Projet domotique

Objectif : développer une interface web d'IHM[1] d'un module domotique.

Cahier des charges : cette interface doit permettre à l'utilisateur de

- d'afficher les états ou les mesures temps réel des capteurs
- de stocker les données des capteurs (état et mesure)
- d'afficher graphiquement ces données sur une période journalière et hebdomadaire
- de forcer la commande des actionneurs en mode manuel

1 Interaction Homme Machine

INTERFACE :

C'est un système de traduction des informations de la chaîne d'information vers la chaîne d'énergie (ordres) et de la chaîne d'énergie vers la chaîne d'information (comptes rendus).

Pour obtenir visualiser un exemple d'IHM domotique, cliquer sur l'image

1.2. Exemple

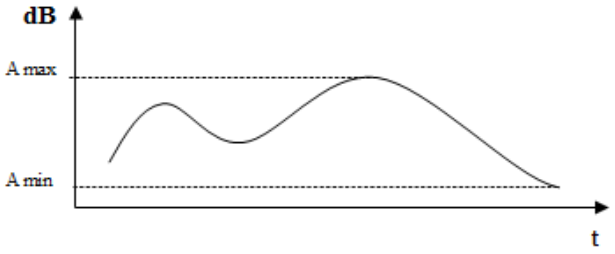
Réf : http://www.selectronic.fr/c/kits-modules/module-domotique.html au 6/09/14

2. Nature de l'information

2.1. Représentation

Dans un système de traitement, l'information peut être :

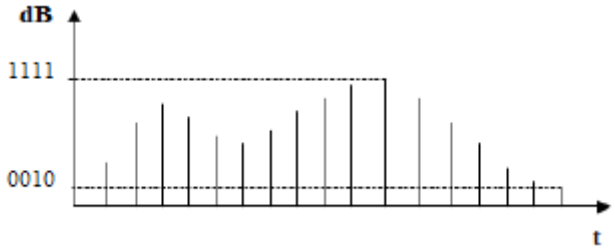
- Analogique (analog) : le signal varie de manière continue au cours du temps (mesure d'une grandeur physique). Son amplitude peut prendre une infinité de valeurs dans un intervalle de temps donné.



ex : mesure de l'intensité lumineuse en fonction du temps

Une grandeur analogique varie de façon continue proportionnellement à l'indication donnée par un capteur (ie :la position de l'aiguille d'un voltmètre est proportionnelle à la valeur de la tension mesurée mais la lecture peut être imprécise).

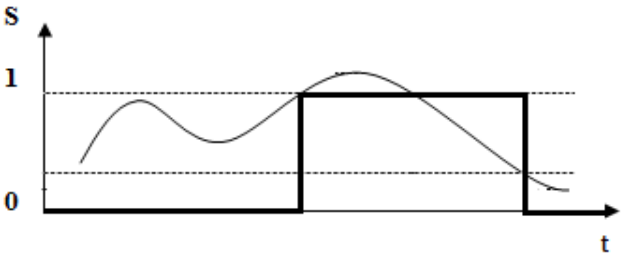
- Numérique (digital) : la représentation du signal varie de façon discrète (ie : discontinue) dans une liste de valeurs.



ex : échantillonnage d'une source lumineuse

Une grandeur numérique varie de façon discontinue et non proportionnelle à l'indication donnée par un capteur (ie : la valeur de la tension mesurée avec un voltmètre numérique est exprimée au moyen de un ou plusieurs chiffres dont la lecture est précise).

- Logique (logic) : le signal est convertit dans un état binaire qui ne prend que deux valeurs, notées par convention 0 et 1 (logique Tout ou Rien, TOR).



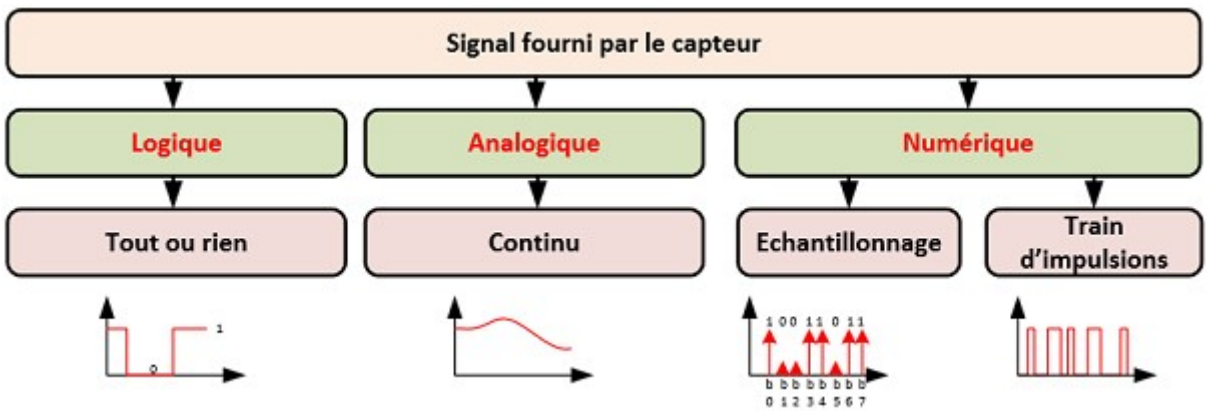
ex : activation d'un moteur en fonction du niveau d'ensoleillement

2.2. Exemple

À partir du schéma de la maison domotique, compléter le tableau ci-dessous :

Grandeur d'entrée	Module domotique	Signal de sortie
	Détecteur de présence	
	Gestion du chauffage	
	Détecteur de fumée	
	Mesure de luminosité	

Le schéma ci-dessous résume les différents signaux électriques normalisés selon l'information à transmettre .



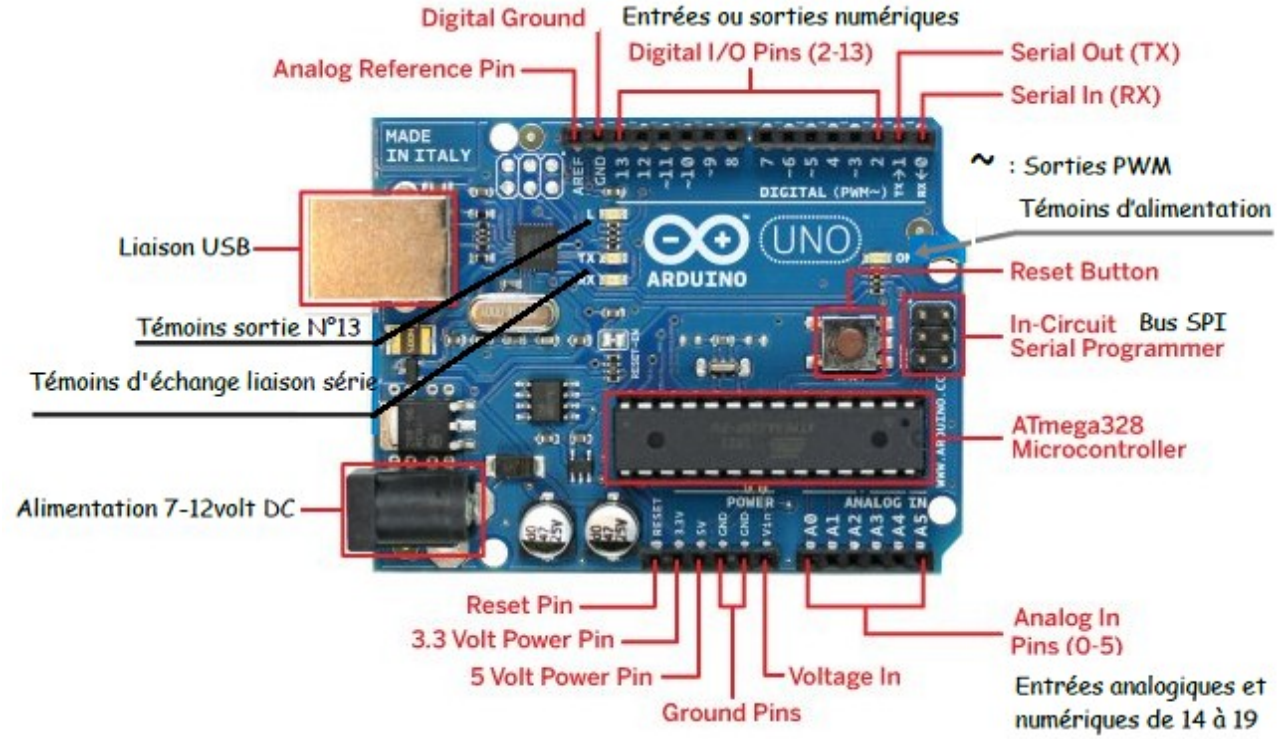
2.3. Caractérisation

On caractérise un signal par :

- sa forme, qui peut être quelconque, sinusoïdale, triangulaire ou rectangulaire.
- sa périodicité. Dans ce cas il peut être soit périodique, soit non-périodique. Si il est périodique on calcule alors sa période[2] T en secondes (s) ou sa fréquence[3] f en Hertz (Hz). $T=\frac{1}{f}$ ou $f=\frac{1}{T}$
- son amplitude. Elle représente la différence entre la valeur maximale et la valeur minimale (qui peut être négative).
- sa valeur moyenne si il est périodique.

3. Arduino

Arduino est un nano automate qui peut être utilisé pour lire et piloter des entrées / sorties analogiques ou numériques ou bien communiquer avec un ordinateur.



3.1. Caractéristiques de l'arduino UNO

Microcontrôleur	ATmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V

2 Durée d'un cycle
3 Nombre de cycles par seconde

Tension d'alimentation (limites)	6-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrées analogiques	6 (utilisables en broches E/S numériques)
Intensité maxi disponible par broche E/S (5V)	40 mA (ATTENTION : 200mA cumulé pour l'ensemble des broches E/S)
Intensité maxi disponible pour la sortie 3.3V	50 mA
Intensité maxi disponible pour la sortie 5V	Fonction de l'alimentation utilisée - 500 mA max si port USB utilisé seul
Mémoire Programme Flash	32 KB (ATmega328) dont **0.5 KB** sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	2 KB (ATmega328)
Mémoire EEPROM (mémoire non volatile)	1 KB (ATmega328)
Vitesse d'horloge	16 MHz

3.2. Entrées et sorties numériques

Chacune des 14 broches numériques de la carte UNO (numérotées des 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode(), digitalWrite() et digitalRead() du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digitalWrite(broche, HIGH).

De plus, certaines broches ont des fonctions spécialisées :

- **Communication Série :**Broches 0 (RX) et 1 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données du port USB de l'ordinateur.
- **Impulsion PWM (largeur d'impulsion modulée)** : Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite().
- **LED :** Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

3.3. Broches analogiques

La carte Uno dispose de 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction analogReference() du langage Arduino.

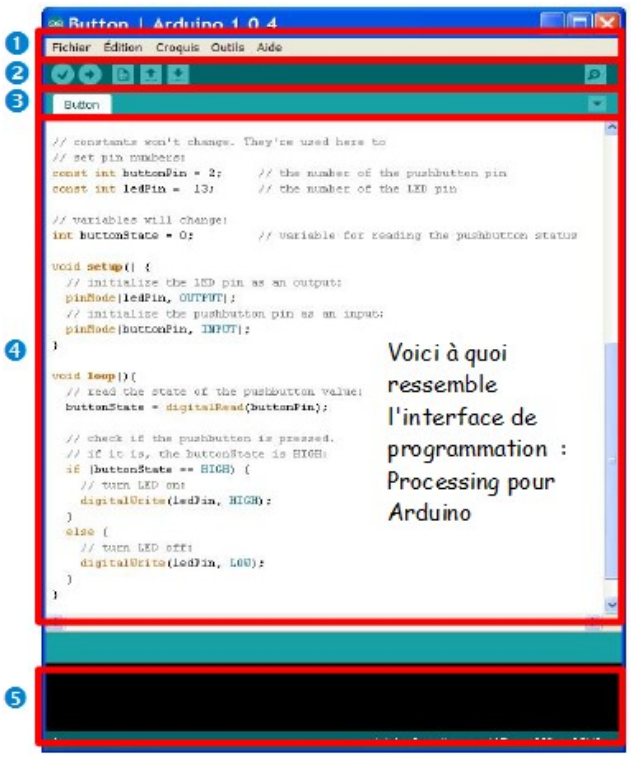
Note : les broches analogiques peuvent être utilisées en tant que broches numériques : elles sont numérotées en tant que broches numériques de 14 à 19.

3.4. Autres broches

Il y a deux autres broches disponibles sur la carte :

- **AREF** : Tension de référence pour les entrées analogiques (si différent du 5V). Utilisée avec l'instruction analogReference().
- **Reset** : Mettre cette broche au niveau BAS entraîne la réinitialisation (= le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.

3.5. Découverte de l'interface (standard)



Avant de commencer à programmer de manière graphique il nous faut comprendre la structure du langage de l'IDE

❶un menu

❷une barre d'actions

❸un ou plusieurs onglets correspondant aux "sketchs"

❹une fenêtre de programmation

❺une console qui affiche les informations et erreurs de compilation et de téléversement du programme

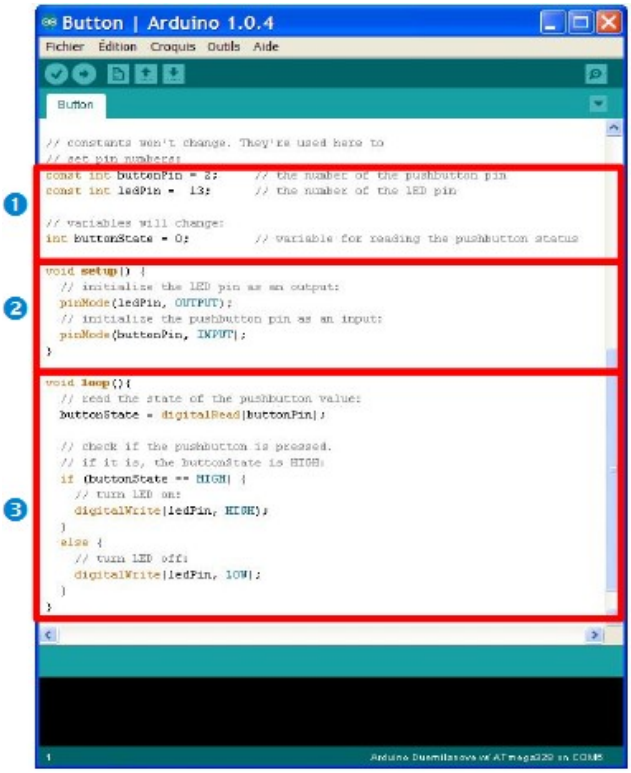
Coloration syntaxique :

Lorsque du code est écrit dans l'interface de programmation, certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

En **orange**, apparaissent les mots-clés reconnus par le langage Arduino comme des fonctions existantes. Lorsqu'on sélectionne un mot coloré en orange et qu'on effectue un clic avec le bouton droit de la souris, on a la possibilité de choisir « Find in reference » : cette commande ouvre directement la documentation de la fonction sélectionnée.

En **bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des constantes.

En **gris**, apparaissent les commentaires qui ne seront pas exécutés dans le programme. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes.



On peut déclarer un commentaire de deux manières différentes :

//Commentaires (1 ligne)

/* zone de commentaire*/

pour éviter de se perdre dans un programme.

❶la partie déclarative, variables et constantes. Dans cette partie sont lancé les différentes bibliothèques qui peuvent simplifier la programmation. Cette partie est optionnelle.

❷la partie initialisation et configuration. Ici peuvent être déclarées des entrées/sorties, ou bien encore lancé la liaison série :

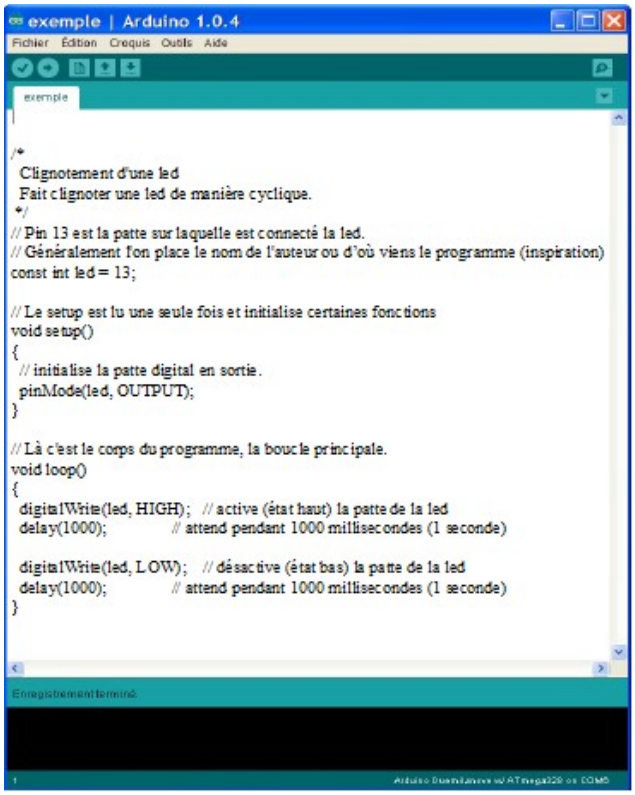
- C'est la fonction setup()

❸la partie principale qui s'exécute en boucle :

- C'est la fonction loop()

Dans chaque partie d'un programme sont utilisées différentes instructions issues de la syntaxe du langage Arduino.

3.5.1. Structure d'un programme Arduino



La ponctuation

Le code est structuré par une ponctuation stricte :

- **toute ligne** de code se termine par un point-virgule **;**
- le contenu d'une **fonction** est délimité par des accolades **{** et **}**
- les **paramètres** d'une fonction sont contenus par des parenthèses **(** et **)**

Les variables

Une variable est un espace réservé dans la mémoire de l'ordinateur. C'est comme un compartiment dont la taille n'est adéquate que pour un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement.

Il existe différents types de variables identifiées par un mot-clé dont les principaux sont :

- **nombres entiers (int)**
- **nombres à virgule flottante (float)**
- **texte (String et string)**
- **valeurs vrai/faux (boolean)**

Un nombre décimales, par exemple *3.14159*, peut se stocker dans une variable de type float. Notez que l'on utilise un point et non une virgule pour les nombres décimales.

Dans Arduino, il est nécessaire de déclarer les variables pour leur réserver un espace mémoire adéquat. On déclare une variable en spécifiant son type, son nom puis en lui assignant une valeur initiale

Exemple : int buttonPin = 2;

La variable s'appelle ButtonPin, elle fonctionne avec des nombres entiers et sa valeur par défaut est 2.

3.5.2. Variables disponibles dans le langage

Une variable est une "boîte" qui contient une donnée. Ces données peuvent être de plusieurs types :

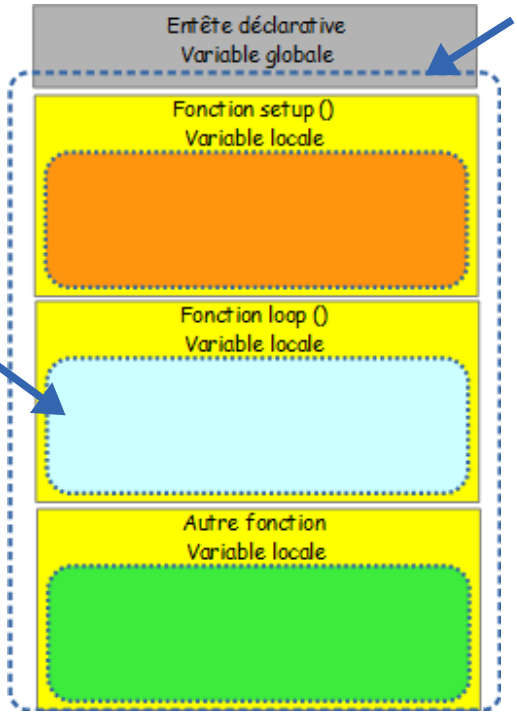
nom	caractéristiques	sans signe unsigned	exemples
Données numériques			

byte	0 à 255	Déclare une variable de type octet (8 bits) qui stocke un nombre entier non-signé	byte a=145 ; // la donnée a est égal à 145
			byte a= b10010 ; // la donnée est en binaire et elle est égale à 18 en DEC

int	-32536 à +32535	0 à +65535	int b=-2458 ; //la donnée b est égale à -2458
word	0 à +65535	représente int unsigned	word w = 10000 ;
long	-2147483648 à +2147483647	0 à +4294967295	long vitessemoteur = 186000L; //déclare une variable de type long
float	Nombre à virgule	$-3{,}4028235*10^{38}$ à $+3{,}4028235*10^{38}$	float capteurvitesse = 1.117; // déclare une variable à virgule appelée capteurvitesse
Données logiques			
boolean	0 ou 1, vrai ou faux true ou false occupe 1 octet		boolean marche = false ; // la variable marche est fausse
void			void setup () // la fonction setup ne fournit aucun résultat
Données caractères ou chaîne de caractères			
string	Chaîne de caractères évoluée	Les chaînes de caractères sont représentées sous forme de tableau de variables de type char et se termine par un zéro [ ]	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	Chaîne de caractères		
char	Tout caractères (code ASCII) -128 à +127	0 à +255 code ASCII étendu	char lettreA = 'A' // la donnée lettreA est égale au caractère A, soit 65 en décimal qui est le code ASCII de A

3.5.3.Comment et ou déclarer une variable

Locale : la variable est visible et utilisable que dans une certaine parties du programme



Globale : la variable est visible et utilisable depuis tout les parties du programme

Il est est également pratique de déclarer et d'initialiser une variable à l'intérieur d'une boucle for. Ceci crée une variable qui ne sera accessible uniquement à l'intérieur des accolades { } de la boucle for.

Il est préférable de déclarer les variable avant les parties setup et loop. Elles seront vus depuis n'importe quelle partie du programme.

3.6. Exemples de programmes

3.6.1. Lecture numérique

```
// Allumage LED par bouton

const int BP  = 2;     // broche 2 pour Bouton Poussoir
const int LED = 5;     // broche 3 pour LED

void setup()
{
      pinMode(BP, INPUT);      // BP en entrée
      pinMode(LED, OUTPUT);    // LED en sortie
}

void loop()
{
      if ( digitalRead(BP) == HIGH )       // capteur TOR
            digitalWrite(LED, HIGH);       // actionneur
      else
            digitalWrite(LED, LOW);        // éteindre
      delay(100); // attente 100 ms
}
```

3.6.2. Lecture analogique

```
// lecture de température

void setup()
{
      //initialisation vitesse liaison série à 9600 bauds
      Serial.begin(9600) ;
      pinMode(A0 , INPUT);     // broche A0 en entrée
```

```
}

void loop()
{
      int t = analogRead(A0) ;      // lecture entrée analogique

      Serial.println(t);            // affichage valeur numérique
      delay(100); // attente 100 ms
}
```

3.6.3. MLI[4]

```
// MLI

const int moteur  = 3;    // broche 3 pour commande moteur

void setup()
{
      pinMode(moteur, OUTPUT);      // moteur en sortie
}

void loop()
{
      for (int i(0) ; i < 256 ; i++) {    // accélération
            analogWrite(moteur, i) ;
            delay(50);  // attente 50 ms
            }
      for (int i(0) ; i < 256 ; i++) {    // décélération
            analogWrite(moteur, 255 - i) ;
            delay(50);  // attente 50 ms
            }
}
```

4 Modulation à Largeur d'Impulsions