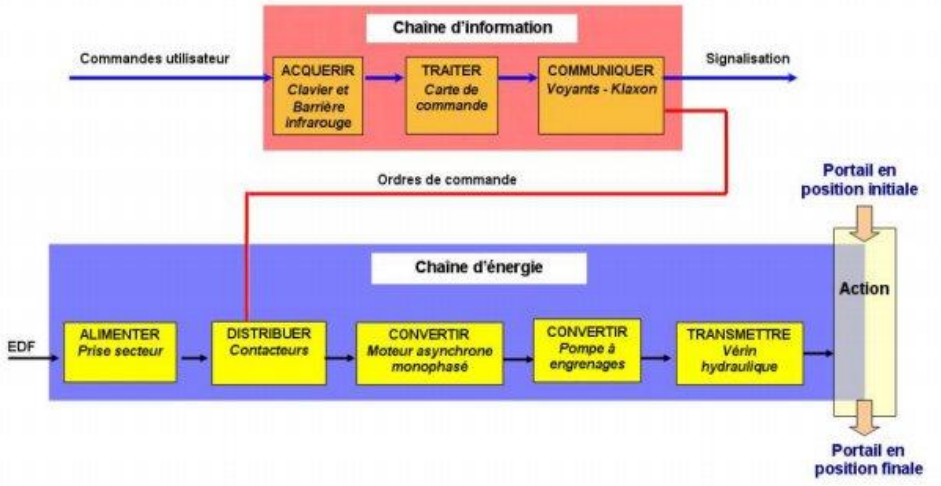
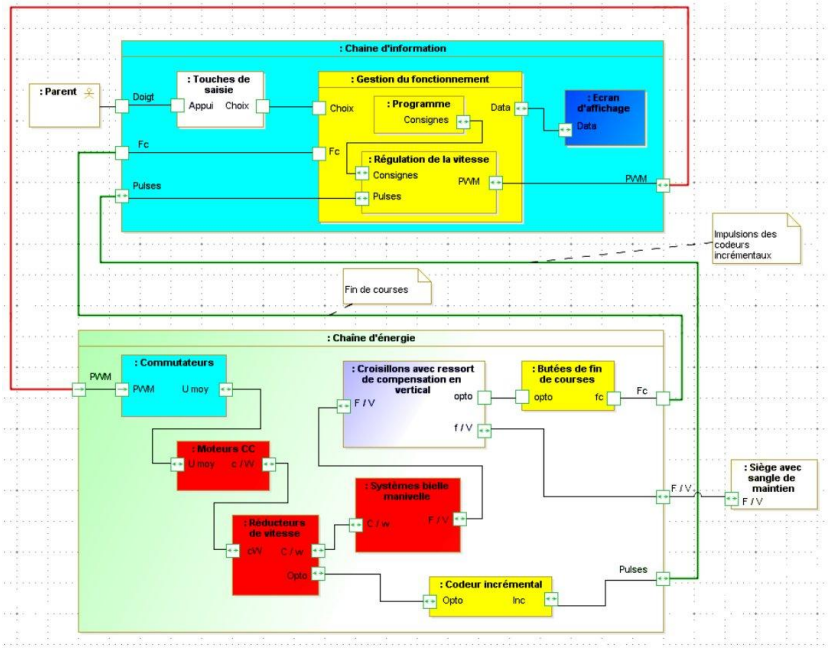
GENERALITES

Le microcontrôleur rassemble en un seul circuit les composants d'un petit ordinateur (processeur, mémoire) auquel on ajoute un nombre important d'entrée-sorties. Sa miniaturisation et son faible coût lui permet de remplacer la logique câblée maintenant obsolète, l'électronique de traitement analogique étant le plus souvent elle aussi réduite au minimum.

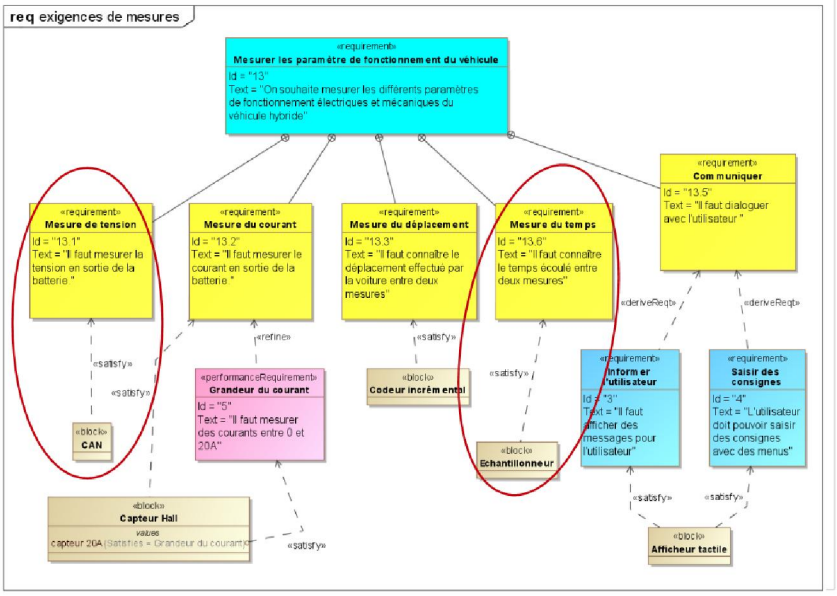
Dans la chaine d'information il assure le traitement de l'information



On peut aussi le retrouver en tant que bloc SysML (gestion du fonctionnement)

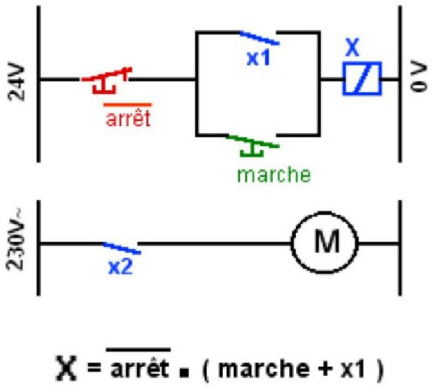


Il peut aussi tout simplement satisfaire quelques exigences par une de ses fonctions, ici la conversion analogique-numérique ou l'échantillonnage.



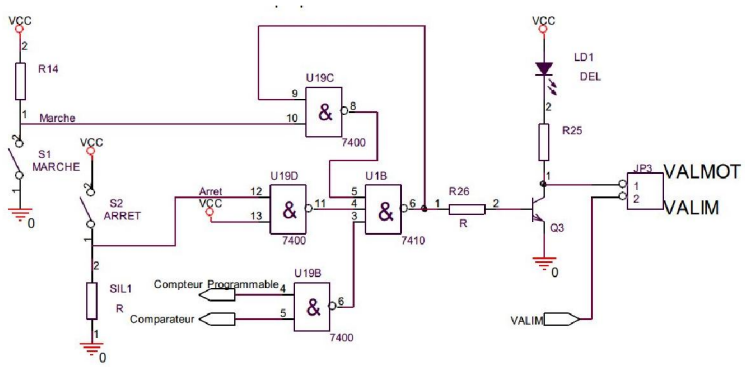
Il remplace bien souvent des technologies devenues obsolètes, on peut le voir ici avec la commande marche-arrêt d'un système.

Version à relais

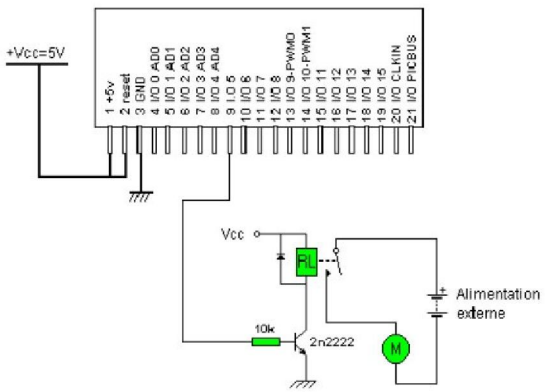


$$X = \overline{arrêt} \cdot (marche + x1)$$

Version Câblée



Version programmée (sans les boutons)



Le moteur est piloté par une sortie du microcontrôleur

C'est l'algorithme qui permet de traiter les consignes de marche et d'arrêt

REPETER sans fin
    SI appui sur le bouton marche
        ALORS mettre à 1 la sortie moteur
    FIN du SI

    SI appui sur le bouton arrêt
        Alors mettre à zéro la sortie moteur
    FIN du SI
FIN du REPETER

Programme correspondant

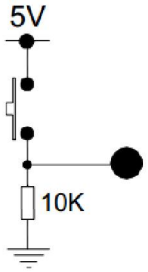
```
DEBUT:      ' début du programme

      IF IN(marche) = 1 THEN OUT moteur,1        ' mise en marche du moteur
      IF IN(arret) = 1 THEN OUT moteur,0         ' arret du moteur

GOTO DEBUT
```

Les entrées sorties des micro-contrôleurs.

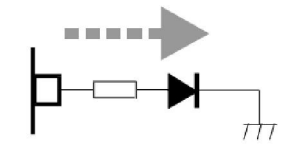
LES ENTREES SORTIES BINAIRES (TOR)

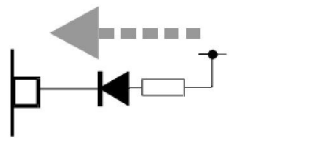
Une broche peut en général être configurée en entrée ou en sortie, le plus souvent avec des niveaux de tension de 0V pour l'état 0 et 5V pour l'état 1.

Câblage en entrée



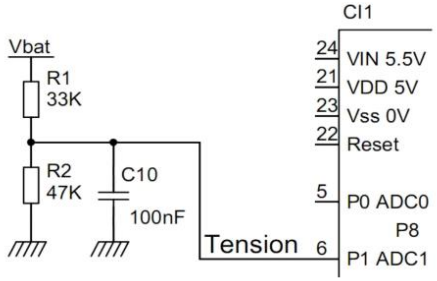
Câblage en sortie

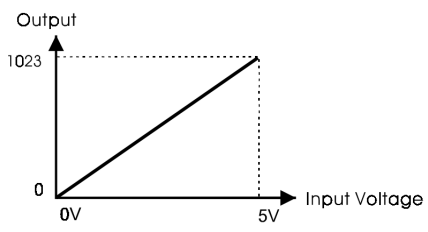




LES ENTREES ANALOGIQUES

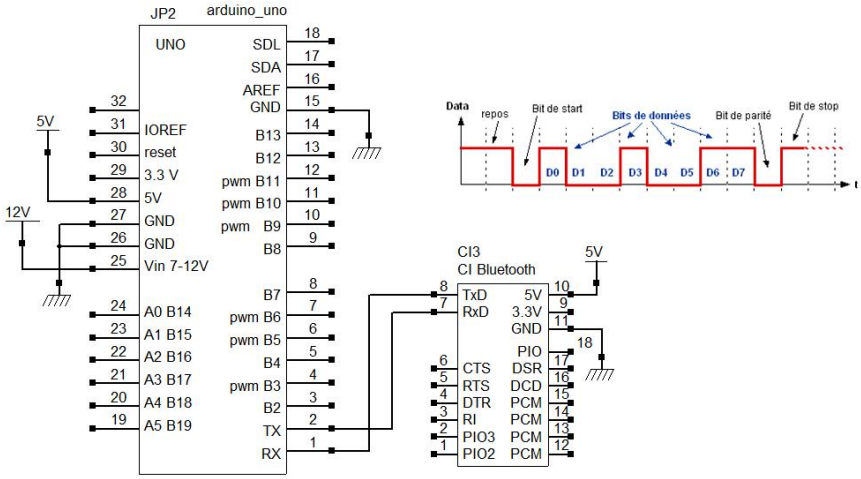
La tension mesurée en entrée est convertie en une valeur numérique sur 10 bits en général.



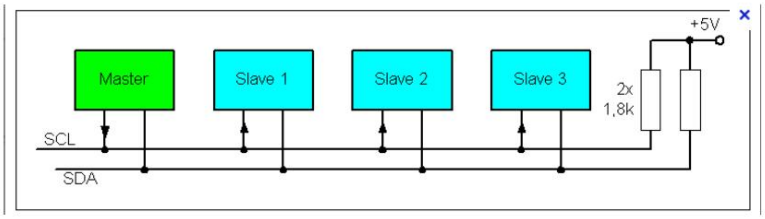


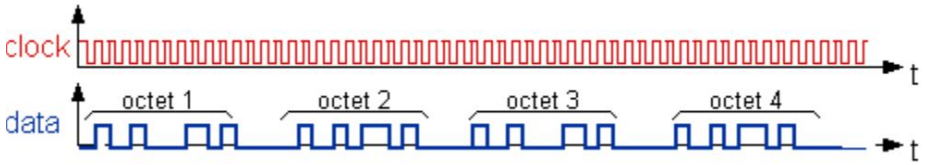
LES PORTS DE COMMUNICATION

On trouve en général un (ou plusieurs) port série pour échanger des données avec un autre circuit. Un seul circuit peut être connecté sur le port série, la sortie transmission de l'un est reliée sur la réception de l'autre et réciproquement.



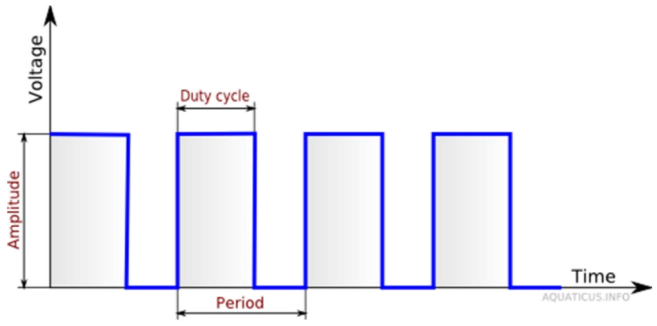
On trouve aussi un ou plusieurs bus de communication pour échanger des données entre circuits (I2C, SPI, CAN), le même bus peut communiquer avec plusieurs circuits, il**s** se distinguent par des adresses différentes.





LES SORTIES PWM

Elles permettent le plus souvent de commander des moteurs en vitesse, ce sont des sorties pseudo-analogiques.



LA PROGRAMMATION

Chaque microcontrôleur dispose de son propre langage plus ou moins standardisé

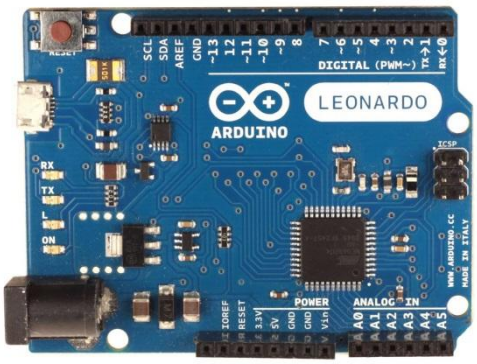
Exemple de programmation en basic Cubloc

```
Sub Creation_fichier()
   'controle de la liste des fichiers sur la carte
   Dim numero_fichier As Integer
   numero_fichier = 0
   lecture = ""
   'balayge des fichiers de mesures
   'attente d'une erreur sur la lecture de la taille du fichier
   'ce qui signifie qu'il n'existe pas donc on le cree
   Do Until(lecture = "E40")                  'message d'erreur
      'nom de fichier suivant
      Incr numero_fichier
      fichier = Dec(numero_fichier)
      fichier = "DATA" + fichier + ".XLS"
      'demande de la taille
      Putstr 1, "fsize " + fichier + "",13,10
      Delay 50
      'lecture du resultat
      nb_octets = Blen(1,0)
      lecture = Getstr(1,nb_octets)
   Loop
   'creation du prochain fichier de mesures qui n'existe pas
   Putstr 1, "fcreate " + fichier + "",13,10
   Delay 250
      'ouverture du fichier en ecriture
   Putstr 1, "fopen "+ fichier + " /a ",13,10
   Delay 50
   'ouverture d'un flux d'ecriture dans le fichier
   Putstr 1, "fputs2 " + " /a ",13,10
   Delay 50
End Sub
```

Exemple de programmation en C++ arduino

```
// Fonction de lecture d'un chaine
void lire_message(void)
{
  char car;
  int pos;
  pos = 0;
  car = 0;
  message[0] = '\0';
  while (car != '#')
  {
    if (Serial.available() > 0)
    {
      car = Serial.read();
      message[pos++] = car;
    }
  }
  message[pos - 1] = '\0';
}
```

La famille ARDUINO



Arduino est un circuit imprimé en matériel libre sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques de manière à effectuer des tâches très diverses comme la domotique (le contrôle des appareils domestiques - éclairage, chauffage…), le pilotage d'un robot, etc. C'est une plateforme basée sur une interface entrée/sortie simple.

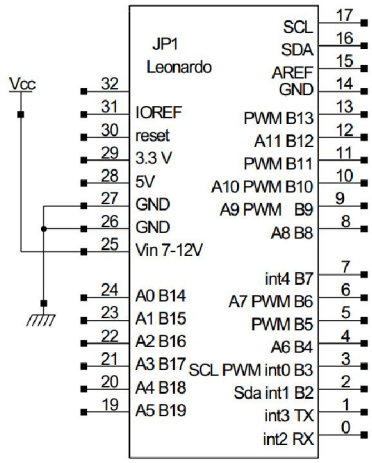
Un module Arduino est généralement construit autour d'un microcontrôleur Atmel AVR (ATmega328 ou ATmega2560 pour les versions récentes, ATmega168 ou ATmega8 pour les plus anciennes), et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède en général un régulateur linéaire 5 V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles).

Le microcontrôleur est préprogrammé avec un bootloader de façon à ce qu'un programmateur dédié ne soit pas nécessaire. Les modules sont programmés au travers d'une connexion USB-série. L'Arduino utilise

la plupart des entrées/sorties du microcontrôleur pour l'interfaçage avec les autres circuits. Le logiciel de programmation des modules Arduino est une application Java, libre et multi-plateforme, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module).

Le langage de programmation utilisé est le C++ lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maîtrisant le C ou le C++.

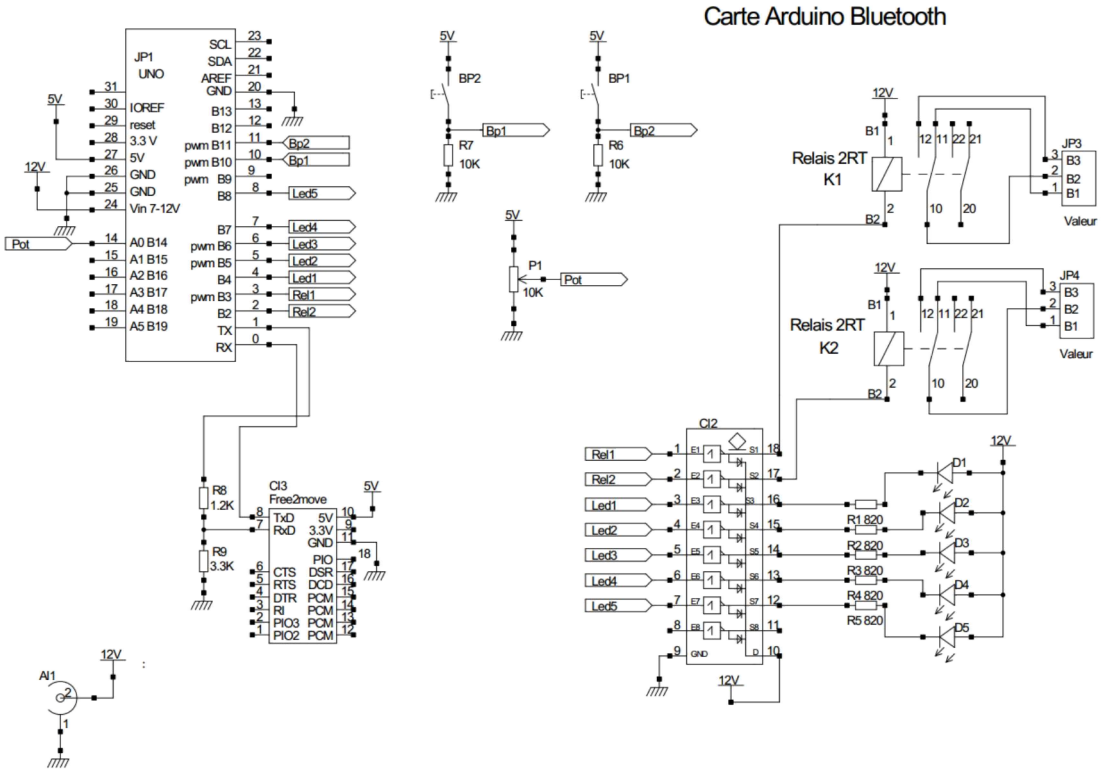
LE CIRCUIT ARDUINO LEODARNO



Les caractéristiques du circuit Leonardo

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz

La carte de test



Mise en pratique

PROGRAMMATION D'UN CHENILLARD.

L'objectif est ici d'allumer et éteindre successivement les cinq leds de la carte dans l'ordre.

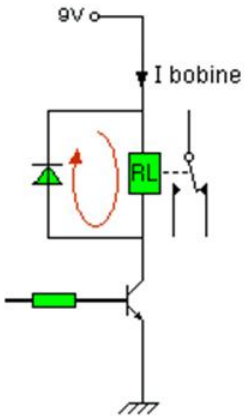
Le programme est à compléter.

```
// Programme chenillard

void setup() {
  // Définition des E/S
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  }

void loop() // tourne en boucle
{
  digitalWrite(8, HIGH);
  delay(1000);


 }
```

COMMANDE D'UN RELAIS (MARCHE ARRET)

On souhaite ici commander le relais avec les deux boutons poussoirs marche et arrêt

Le programme est à compléter




```
// Programme marche/arret

// Mapping des E/S
int REL1=3;
int REL2=2;
int BP1=10;
int BP2=11;

void setup() {
  // Définition des E/S
  pinMode(REL1, OUTPUT);
  pinMode(REL2, OUTPUT);
  pinMode(BP1, INPUT);
  pinMode(BP2, INPUT);
  }

void loop() // tourne en boucle
{
  if (digitalRead(BP1)==1) digitalWrite(REL1, HIGH);

}
```

MESURE D'UNE TENSION

On souhaite maintenant mesurer la tension issue du potentiomètre, celle-ci varie entre 0 et 5v.
Le microcontrôleur va donc devoir effectuer une conversion analogique numérique.

La conversion analogique numérique

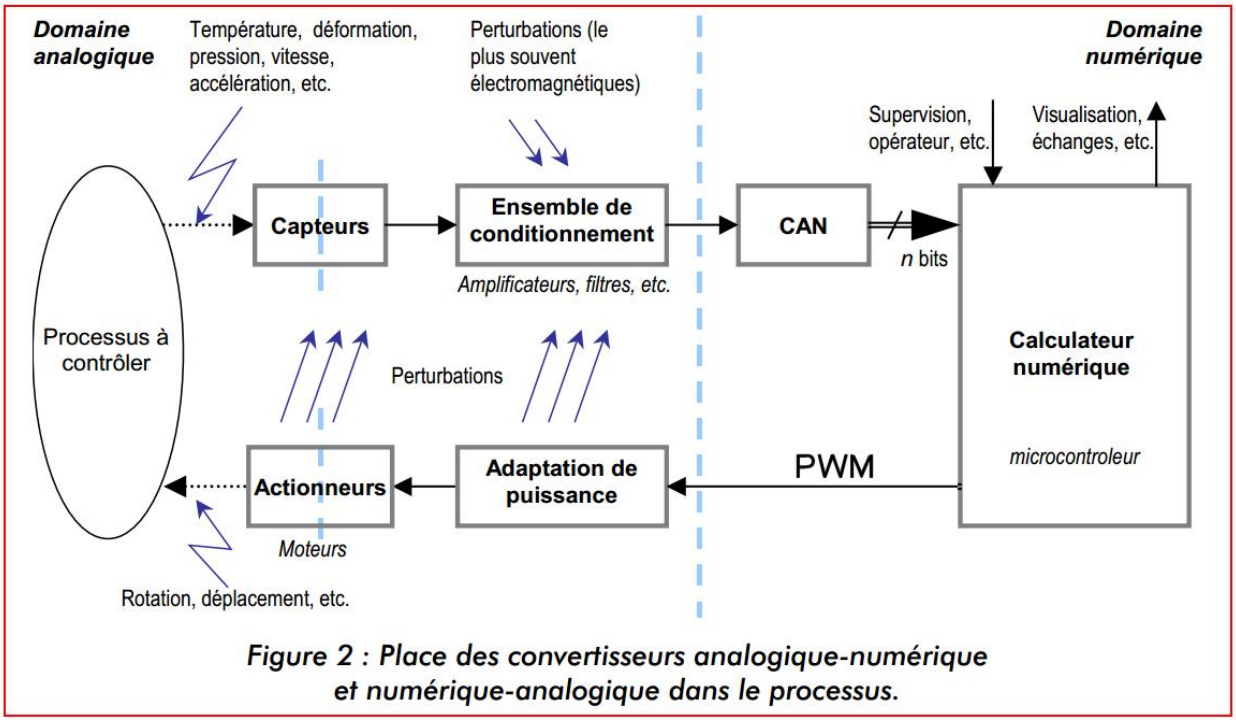
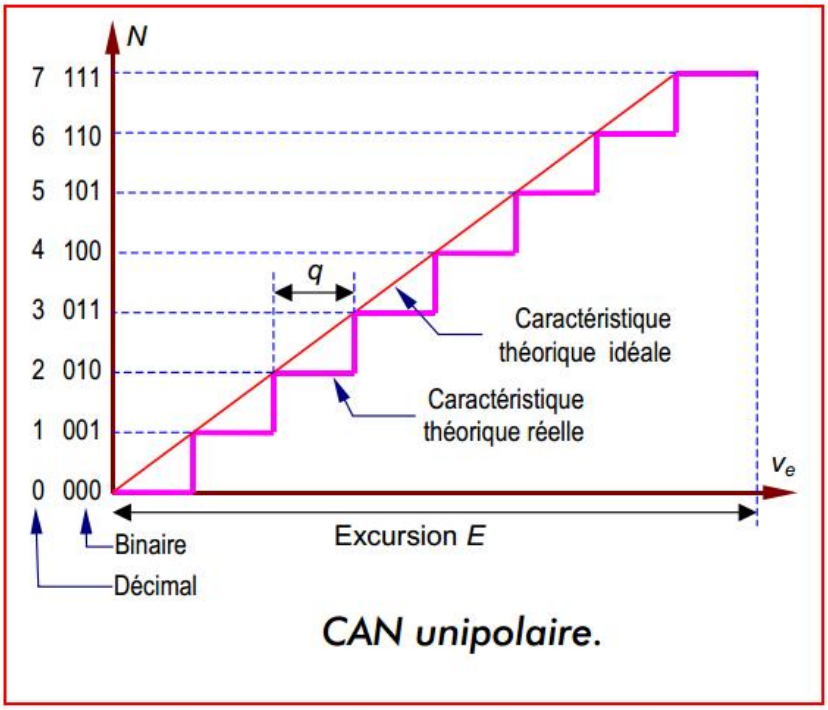


Figure 2 : Place des convertisseurs analogique-numérique et numérique-analogique dans le processus.

Source : www.abcelectronique.com

Caractéristique de transfert



Source : www.abcelectronique.com

Programme avec renvoi sur le port série

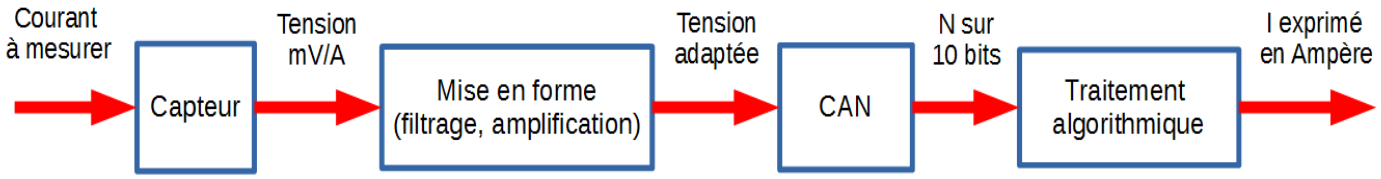
```
// Programme mesure de tension

unsigned int mesure;          // resultat de la conversion CAN sur 10 bits
float tension;                // tension mesuree

void setup() {
  // Initialisation de la liaison série
  Serial.begin(115200);
  }

void loop() // tourne en boucle
{
  mesure = analogRead(0);
  Serial.println (mesure);
  delay (500);
}
```

Compléter ce programme pour calculer et afficher la tension en volt

MISE EN ŒUVRE D'UN CAPTEUR

La mise ne œuvre d'un capteur nécessite le traitement du flux d'information entre la grandeur à mesurer et la valeur finale exprimée dans une grandeur du système d'unité SI.

Chaine d'acquisition

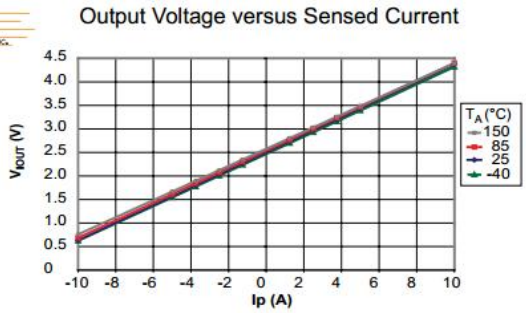


Le capteur utilisé ici est un capteur de courant à effet hall qui converti le courant qui le traverse en une tension. La mesure étant bipolaire, le courant de 0 Ampère correspond à une tension de sortie de 2,5 volts.

La grandeur à mesurer est en général convertie en une tension, le facteur de conversion est ici de 185 mV/A.

Ce signal peur être filtré, avec un filtre passe bas par exemple, pour le rendre plus « propre », si la sensibilité du capteur n'est pas suffisante, il peut être amplifié pour avoir une excursion en entrée proche de 5V.

ACS712

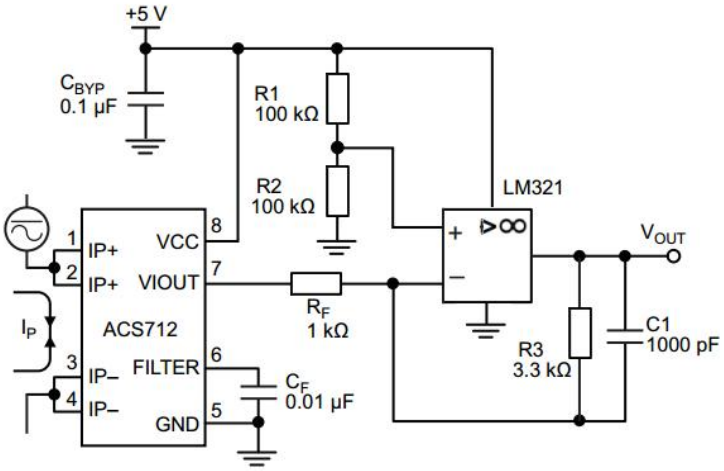


Selection Guide

Part Number	T_{OP} (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	–40 to 85	±5	185
ACS712ELCTR-20A-T	–40 to 85	±20	100
ACS712ELCTR-30A-T	–40 to 85	±30	66

*Contact Allegro for additional packing options.

Le schéma structurel du dispositif de mesure.

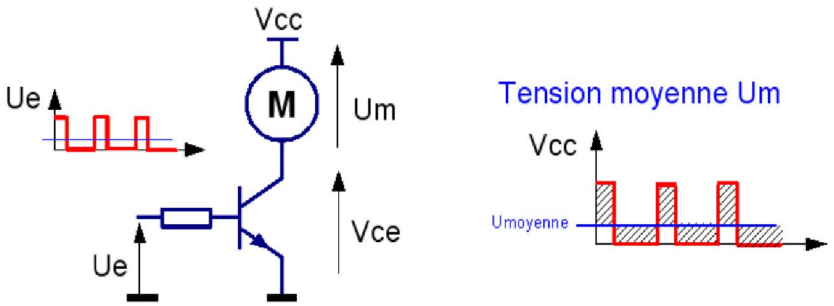


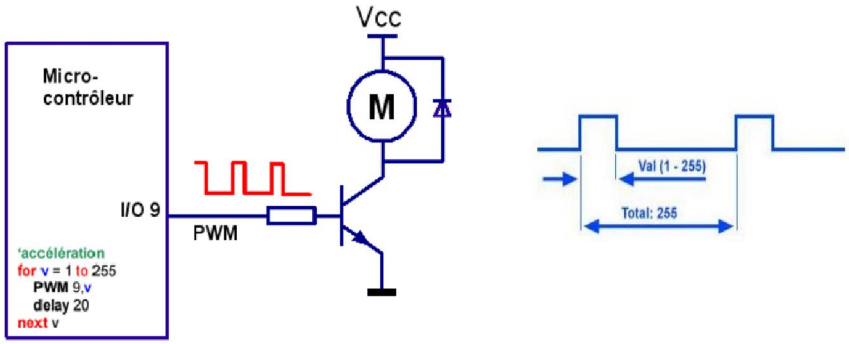
Mesure du courant en Ampère

```
// Déclaration des variables
int mesure;      // valeur issue de la CAN (10 bits)
int mesure_ini;  // valeur mesurée au repos (courant nul)
float courant;   // valeur du courant en Ampère (décimal)

void setup() {
  mesure = analogRead(2);  // CAN
  mesure_ini = mesure;     // courant nul
}

void loop()
{
  mesure = analogRead(2);   //CAN
  mesure = mesure_ini - mesure; //difference avec le courant de repos
  courant = mesure * 27 /1024.0;  //quantum 180mv/A,  5V, 10 bits
  courant = courant / 2.7;  // ampli
  Serial.print(courant);
  Serial.println(" A");
  delay(50);
}
```

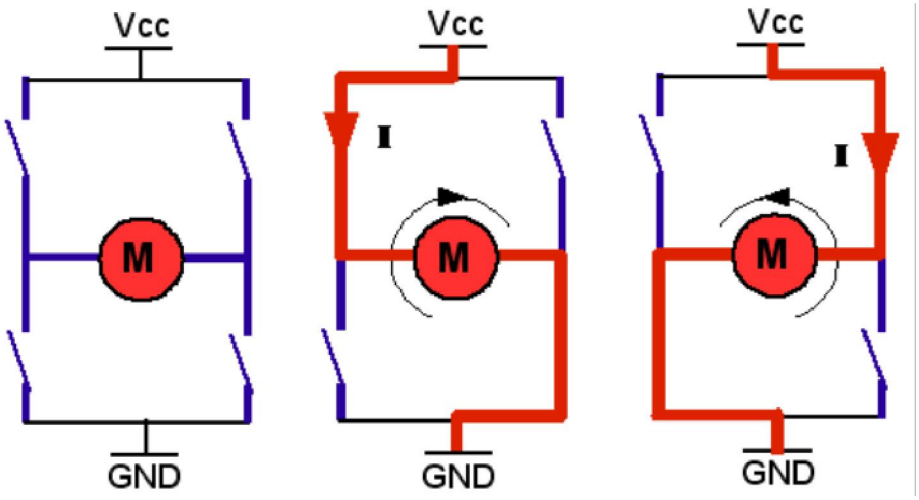
COMMANDE D'UN MOTEUR PAR UNE SORTIE PWM

La sortie PWM est une sortie sur laquelle on peut faire varier le rapport cyclique du signal, la tension moyenne d'alimentation du moteur est alors égale au produit de la tension d'alimentation par ce rapport cyclique.





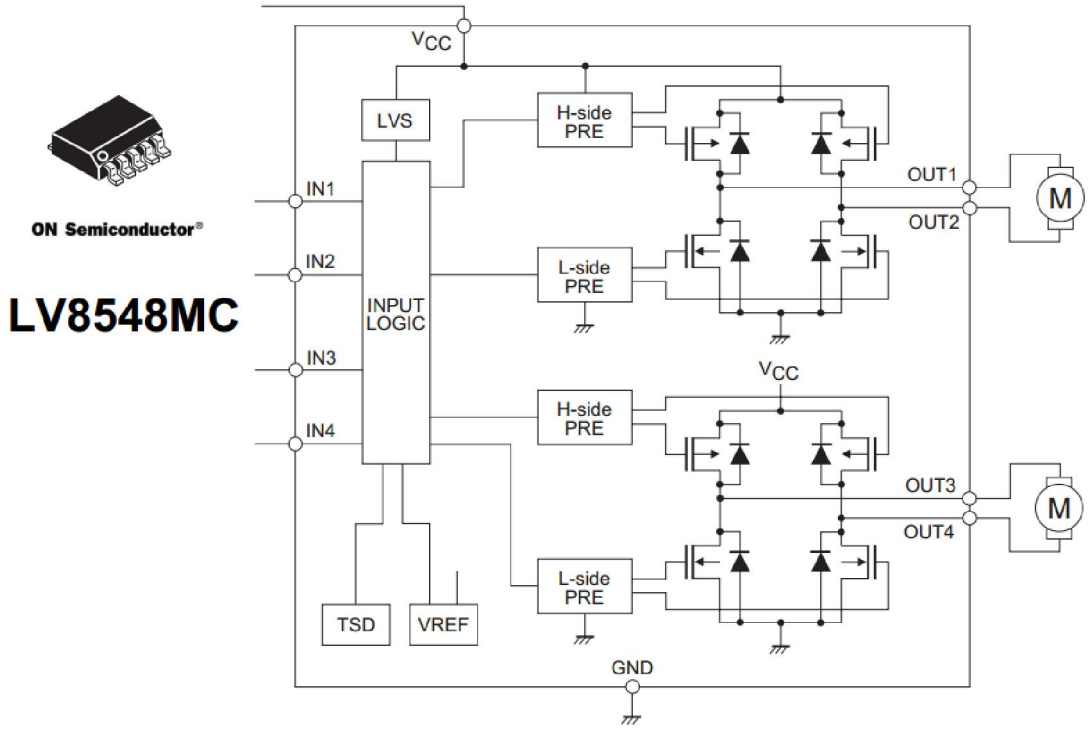
Pour commander le moteur avec deux sens de rotation, il faut mettre en place un dispositif à pont en H, les quatre contacts représentés ici seront en fait des transistors.



Nous utiliserons un petit motoréducteur équipé d'un codeur incrémental, il sera ainsi possible de mesurer sa vitesse voire de faire un petit asservissement.

Le moteur avec encodeur

Le driver de moteur (pont en H)



1. DCM output control logic

Input				Output				Remarks	
IN1	IN2	IN3	IN4	OUT1	OUT2	OUT3	OUT4		
L	L	L	L	OFF	OFF	OFF	OFF	Stand-by	
L	L			OFF	OFF			1CH	Stand-by
H	L			H	L				Forward
L	H			L	H				Reverse
H	H			L	L				Brake
		L	L			OFF	OFF	2CH	Stand-by
		H	L			H	L		Forward
		L	H			L	H		Reverse
		H	H			L	L		Brake

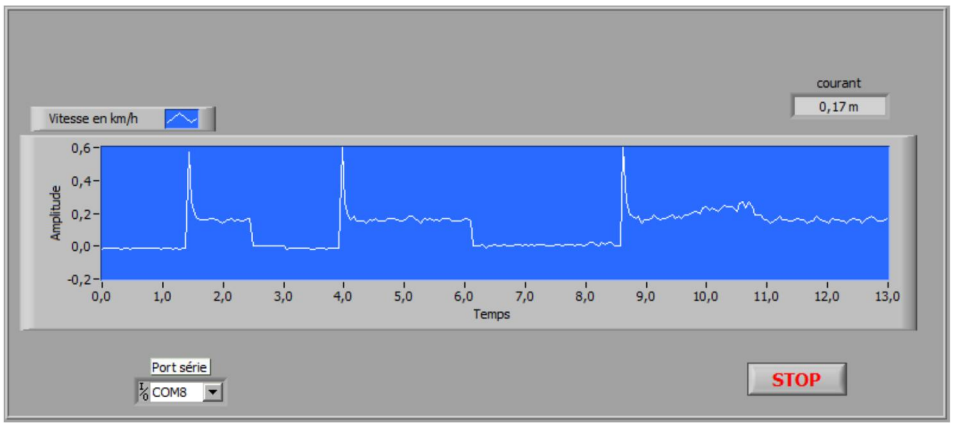
Le programme proposé va recevoir une consigne de vitesse envoyée pas le terminal série, la vitesse mesurée sera affichée en retour.

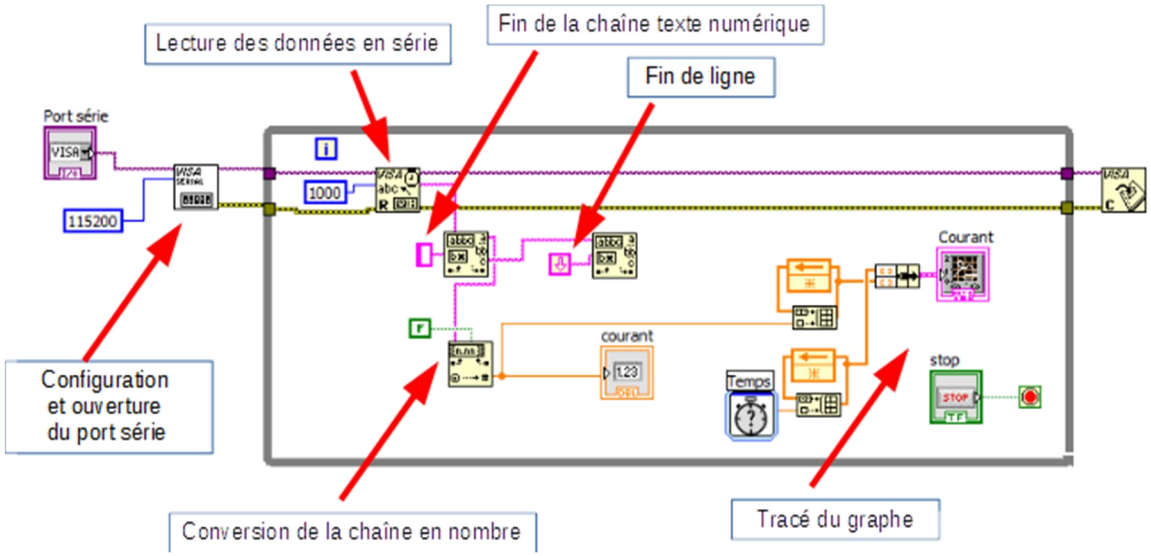
La consigne de vitesse est envoyée avec la moniteur série du programme IDE Arduino et terminée par une fin de ligne (caractère 10).

```
// Programme moteur PWM

// Déclaration des variables
String message = "";              // message reçu sur la liaison série
unsigned long impulsion;
byte pwm;                         //vitesse souhaitée entre 0 et 255

void setup() {
  // Définition des E/S
  pinMode(5, OUTPUT);   //PWM
  pinMode(7, OUTPUT);   //Marche/stop
  pinMode(2, INPUT);    //codeur incremental

  Serial.begin(115200);  // Initialisation de la liaison série
  digitalWrite(7,LOW); //Sortie choix du sens de rotation
}

// Fonction de lecture d'un chaine de caratère série terminée par #
void lire_message(void)
{
  char car;  //variable caractere reçu
  message = "";
  while (car != 10){                    //configurer le passage a la ligne du moniteur serie
    if (Serial.available() > 0){        //si un caractere est présent sur la liaison serie
      car = Serial.read();              //lecture du caractere
      message = String(message + car);  // ajour au message (concaténation)
    }
  }
}

void loop()
{
  lire_message();
  pwm = (message.toInt()); //conversion ASCII en nombre
  analogWrite(5,pwm);    //envoi de la valeur vers la sortie PWM
  delay (500);  //delai de prise de vitesse
  impulsion = pulseIn(2, HIGH,50000);   //mesure de la vitesse (durée impulsion)
  Serial.print (pwm);
  Serial.print (" ");
  Serial.println (impulsion);
}
```

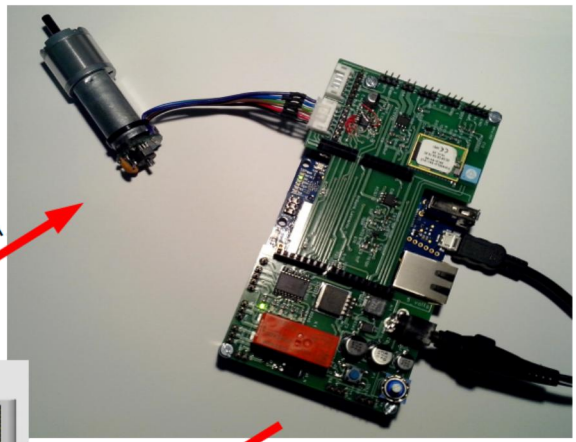
```
// Mesure de courant dans le moteur

// Déclaration des variables
float courant;
int mesure_ini;
long mesure;

void setup() {
  pinMode(5, OUTPUT);   //PWM
  pinMode(7, OUTPUT);   //Marche/stop
  // Initialisation de la liaison série
  Serial.begin(115200);
  //choix du sens de rotation
  digitalWrite(7,LOW);
  //mesure du courant au repos
  mesure_courant();
  mesure_ini = mesure;
}

void mesure_courant() //moyenne de 10 mesures
{
  mesure = 0;
  for (int i=0; i < 20; i++){
      mesure = mesure + analogRead(2);
      delay(1);
   }
   mesure = mesure / 20;
}

void loop()
{
  if (digitalRead(11)==1) //marche arret
      {analogWrite(5,200);}
    else
      {analogWrite(5,0);};
  mesure_courant();
  mesure = mesure_ini - mesure; //difference avec le courant de repos
  courant = mesure * 27 /1024.0;  //quantum 180mv/A,  5V, 10 bits
  courant = courant / 2.7;  // ampli
  Serial.print(courant);
  Serial.println(" A");

  delay(50);
}
```

La mesure du courant envoyée sur le port série peut être récupérée dans un programme Labview





Exemple d'asservissement de vitesse avec Labview



Labview gère
l'asservissement

Consigne PWM

Vitesse

La carte Arduino pilote le moteur avec la consigne demandée et retourne la vitesse

Correction des exemples

```
// Programme marche/arret relais

int REL1=3;
int BP1=10;
int BP2=11;

void setup() {
  // Définition des E/S
  pinMode(REL1, OUTPUT);
  pinMode(BP1, INPUT);
  pinMode(BP2, INPUT);
  }

void loop() // tourne en boucle
{
  if (digitalRead(BP1)==1) digitalWrite(REL1, HIGH);
  if (digitalRead(BP2)==1) digitalWrite(REL1, LOW);
}
```

```
// Programme mesure de tension

unsigned int mesure;          // resultat de la conversion CAN sur 10 bits
float tension;       // tension mesuree

void setup() {
  // Initialisation de la liaison série
  Serial.begin(115200);
  }

void loop() // tourne en boucle
{
  mesure = analogRead(0);
  tension = (mesure * 5.0) / 1023;
  Serial.print (tension);
  Serial.println (" volts");
  delay (500);
}
```

```
// Programme chenillard

void setup() {
  // Définition des E/S
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);


  digitalWrite(8, HIGH);  // allumage d'une led
  }


void loop() // tourne en boucle
{
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
  digitalWrite(7, HIGH);
  delay(1000);
  digitalWrite(7, LOW);
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(6, LOW);
  digitalWrite(5, HIGH);
  delay(1000);
  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  delay(1000);
  digitalWrite(4, LOW);
}
```