Formation Arduino appliquée au monde radioamateur

F4GSC - Jean-Luc Levant - ARALA Nantes - 2014

Plan de la formation

- Objectifs de cette formation
- Description de la carte ArduinoUNO R3
- Description de quelques shields Arduino
- Description du Mega328P
- Quelques éléments du language de programmation
  - Structure d'un programme Arduino
  - Définition des variables et des constantes
  - Structures de répitition
  - Structures de test
- Description des interfaces numériques et analogiques
  - But de ces interfaces
    - Numériques
    - Analogiques

Plan de la formation

- Description des entées/sorties numériques simples
  - Description et fonctionnement
    - Exemple 1 : Clignotant d'une LED à vitesse programmable.
    - Exemple 2 : Décodage et envoi du message « SMS » en code morse.
    - Exemple 3 : Commande d'un pont en H (à faire).

- Le convertisseur analogique numérique
  - Description et fonctionnement
    - Exemple 1 : Réalisation d'un voltmètre , 0-10v.
    - Exemple 2 : Réalisation d'un capacimètre , Cx >2nF.

- Le PWM (Pulse Width Modulation)
  - Description et Fonctionnement
  - Contrôle de l'intensité lumineuse d'une diode LED
    - Contrôle continu
    - A partir des touches '+' et '-'
  - Convertisseur Numérique/analogique
    - Exemple 1 : Génération d'une rampe.
    - Exemple 2 : Génération d'un signal triangulaire.
    - Exemple 3 : Génération d'un signal sinusoïdal.
  - Modulation tout ou rien
    - Générateur d'appel général audio (CQ de F5KEQ).

Plan de la formation

- Les Liaisons Séries Synchrones et Asynchrones
  - Série Asynchrone
    - Description et Fonctionnement
    - Interface avec l'Hyperterminal
  - I2C Synchrone
    - Description et Fonctionnement
- Les interruptions

Introduction

Introduction

- D'où vient le nom arduino?
  - En l'an 1002, le roi Arduin (Arduino en italien) devint le seigneur de la petite ville d'Ivrea.
  - Un bar dans une rue pavée de la ville, honore sa mémoire, Bar di Re Arduino.
  - Massimo Banzi, un des fondateur de ce concept, a pour habitude d'étancher sa soif dans ce bar. Le nom du projet électronique Arduino fait référence à ce bar.
- La philosophie
  - Populariser le développement de l'électronique et de l'informatique industrielle.
  - Apprendre l'électronique par la pratique au lieu de commencer par apprendre l'algèbre et les lois de l'électronique (approche anglo-saxonne).
  - Disposer d'une plateforme matérielle et logicielle à bas coût.
  - Partager les développements matérielles sous licence Creative Commons et logicielles sous licence open source.

Introduction

- Les plus
  - Pouvoir réutiliser les développements matériels et logiciels (librairies) créer par d'autres développeurs.
  - L'héritage des librairies permet de réduire considérablement le temps développement des systèmes.
  - La complexité de l'assemblage de fonctions électroniques et informatique est réduite[1].
  - Le développeur peut se concentrer sur le développement de son idée et facilite donc la création et l'innovation.
  - Cette simplification permet à tout un public n'ayant pas les compétences en électronique et informatique de créer et innover.

- Note

[1] Le temps d'apprentissage d'un microcontrôleur peut prendre quelques jours à quelques mois en fonctions de la complexité des périphériqes qu'il intègre. Les librairies open source permettent de s'affranchir de cet apprentissage en utilisant des fonctions logicielles qui intègrent la complexité de programmation des périphériques.

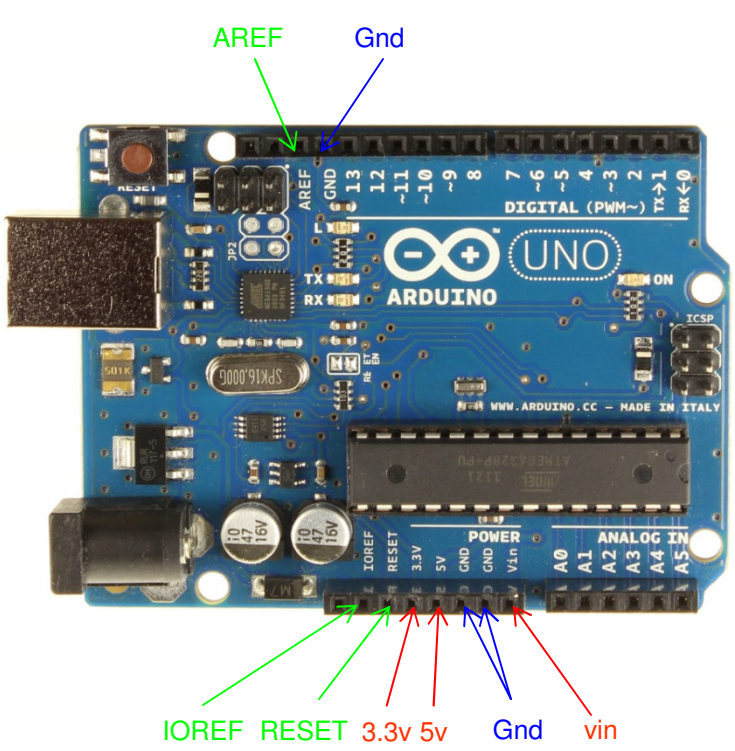
Introduction

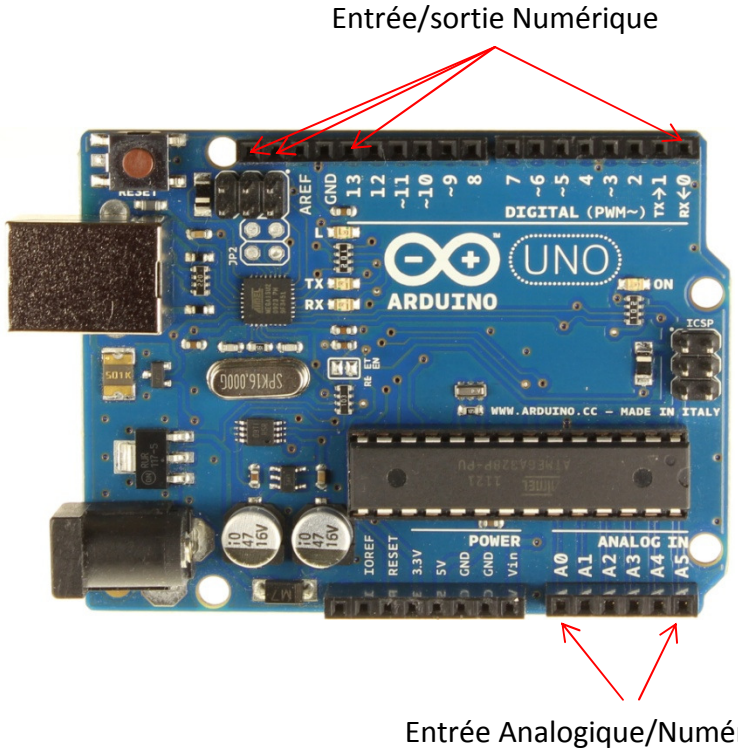
- Les moins
  - Les optimisations logicielles (taille,temps d'éxécution…) se sont possibles qu'aux développeurs ayant des connaissances approndies.

Description de la carte Arduino Uno

Description de la carte Arduino UNO R3

- Les interfaces d'entrées/sorties numériques et analogiques





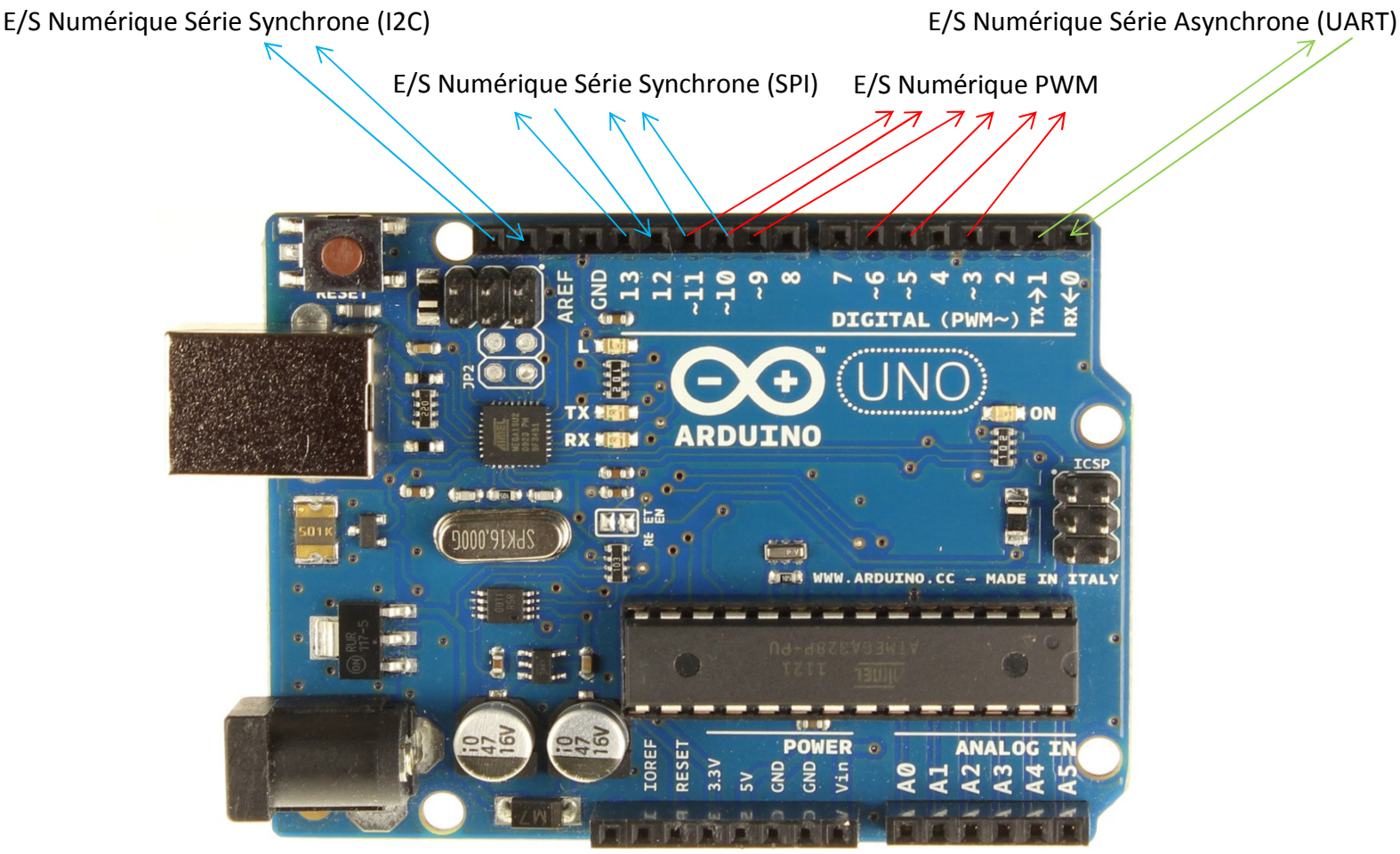
Vin : alimentation externe (7-12v)

AREF : source de référence de tension externe

IOREF : Fournit le 5v comme référence pour les cartes shields.

Description de la carte Arduino UNO R3

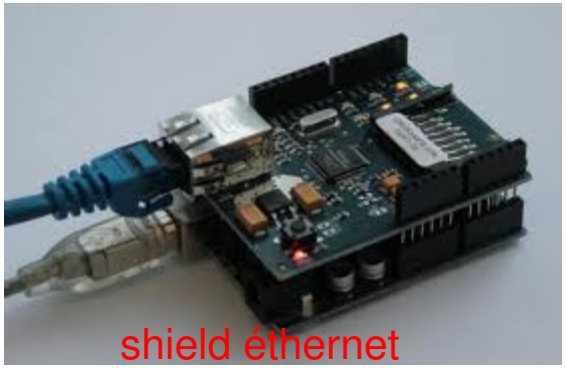
- Les Entrées/Sorties Spéciales

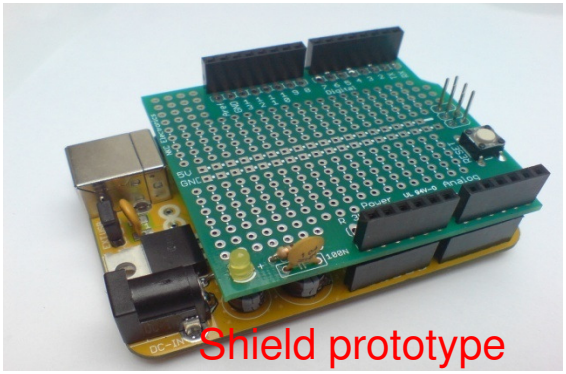


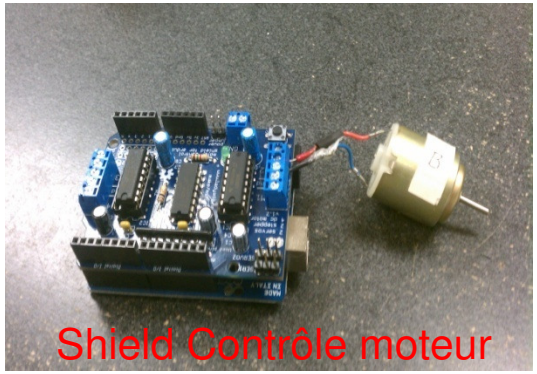
Exemples de quelques shields

Exemples de quelques shields

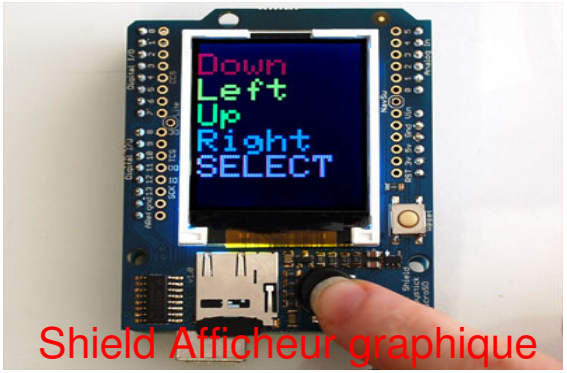
- Attention à la compatibilité des tensions d'alimentation. 5v pas compatible avec le 3.3v.


shield éthernet


Shield prototype

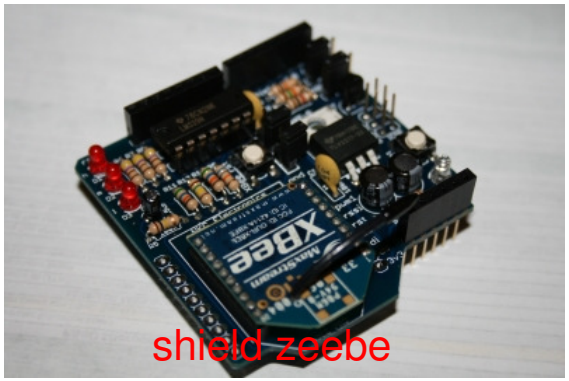

Shield Contrôle moteur


shield GPS/GSM



Shield Afficheur graphique


Shield télémétrie ultrason



shield zeebe

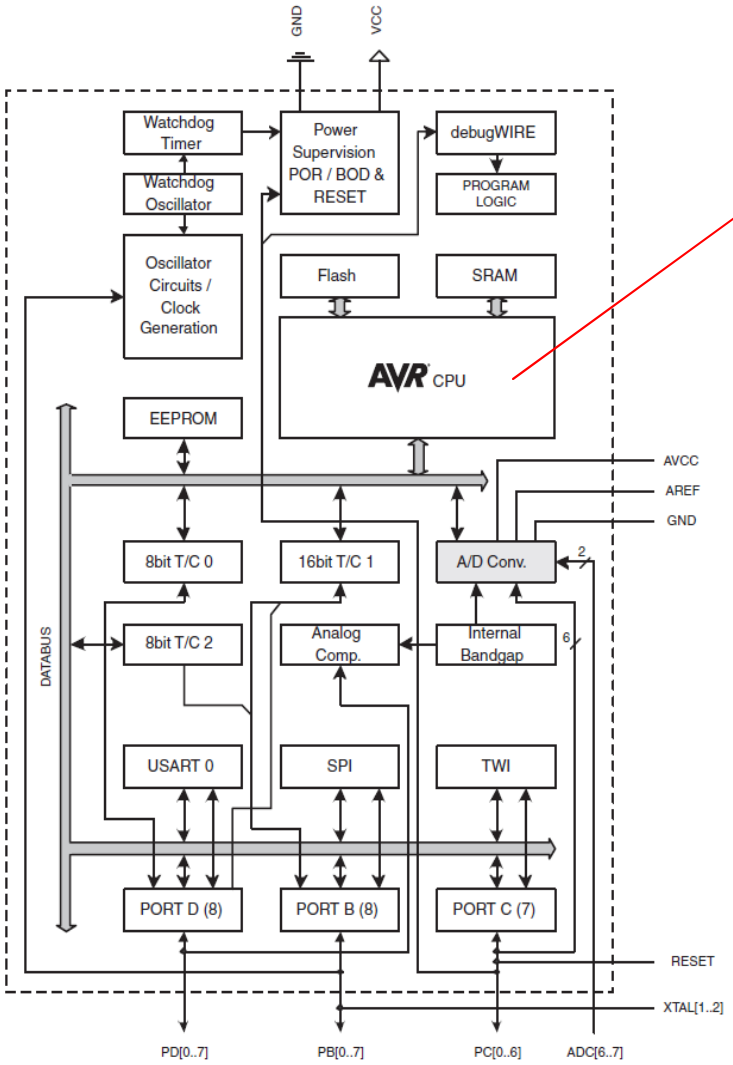

shield audio


Shield SD card

Description de l’Atmega328P

Description de l'ATmega328P

- Les fonctions embarquées dans l'ATmega328P



- **CPU (Computing Processor Unit)**
  - **Processeur RISC 8-Bit**

- Architecture Harward → les bus programme et données sont séparées. Une instruction et une donnée peuvent être accédées en parallèle.

- RISC (Reduced Instruction Set) avec 2 opérations éxécutées en paralèle (20MIPS).

Lecture de l'instruction à éxécuter → n

Exécution de l'instruction → n-1

- 32 registres internes pour accélérer les calculs et les accès aux données.

- En charge une instruction est éxécutée en 1 cycle d'horloge → Fcpu= 16MHz → Tcpu=66ns.

- 2 multiplieurs 16-bits

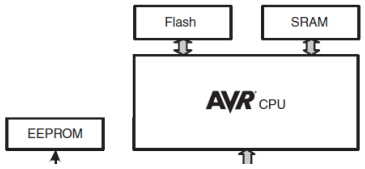
- Architecture optimisée pour le langage C.

Description de l'ATmega328P

- Les fonctions embarquées dans l'ATmega328P
  - Les ressource matérielles
    - **Les plans mémoire**

Mémoire	Taille	Opération d'écriture/Lecture
Programme	- **32 kOctets** de mémoire Flash.	- Ecriture au chargemnt du programme.
Données Volatiles	- **2 koctets** de mémoire RAM.	- Ecriture/lecture Accessible à tout moment.
Données non Volatiles	- **1 koctets** de mémoire EEPROM. - **Plusieurs koctets** de mémoire Flash.	- Ecriture/lecture Accessible à tout moment. - Ecriture au chargemnt du programme. - Lecture à tout moment.

Notes :



- **FLASH** : **Proppriété qui permet d'effacer/écrire rapidement une grande portion de la mémoire**
  -Technologie qui permet de réaliser une mémoire qui peut être **écrite/effacée électriquement**.
  - Le contenu de la mémoire est **maintenu** lorsque l'alimentation est éteinte.

- **RAM** : **Random Access Memory**
  - Technologie qui permet de lréaliser une mémoire qui peut être **écrite/effacée électriquement.**
  - Le contenu est **perdu** lorsque l'alimentation est éteinte.

- **EEPROM** : **Electrically-Erasable Read Only-Memory**
  - Technologie qui permet de réaliser une mémoire qui peut être **écrite/effacée électriquement.**
  - Le contenu de la mémoire est **mainetnu** lorsque l'alimentation est éteinte.

Description de l'ATmega328P

- Les fonctions embarquées dans l'ATmega328P
  - Les ressource matérielles (boîtier PDIP)
    - **Les périphériques**

      - Oscillateur RC interne calibré (8MHz).
      - Oscillateur externe (32KHz /0.4MHz - 16MHz).
      - 2 x Timer/Compteur 8bits.
      - 1 x Timer/Compteur 16bits.
      - 1 x Timer/compteur temps réel.
      - 1 x timer chien de garde avec oscillateur séparé.
      - Pré-diviseur d'horloge 8bits.
      - 6 x PWM.
      - 6 x Entrées analogiques 10-bits avec mesure de la température.
      - 1 x UART.
      - 1 x liaison série synchrone SPI (Maître et esclave).
      - 1 x liaison série synchrone I2C.
      - 1 x comparateur analogique.
      - 26 x sources d'interruption internes et externes.
      - Tension d'alimentation : 1.8v à 5.5v.

Description de l'ATmega328P

- Les fonctions embarquées dans l'ATmega328P
  - Les ressource matérielles
    - **Les autres membres de la famille**

Device	Flash	EEPROM	RAM
ATmega48PA	4K Bytes	256 Bytes	512 Bytes
ATmega88PA	8K Bytes	512 Bytes	1K Bytes
ATmega168PA	16K Bytes	512 Bytes	1K Bytes
ATmega328P	32K Bytes	1K Bytes	2K Bytes

Quelques éléments du language de programmation C

Quelques éléments du language de programmation C

- Structure d'un programme Arduino

Inclusion d'un fichier comprenant les fonctions logicielles permettant le pilotage des fonctions Électroniques matérielles ("shield").

```
//============================================================
//-- La librairie LCD est utilisée.
//============================================================
#include <LiquidCrystal.h>
```

Affectation des noms symboliques au broches d'E/S de la carte Arduino.

```
//============================================================
//-- La diode LED est connectée à la broche 13.
//============================================================
 Int led=13;
```

Construction du périphérique "lcd" et précise comment ce périférique se connnecte à la carte arduino (quels signaux sont utilisés).

```
//============================================================
//-- Construit un objet LCD et précise que les broches 12,11,5,4,3,2 connecte
//-- le LCD à la carte Arduino.
//------------------------------------------------------------------------------------------
//--  broche 12 : RS -11:Enable – 5: D4 – 4:D5 -3 : D6  - 2 : D7
//============================================================
 LiquidCrystal  lcd(12, 11, 5, 4, 3, 2);
```

Boucle de programme éxécutée à la mise sous-tension ou après une remise à zéro.

```
//============================================================
 void setup()
  //-- Définit le nombre de caractère par ligne ainsi que le nombre de ligne
  lcd.begin(16, 2);
  //-- Affiche le message "Hello World".
  lcd.print("hello, world!");
}
```

Boucle de programme principale éxécutée à la mise sous tension ou après une remise à zéro (boucle sans fin).

```
//============================================================
 void loop()
 Int compteur;
  // set the cursor to column 0, line 1
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
  ++compteur;
}
//============================================================
```

Quelques éléments du language de programmation C

- Définition des variables et des constantes

  - **Constante** → FLASH, E²PROM, RAM
    - Donnée fixe qui **ne change pas** lors de l'éxécution du programme.

  - **Variable** → E²PROM et RAM
    - **Définition**
      - Donnée dont la **valeur est modifiée** en fonction de l'activité du programme.

    - **Type d'une variable**
      - Définie la **nature de la donnée** ainsi que sa **taille** (dépend de l'architecture du microcontrôleur (8,16,32 ou 64-bits);

      - Définie le **signe de la donnée** : signée → positive ou négative, non signée → positive.

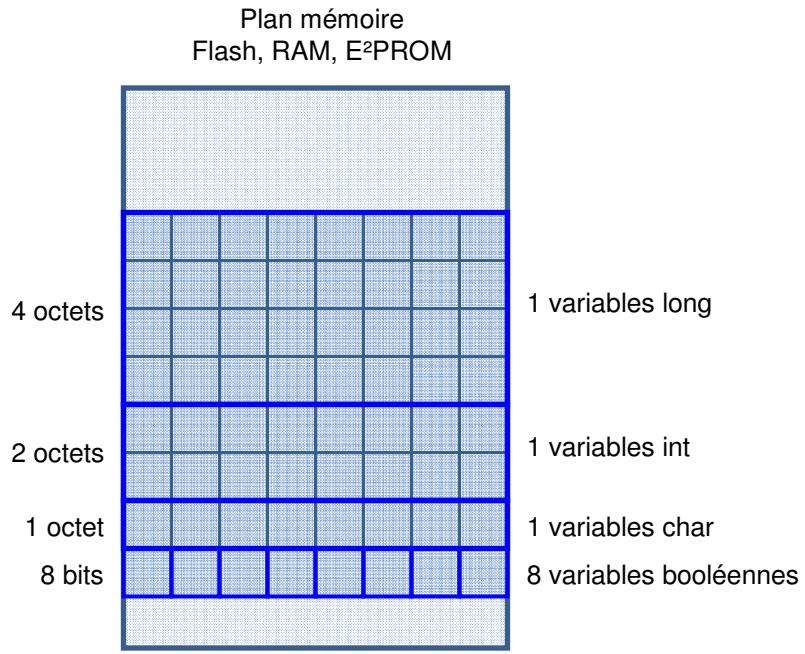
Quelques éléments du language de programmation C

- Définition des variables et des constantes
  - Taille et dynamique de codage des données (variables, constantes…)

Type (Atmega328)	boolean	char	int	long	float	double
Taille	1-bit	8-bits	16-bits	32-bits	32-bits	64-bits
Nbre d'octets		1	2	4	4	8
Dynamique de codage (signé)	0 → 1	-127 → +128	-32768 → +32767	−2 147 483 648 → +2 147 483 647	-3.40282347e+38 → +3.40282347E38	-1.79769313486231570e308 → 1.79769313486231570e308
Dynamique de codage (non signé)	0 → 1	0 → 255	0 → 65535	0 → 4 294 967 295		
Exemple	Boolean ON;	Char freq;	Int vitesse;	long distance;	float freq ;	double freq;

Quelques éléments du language de programmation C

- Définition des variables et des constantes
  - Type d'une variable (suite)
    - Occupation mémoire
      - Le type détermine **combien d'octets** seront utilisées pour **loger la variable** ou la constante dans le **plan mémoire**.



Quelques éléments du language de programmation C

- Définition des structures de répitition conditionnelle
  - Structure qui permet de répéter plusieurs fois le même traitement jusqu'à ce que la condition de fin apparaisse.

- **Tant que** la condition n'est pas vérifiée **alors faire** le traitement

```
int i;
 ….

for (i=1 ; i<=10 ; i++)
 {
  serial.print("I = "); serial.printl(i);
 }
```

Le traitement est effectué 10 fois. La variable **i** est testée puis incrêmentée avant chaque traitement. Le nombre de passage dans la boucle est déterminée dès le départ.

Quelques éléments du language de programmation C

- Définition des structures de répitition conditionnelle
  - Structure qui permet de répéter plusieurs fois le même traitement jusqu'à ce que la condition de fin apparaisse.

- **Si** la condition est vérifiée **alors répéter** le traitement

```
int i;
 ....
i=1;
 while (i<=10)
{
 serial.print("I = "); serial.printl(i);
 i=i+1;
}
```

Le traitement est effectué 10 fois. La variable **i** est d'abord testée puis incrêmentée dans la boucle de traitement si la condition de fin n'est pas atteinte. Le nombre de passage dans la boucle dépend de comment est géré la variable i.

Quelques éléments du language de programmation C

- Définition des structures de répitition conditionnelle
  - Structure qui permet de répéter plusieurs fois le même traitement jusqu'à ce que la condition de fin apparaisse.

- **Faire** le traitement
  si la condition est vérifiée **alors répéter** le traitement

```
int i;
 ....
i=1;
do
{
 serial.print("I = "); serial.printl(i);
 i=i+1;
}
while (i<=10);
```

Le traitement est effectué au minum une fois puis répété tant que **I <= 10**

Quelques éléments du language de programmation C

- Définition des structures de test conditionnelle
  - Ces structures permettent d'appliquer un traitement logiciel particulier en fonction d'un évènement qui apparait ou en fonction de l'évolution d'une variable.

- **Si** la condition est vérifiée **alors faire** traitement1
  sinon faire traitement2

```
int condition;
 ....
 condition=10;
 ....
 if (condition==1)
 {
   serial.print("condition = "); serial.println(condition);
 }
 else
 {
   serial.print("condition = "); serial.println(condition);
 }
```

Quelques éléments du language de programmation C

- Définition des structures de test conditionnelle
  - Ces Structures permettent d'appliquer un traitement logiciel particulier en fonc tion d'un évènement qui apparait ou en fonction de l'évolution d'une variable.

- **Si** la condition1 est vérifiée **alors faire** traitement1
  sinon si la condition2 est vérifiée **alors faire** traitement2
  sinon si la condition3 est vérifiée **alors faire** traitement3
  sinon si la condition4 est vérifiée ….

```
int condition;
….
condition=2;
….
if (condition==1)
{
  serial.print("condition 1 = "); serial.printl(condition);
}
elseif (condition==2)
{
  serial.print("condition 2 = "); serial.printl(condition);
}
elseif (condition==3)
{
 serial.print("condition 3 = "); serial.printl(condition);
}
elseif (condition==4) {….}
```

Quelques éléments du language de programmation C

- Définition des structures de test conditionnelle
  - Ces Structures permettent d'appliquer un traitement logiciel particulier en fonc tion d'un évènement qui apparait ou en fonction de l'évolution d'une variable.

- **Test** (var)
  si var=condition1 **alors faire** traitement1
  si var=condition2 **alors faire** traitement2 ……

```
char var;
 ….
 var = 'A';
  ….
 Switch (var)
 {
   case 'A' :
   {
     serial.print("var_A = "); serial.printl(var);break ;
   }
   case 'B' :
   {
      serial.print("var _B= "); serial.printl(var);break ;
   }
  …
 }
```

Description des fonctions d’Entrées/Sorties

Les Entrées/Sorties Numériques Simples

- **Propriétés**
  - Ce sont des E/S qui ont deux états et dont ceux-ci sont contrôlés par l'utilisateur (programme).
  - Celles qui ne sont pas des E/S simples sont des E/S analogiques et des E/S spécifiques.

- **Actions à faire par programmation pour utiliser ces E/S.**
  - **Indiquer quelle broche est utilisée.**
    - **int Signal1 =2;**
  - **Indiquer la direction de l'E/S → est-ce que l'utilisateur à besoin d'une Entrée ou une Sortie?**
    - **pinMode(Signal1,OUTPUT);** ou **pinMode(Signal1,INPUT);**
  - **En fonction du besoin du programme, forcer la Sortie à l'état haut(VDD) ou bas (GND) ou lire l'Entrée.**
    - **Sortie** : **digitalWrite(Signal1,HIGH);** ou **digitalWrite(Signal1,LOW);**
    - **Entrée** : **var = digitalRead(Signal1);**

Les Entrées/Sorties Numériques Simples

- Exemple 1 – Utilisation des entrées/sorties
  - Faire clignoter la led associée à la broche 13 avec une période de 1 seconde si la broche 2 est à l'état 1 (VDD=5v) et à 2 secondes si elle est à 0 (Gnd=0v).

    - **Déclaration des broches utilisées**

```
int Led=13 ;
int Interrupteur= 2 ;
```

    - **Indiquer la direction des broches 2 et 13**

```
Void setup
{
    pinMode(Led,OUTPUT);
    pinMode(Interrupteur,INPUT);
}
```

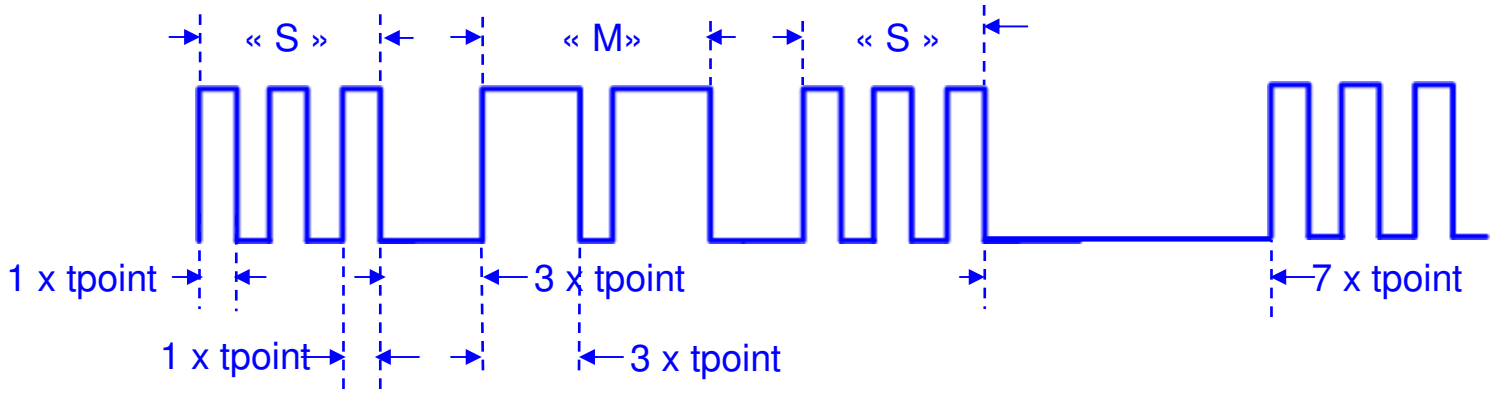
    - **Programme**

```
Void loop()
{
boolean Toggle;

    if (digitalRead(Interrupt)==1) delay(1000) ;
    else delay(2000) ;
    if (Toggle==1)
    {
       digitalWrite(Led,LOW) ; Toggle=0;
    }
    else
    {
        digitalWrite(Led,HIGH) ; Toggle=1;
    }
}
```

C.F expérimentation_1

Les Entrées/Sorties Numériques Simples

- Exemple 2 – Utilisation des sorties
  - Décoder et Générer en code morse le message « SMS »
    - Durée point : tpoint = 200ms
    - « trait : 3 x tpoint
    - Interval entre deux lettres : Interv1 = 3 x tpoint
    - Interval dentre deux mots : Interv2 = 7 x tpoint



C.F expérimentation_2

Les Entrées/Sorties Numériques Simples

- Exemple 2
  - Programme

```
int Led=13;

//---- 0 : 1 point, 1: un trait, 2: séparation entre
 //---deux lettres, 3 : séparation entre deux mots.
byte Morse_Tab[]={1,1,1,2,1,1,2,1,1,1,3};
int  point=200;

void setup() {
 pinMode(Led,OUTPUT);
}
void loop() {
  byte code ;
  int i;
```

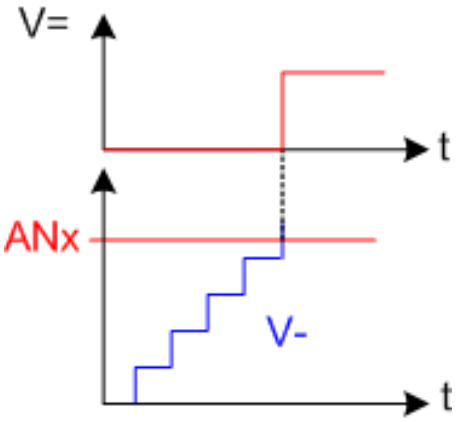
```
while(1){

   //--- Décodeur morse
   switch (Morse_Tab[i]){
    case 0: {
      digitalWrite(Led,HIGH); delay(point);
      digitalWrite(Led,LOW); delay(point);
      break;
     }
    case 1: {
      digitalWrite(Led,HIGH); delay(2*point);
      digitalWrite(Led,LOW) ; delay(point);
      break;
    }
    case 2:{
      digitalWrite(Led,LOW); delay(3*point);
      break;
    }
    case 3:{
      digitalWrite(Led,LOW); delay(7*point);
      break;
    }
   };
   ++i;
   if (i>10)i=0;
  }
 }
```

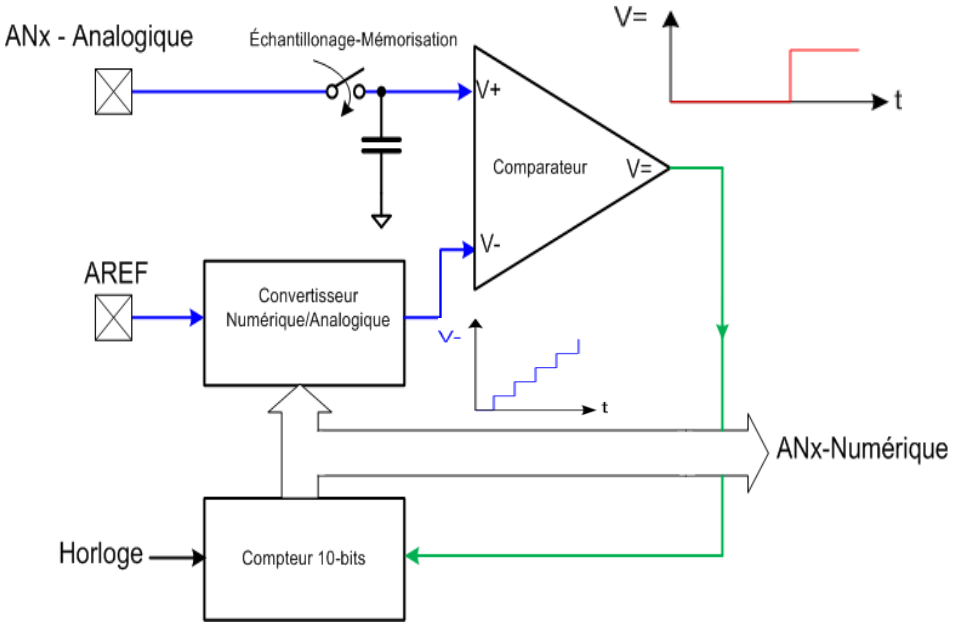
Convertisseur Analogique/Numérique

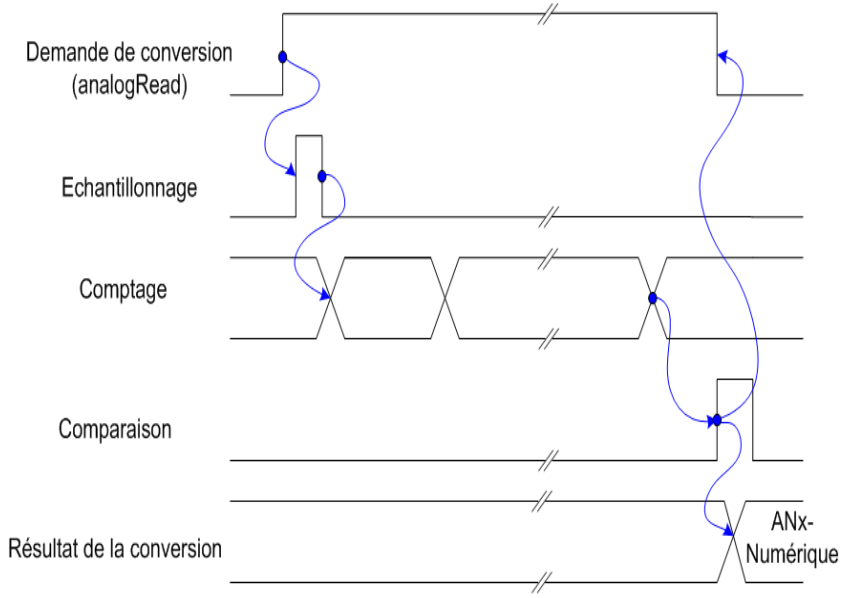
- **Définition et fonctionnement**
  - Fonction électronique intégrée dans µCTRL qui asssure l'interface entre me monde analogique et le monde numérique → Conversion analogique/numérique (CAN).
    - **Monde numérique**
      - **Bit**: chiffre binaire qui ne peut prendre que deux valeurs :
        - » 0 ou 1
        - » 0 ou VDD : Grandeur binaire au sens électrique du terme → 0 la masse → VDD 5v,3v,…
      - **Chiffre codé sur N bits**
        - » 4 bits : codage de 16 valeurs
        - » 8 bits : codage de 256 valeurs
        - » 16 bits : codage de 65536 valeurs
    - **Monde analogique**
      - Grandeur qui peut prendre une infinité de valeur au cours du temps qui passe.
      - Il faudrait une infinité de bit pour coder une grandeur analogique → pas réalisable physiquement.
      - Résolution (la plus petite valeur que le CAN peut coder:
        - » **Res= Vref/2^Nbits,** exemple Vref=5v , Nbits=8 → Res = 5/256 =19mV.
      - Le convertisseur analogique numérique arrondit donc la grandeur analogique (troncature).

Convertisseur Analogique/Numérique

- **Description et fonctionnement**
  - Un convertisseur numérique/analogique 10-bits fournit une tension de comparaison à l'entrée V- du comparateur qui est contrôlée par le compteur 10-bits.
  - Lorsque la tension de référence V- est égale à la tension analogique V+ le processus de conversion s'arrête et fournit la grandeur analogique convertie en numérique (ANx-analogique → ANx-Numérique).

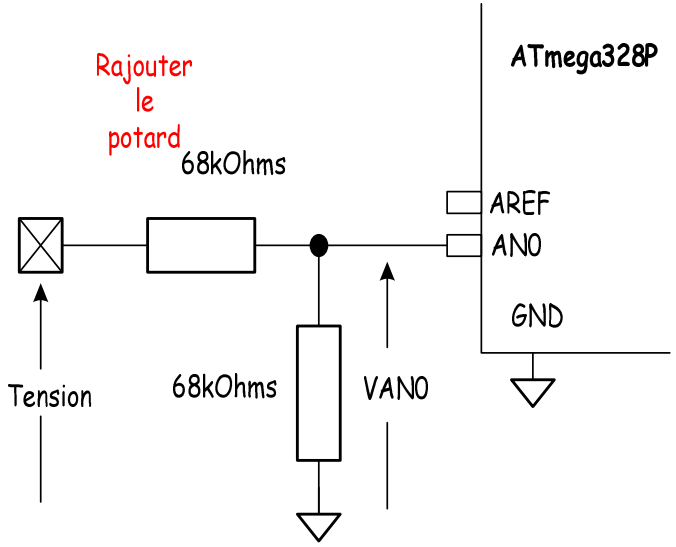






Convertisseur Analogique/Numérique

- **Example 1 : Réalisation d'un voltmètre**
  - Le principe consiste à utiliser le CAN pour mesurer une tension analogique comprise entre 0 et 10v. La lecture sera envoyée sur le port de la liaison de mise au point (Tool → Serial link).



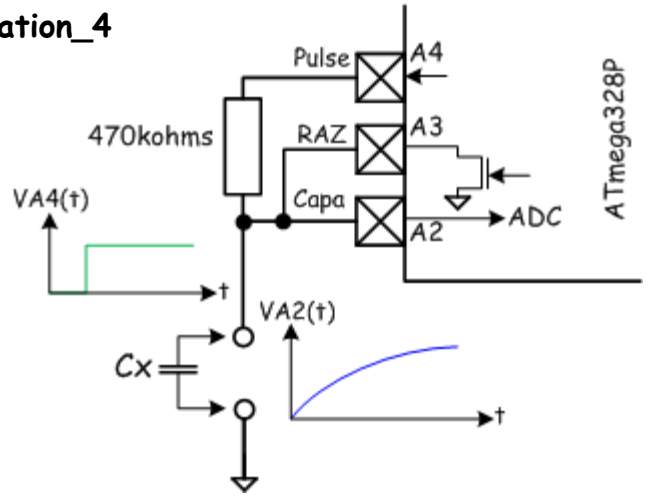

```
const int voltMetrePin = A1;
float Tension;
void setup() {
Serial.begin(9600);
analogReference(DEFAULT);
}

void loop() {
  Tension=analogRead(voltMetrePin)*10.0/1023;

  Serial.print(" Tension (mV) = ");Serial.println(Tension);
}
```

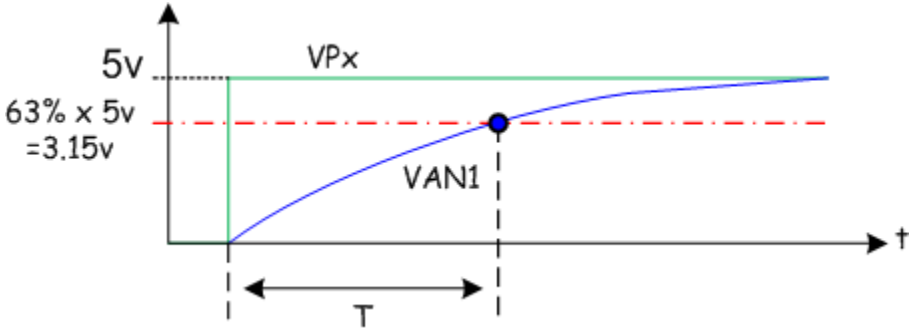
C.F expérimentation_3

Convertisseur Analogique/Numérique

- **Exemple 2 : Réalisation d'un capacimètre.**
  - Le principe de la mesure de Cx consiste à la charger électriquement par l'intermédiaire d'un échelon de tension et d'une résistance connue.
  - Le convertisseur analogique numérique poursuit la tension jusqu'à ce qu'elle ait atteinte 63% de sa valeur finale. Un compteur logiciel mesure le temps entre le début et la fin de la charge.
  - Une fois Cx mesurée, celle-ci est déchargée par la broche RAZ.
  - La fonction delay(1ms) est utilisée comme base de temps de référence.

C.F expérimentation_4





- Equation pour déterminer C en connaissant T et R : $T = RxC \text{ --- } > C = \frac{T}{R}$

Exemple : T= 5ms et R=470kohms → C=10nF, Tmin= 1ms → Cmin=2nF

Convertisseur Analogique/Numérique

- **Exemple 2 : Réalisation d'un capacimètre.**

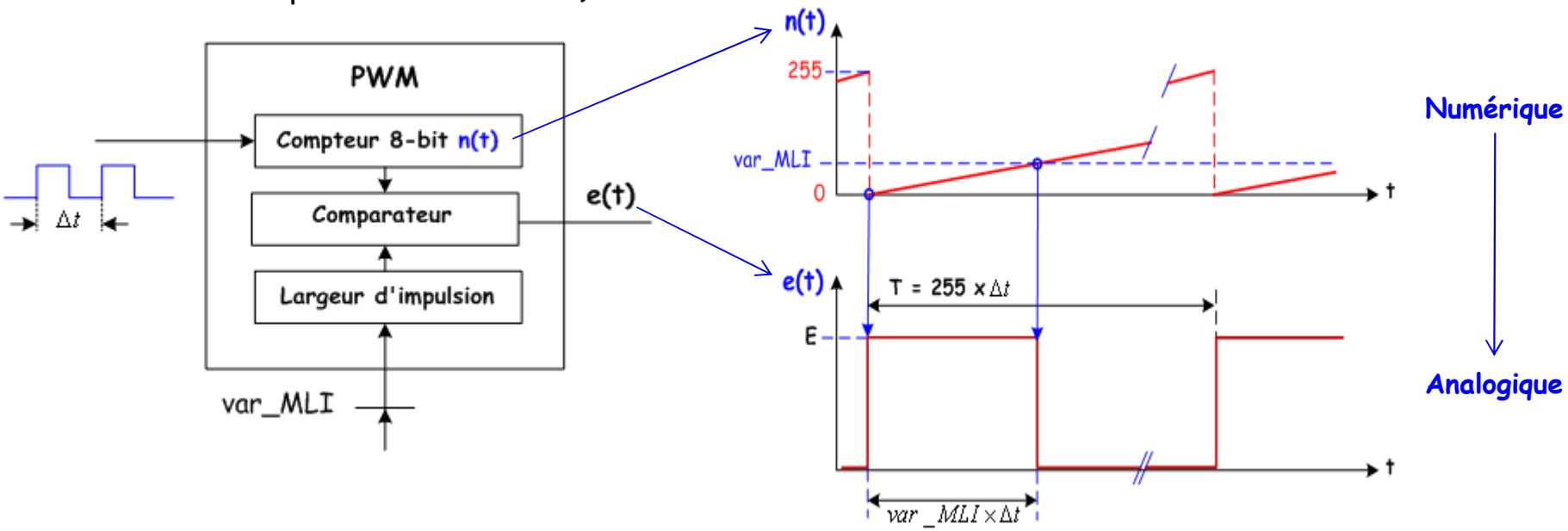
```
//--- Impulsion de charge de la capacité à mesurer
 int const Pulse  = A5;
//--- Entrée de mesure de la tension de charge
 int const Capa   = A4;
//--- Entrée/Sortie de décharge de la capacité
 int const RAZ    = A3;
//--- Résistance calibrée de charge (en kohms)
 int const R      = 460;
//--- Seuil de tension de charge pour T = RxC
 float const Seuil = 5*0.63;

void setup() {
 Serial.begin(9600);
//-- Pulse est une sortie
  pinMode(Pulse,OUTPUT);
//-- Pulse est mise à zéro au démarrage
 digitalWrite(Pulse,LOW);
//-- Capa est une entrée
 pinMode(Capa,INPUT);
//--- RAZ est une entrée au démarrage
 pinMode(RAZ,INPUT);
}
```

```
void loop() {
float capa;
float tc;
//--- Implusion de charge
 digitalWrite(Pulse,HIGH);
tc=0;
//--- boucle de mesure du temps de charge de la capa Cx correspondant à T = RC
while ((analogRead(Capa)*5.0/1023) < Seuil)
{
   delay(1) ;   ++tc;
}
//--- Calcul de Cx, t > 1 correspond à la valeur de Cx mimnimale.
if (tc>1) capa=tc*1000/R - Cpar;
else capa=0;
//--- Affichage de la valeur via le serial port
Serial.print(" Capa(nF) =")          ; Serial.println(capa);
Serial.print(" Temps de charge(ms) =") ; Serial.println(tc);
Serial.println(" ");
//--- RAZ de l'implusion
digitalWrite(Pulse,LOW);
//--- Décharge de Cx
pinMode(RAZ,OUTPUT) ; digitalWrite(RAZ,LOW);
Delay(1);
//--- Remise en entrée de RAZ
pinMode(RAZ,INPUT);
delay(1000);
}
```

Modulation par Largeur d'Impulsion (PWM)

- **Description et fonctionnement**
  - Signal périodique
    - Pin 5 : Période fixe standard T=490Hz
    - Pin 6 : Période fixe standard T=980Hz
    - La période peut est ajustée.

  - Impulsion dont la largeur peut avoir 256 valeurs différentes (0 → 255)
    - PWM 8-bits
    - Largeur de l'impulsion est fonction du traitement numérique (la largeur de l'impulsion est modulée).

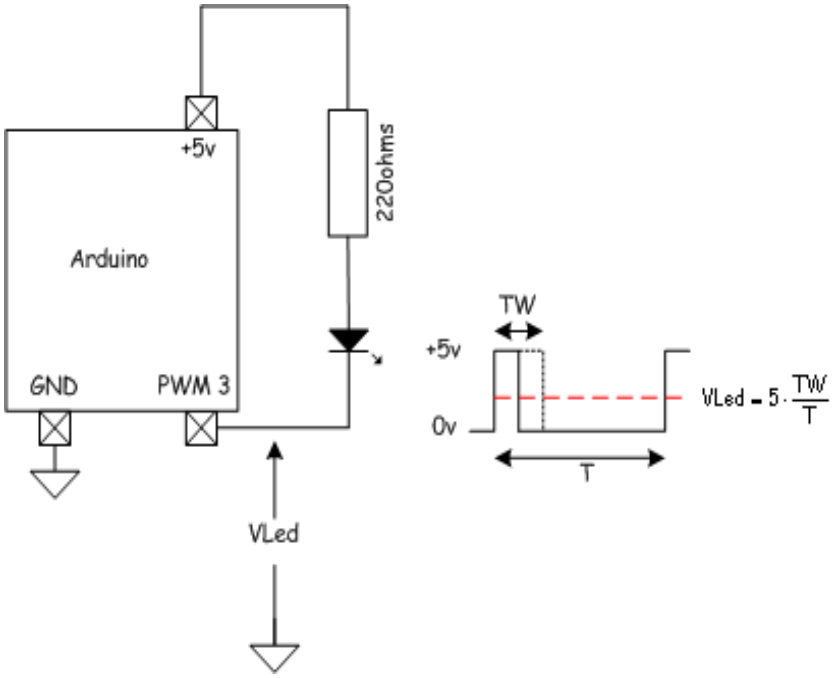


Modulation par Largeur d'Impulsion (PWM)

- **Contrôle de l'intensité lumineuse d'une diode LED.**
  - Exemple 1
    - L'intensité varie de 0 à 100% → var_MLI =[0.. 255] tous les 20ms.

    - La valeur moyenne de la tension de commande de la LED est de:

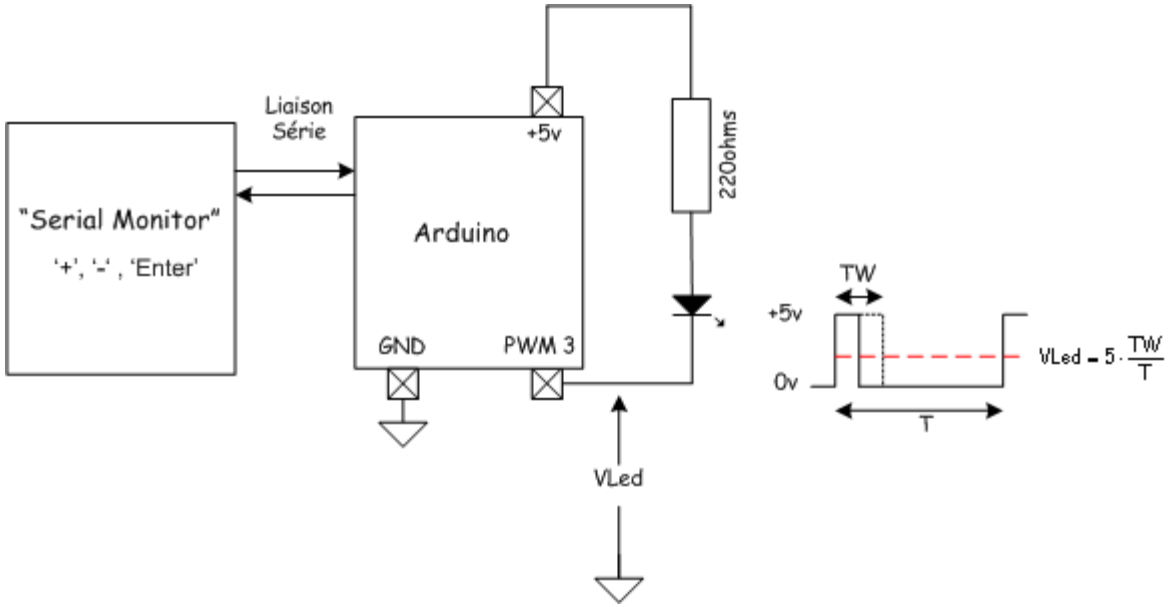
$$VLed = 5 \cdot \frac{TW}{T} = 5 \cdot \frac{var\_MLI}{255}$$




```
const byte Periode=20;
void setup() {
}

void loop() {
 byte var_MLI;

 for (var_MLI=0;var_MLI<256;var_MLI++)
 {
  //=======================================================
  //=== La largeur de l'impulsion varie de 0 à 255:
  //       PWM0       : est la sortie PWM sur laquelle
  //                    est branchée la LED.
  //       var_MLI    : est la variable contenant la
  //                    valeur de la largeur d'impulsion.
  //=======================================================
  analogWrite(PWM0,var_MLI);
  delay(Periode);
 }
}
```

Modulation par Largeur d'Impulsion (PWM)

- **Contrôle de l'intensité lumineuse d'une diode LED.**
  - Exemple 2
    - Contrôle de l'intensité lumineuse d'une diode LED à partir des touches du clavier '+' et '-', validées par l'appui de la touche "Enter".
      - Incêmente et décrêmente la largeur de l'impulsion à partir de la touche '+' et '-' du clavier (utilisation du "Serial Monitor").
      - Retard de 500ms entre deux changements.



Modulation par Largeur d'Impulsion (PWM)

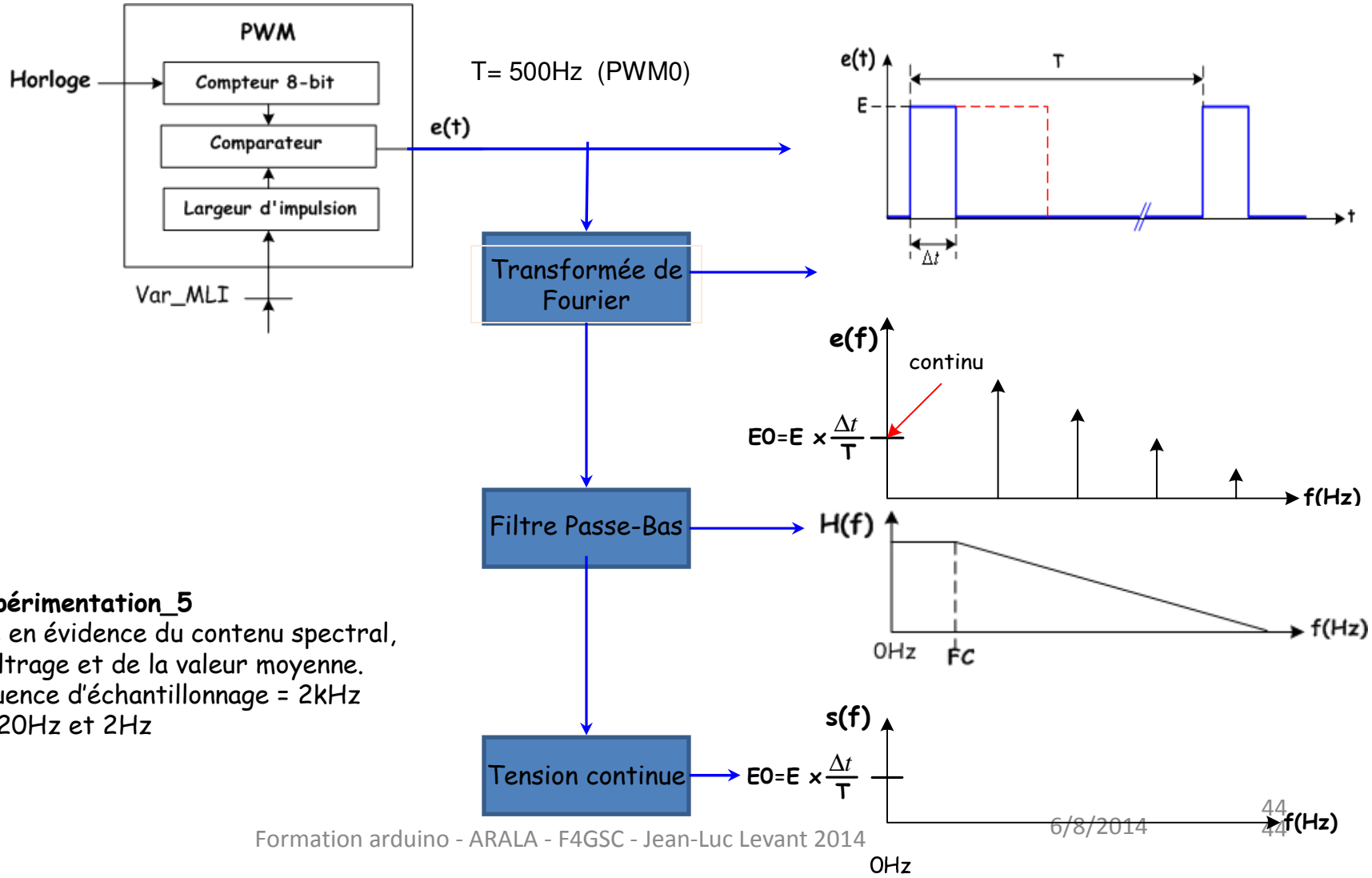
- **Contrôle de l'intensité lumineuse d'une diode LED.**
  - Exemple 2
    - Contrôle de l'intensité lumineuse d'une diode LED à partir des touches du clavier '+' et '-', validées par l'appui de la touche "Enter".

```
const int PWM0=3;
const byte Periode=20;
char IncDec;
byte var_MLI;

void setup() {
 Serial.begin(9600);
 //— Attention: 5v —> LED éteinte, 0v —> LED allumée.
 //— Eteint la LED
 var_MLI=255;
 analogWrite(PWM0,var_MLI);
}
```

```
void loop() {
//=============================================
 //   La largeur du pulse varie de 0 à 255
//=============================================
 //    PWM0 : est la sortie PWM sur laquelle est branchée la LED
 //         Attention,  5v —> LED éteinte, 0v —> LED allumée.
 //                              '+' décrémente , '-' incrémente.
//=============================================

//— si touche appuyée
 if (Serial.available() > 0)
  {
   IncDec=Serial.read();
   //— Touche '-'
   if (IncDec==45)
   {
    if (var_MLI+40<=255)
    {
     var_MLI=var_MLI+40 ; analogWrite(PWM0,var_MLI) ;
     Serial.print("Largeur Pulse : ");Serial.println(var_MLI);
    }
   }
   //— Touche '+'
   if (IncDec==43)
   {
    if (var_MLI-40>=0)
    {
     var_MLI=var_MLI-40 ; analogWrite(PWM0,var_MLI) ;
     Serial.print("Largeur Pulse : ");Serial.println(var_MLI);
    }
   }
   delay(500);
  }
}
```

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - Principe de fonctionnement



C.F expérimentation_5

- Mise en évidence du contenu spectral, du filtrage et de la valeur moyenne.
- Fréquence d'échantillonnage = 2kHz
- FC = 20Hz et 2Hz
- E=5v

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - Loi de conversion numérique → analogique

Compteur 8-bit : 0<= n <=255

- n = 1 → t = Δt
- n = k → $t = k \times \Delta t$
- n = 255 → t = T

Δt : est la résolution du PWM

$$E0 = E \times \frac{\Delta t}{T} = E \times \frac{n}{255}$$

	Numérique	Analogique	Exemple
Excursion Anlogique	$0 \le E0 \le E$	$0 \le E0 \le E$	$0 \le E0 \le 5v$
Excursion numérique	$0 \le n \le 255$	$0 \le E \times \frac{n}{255} \le E$	$0 \le E \times \frac{255}{255} \le 5v$
Résolution	1	$\frac{E}{255}$	$\frac{5}{255} = 19.6mV$

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - **Exemple 1** :Réalisation d'une rampe à une pente:
    - Fréquence = 500ms
    - Amplitude 5v
    - PWM sur la broche 5

```
const int PWM0=5;
const byte Periode=2;
byte var_MLI;

void setup() {
}

void loop()
  {
   for (var_MLI =0; var_MLI <255; var_MLI ++)
  {
    //-- Mise à jour du PWM avec la nouvelle valeur
     analogWrite(PWM0, var_MLI);
     delay(Periode);
  }
}
```

Modulation par Largeur d'Impulsion (PWM)

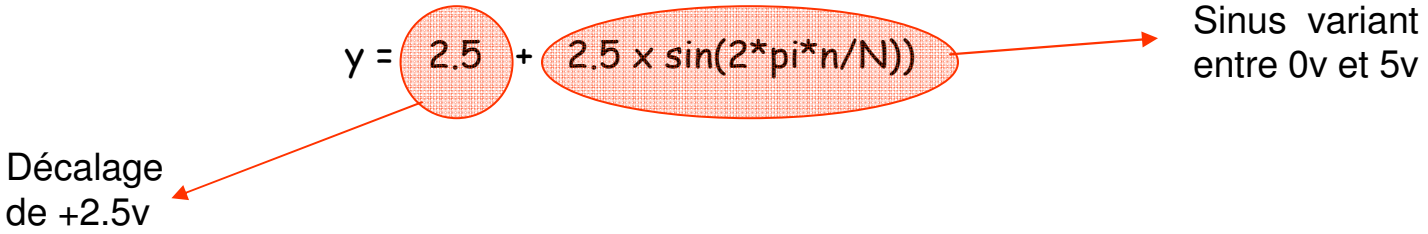
- **Convertisseur Numérique/Analogique**
  - **Exemple 2**: Réalisation d'une rampe à double pentes:
    - Fréquence = 500ms
    - Amplitude 5v
    - PWM sur la broche 5

```
const int PWM0=5;
const byte Periode=2;
void setup()
{
 byte var_MLI =0;
}

void loop()
{
for (var_MLI =0; var_MLI <255; var_MLI ++)
 {
  //-- Mise à jour du PWM avec la nouvelle valeur
    analogWrite(PWM0, var_MLI);
 //-- Fixe le pas de mise à jour du PWM
   delay(Periode);
 }
 for (var_MLI =255; var_MLI >0; var_MLI --)
 {
  analogWrite(PWM0, var_MLI);
  delay(Periode);
 }
}
```

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - **Exemple 3**: Réalisation d'un signal sinusoïdal
    - Fréquence = 500ms
    - Amplitude 2.5v centré sur 2.5v
    - Tension d'alimentation, VDD = 5 ;
    - Nombre de point dans la sinusoïde, N = 255;

    - Equation de la sinusoïde



    - Le PWM n'accepte que des nombres entiers (pas de nombre décimal)

m = partie entière (y*N/VDD)

C.F expérimentation_8

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - **Exemple 3**: Réalisation d'un signal sinusoïdal

```
#include <avr/pgmspace.h>
// --- Enregistrement des points définissant  une période de sinusoïde complète et  à fournir au PWM.
prog_uchar SinusTab[] PROGMEM=
{128,131,134,137,140,143,146,149,152,156,159,162,165,168,171,174,176,179,182,185,188,191,193,196,199,201,204,206,209,211,213,216,218,220,222,224,226,228,230,23
2,234,235,237,239,240,……37,39,42,44,46,49,51,54,56,59,62,64,67,70,73,76,79,81,84,87,90,93,96,99,103,106,109,112,115,118,121,124,127
};

// ------- Définition des constantes et  affection de la broche à la fonction PWM
const int PWM0     = 5 ; const byte Periode = 2;

// --- Programme d'initialisation → Ici aucune initialisation n'est requise.
void setup() {
}

// --- Programme proncipale qui s'éxécutera jusqu'à l'extinction de l'alimentation
void loop()
{
  // ---- Déclaration des variables du programme
  byte Pulse=0 ; byte k=0 ;  unsigned char sinus;

  //--- Boucle sans fin qui fournir les points de la sinusoïde au PWM.
 for (k=0;k<256;k++)
 {
   // ---- Acquisition d'unpoint de la sinusoïde depuis la tabke SinusTab
     sinus=pgm_read_byte_near(&SinusTab[k]);

   // ---- Chargement dans le PWM
    analogWrite(PWM0,sinus);

   // ---  Ajuste le délais entre deux points de la sinusoïde et de sa période ( T= 255 x délais)
   delay(Periode);
 }
}
```

Modulation par Largeur d'Impulsion (PWM)

- **Convertisseur Numérique/Analogique**
  - **Exemple 3**: Réalisation d'un signal sinusoïdal

    - Effet du filtre sur l'amplitude de la sinusoïde et le niveau des harmoniques

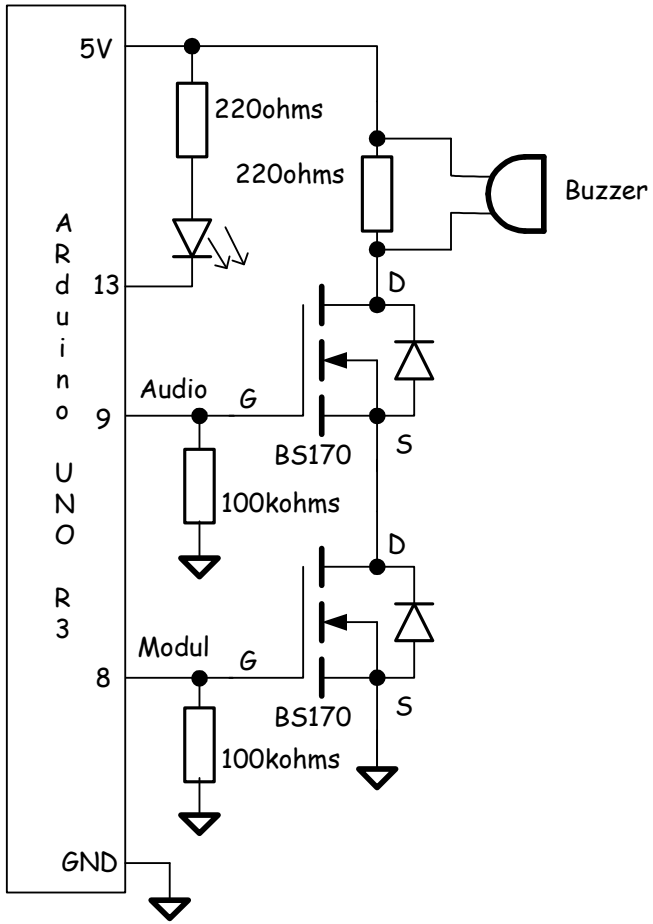
      - Pour une sinusoïde de fréquence de 2Hz:
        - » FC = 2Hz, le filtre apporte une atténuation de -3db soit 0.7 x l'amplitude maximale
        - » VM =0.7*2.5=1.8v.

        - » FC=20Hz est à une décade de la fréquence du signal l'amplitude du signal ne subit pas d'atténuation due au filtre par contre les harmoniques sont moins atténuées de 20dB par rapport à FC = 2Hz.

C.F expérimentation_2
- Mise en évidence du contenu spectrale, du filtrage et de la valeur moyenne.
- Fréquence d'échantillonnage = 2kHz
- FC = 20Hz et 2Hz
- E=5v

Modulation par Largeur d'Impulsion (PWM)

- **Modulation tout ou rien**
  - Générer un appel général en morse « CQ de F5KEQ «
    - Interface matérielle



Modulation par Largeur d'Impulsion (PWM)

- **Description et fonctionnement**
  - Exemple 4: Générer un appel général en morse en audio: CQ de F5KEQ

```
//---- 0 : 1 point, 1: un trait, 2: séparation entr deux lettres
//---- 3 : séparation entre deux mots.
//---- Message à trasmettre en morse :
//     C Q  de    F 5K Q
Byte Morse_Tab[]={1,0,1,0,2,1,1,0,1,3,1,0,0,
2,0,3,0,0,1,0,2,0,0,0,0,0,2,1,0,1,2,0,2,1,1,0,1,3};
//--- Nbre de symbole contenu dans le message
 const int  NbSymb   = 38;
//--- Vitesse du point
 int  point    = 110;
//--- Freq audio = 490hz
 int  Audio    = 9;
//--- Contrôle de la modulation
 int  Modul    = 8;
//---  Led
 int  Led      = 13;

void setup() {
 pinMode(Led,OUTPUT);
 analogWrite(Audio,127);
}
void loop() {
  byte code ;
  int i;
```

```
while(1){
 //--- Décodeur morse audio et visuel
  switch (Morse_Tab[i]){
   case 0: {
     digitalWrite(Led,HIGH) ;  digitalWrite(Modul,HIGH); delay(point);
     digitalWrite(Led,LOW) ;   digitalWrite(Modul,LOW); delay(point) ; break;
    }
   case 1: {
     digitalWrite(Led,HIGH); digitalWrite(Modul,HIGH); delay(2*point);
     digitalWrite(Led,LOW) ; digitalWrite(Modul,LOW) ; delay(point) ;  break;
   }
   case 2:{
     digitalWrite(Modul,LOW); digitalWrite(Led,LOW); delay(2*point) ; break;
   }
   case 3:{
     digitalWrite(Modul,LOW); digitalWrite(Led,LOW);delay(7*point) ; break;
   }
  };
  ++i;
  if (i>NbSymb)i=0;
 }
}
```

Les Liaisons Séries Synchrones et Asynchrones

Les Liaisons Séries Synchrones et Asynchrones

- Introduction
  - Les liaison séries qui permet la communication (échange de données, contrôle commande,…) des équipements informatique et des circuits intégrés.
  - Les données sont transmises sur un fil ou plusieurs fils suivants la nature de la transmission (UART, I2C,SPI,…).

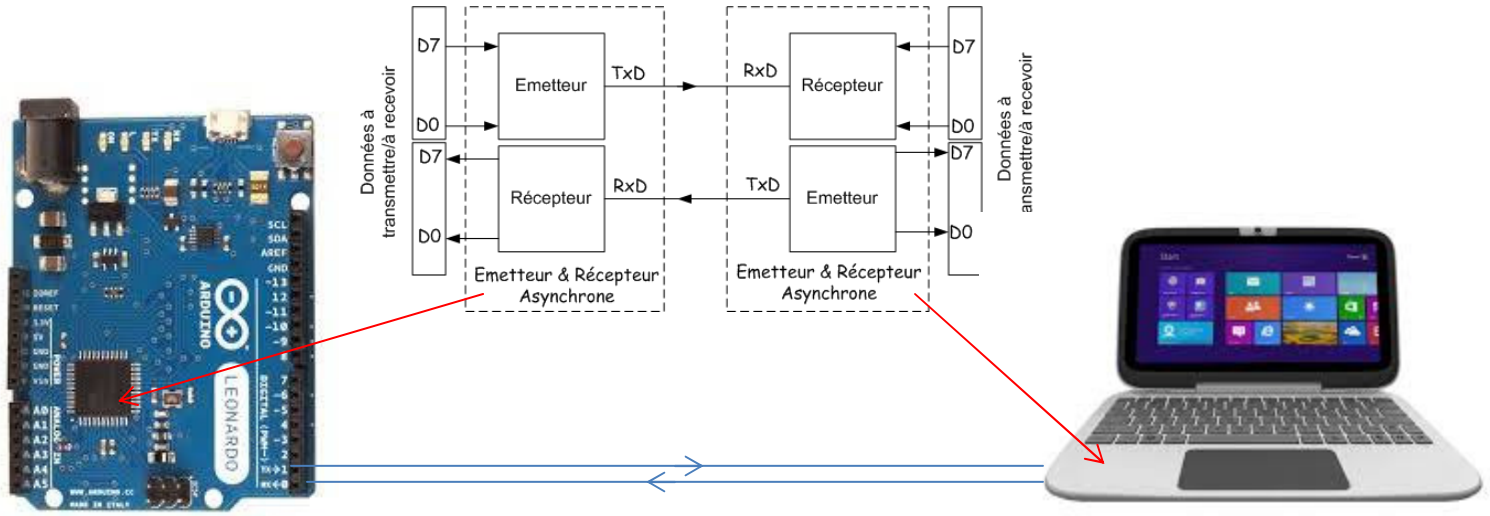
  - La communication série est gérée par un émetteur/récepteur intégré au niveau de l'équipement ou du circuits intégré (Atmega328p).

Les Liaisons Séries Synchrones et Asynchrones

- Introduction
  - La liaison peut être ASYNCHRONE si l'HORLOGE n'est pas transmise avec les données.
  - La liaison est SYNCHRONE si l'Horloge est physiquement transmises en même temps que le flux de données.
  - Les liaisons séries se caractérisent par leur protocole et leurs spécifications électriques.

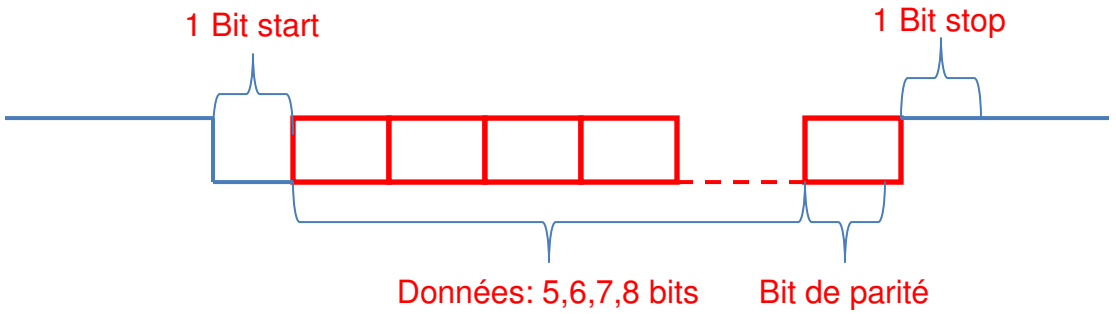
Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Structure
    - Un émetteur / récepteur transmet en série (1 fil) les 8-bits de la donnée.



Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Protocole
    - Décrit la structure de l'informations transmise.



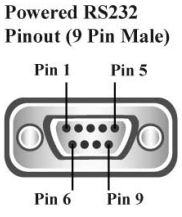
  - Bit Start : début de communication et de synchronisation du récepteur.
  - Bit Stop : fin de communication, longueur du bit = 1, 1.5 ou 2.
  - Bit Parité : permet de détecter des erreurs de transmission basé sur le nombre de bit codé à 1 dans la donnée:
    - Bit de Parité = 1 si le nombre est impaire
    - Bit de Parité = 0 si le nombre est impaire.
  - Bits de données : information utile.

Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Le standard RS232
    - Regroupe plusieurs autres standards :
      - les recommandations UIT-T **V.24** (définition des circuits).
      - **V.28** : caractéristiques électriques
        - » Tension :
          - CMOS et TTL pour des longueurs de connexions que dizaines de centimètres.
          - 3 à 35v pour les longueurs de plusieurs mètres (+/-12v en standard).
        - » débit (bauds, nombre de symbole transmis par seconde):
          - 300,600,1200,2400,4800,9600,19200,...
      - norme **ISO 2110** pour la connectique :
        - » DB9, DB25 et RJ25.

Powered RS232

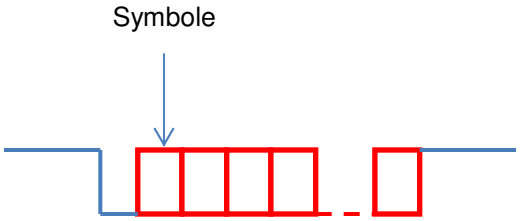
Pin 1	DCD/12V/GND
Pin 2	RXD
Pin 3	TXD
Pin 4	DTR
Pin 5	GND
Pin 6	DSR
Pin 7	RTS
Pin 8	CTS
Pin 9	RI/12V/5V



Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Le standard RS232
    - Le standard ne supporte
      - Par contre le standard ne définit pas :
      - Le codage des caractères (ASCII, Code Baudot ou EBCDIC par exemple).
      - La façon dont les caractères sont répartis en trames.
      - Les protocoles de détection d'erreur ou les algorithmes de compression de données.
      - Les débits de transmission : seule une limite supérieure de 20 000 bauds est indiquée.
      - La fourniture de courant électrique à des périphériques.

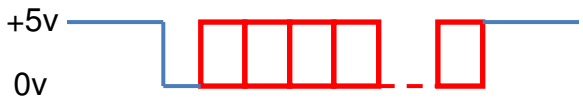
  - Baud
    - Nombre de symbole transmis par seconde
      - 2400 bit/s, en 4-QAM (2 bits transmis par symbole)
      - 3600 bit/s, en 8-QAM (3 bits transmis par symbole)
      - 4800 bit/s, en 16-QAM (4 bits transmis par symbole)

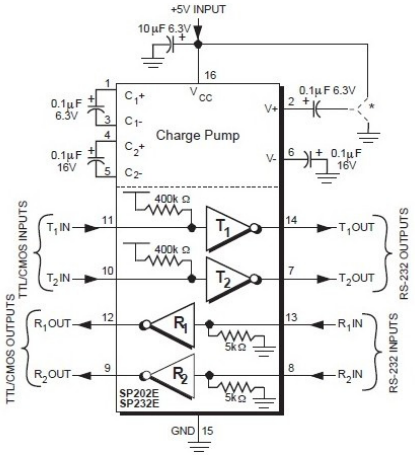


Les Liaisons Séries Synchrones et Asynchrones

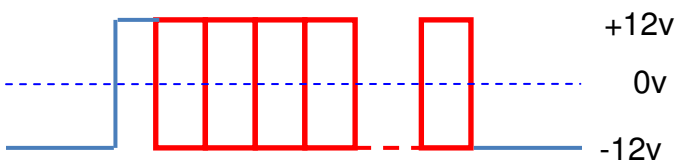
- La liaison série ASYNCHRONE
  - Le standard RS232
    - Les interfaces électriques +/-12 v

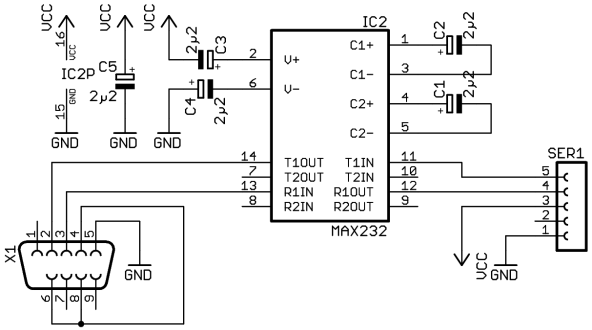
Côté Equipement et carte





Câble de liaison





Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Le standard RS232
    - Comment programmer

```
//=====================================================
//-- Envoie un caractère via la liaison série
//=====================================================
void setup()
 {
 //--- Création d'un objet liaison série Asynchrone

  Serial.begin(9600);

 }


 void loop()
{
//=====================================================
//--- Début d'une transmission
//=====================================================
 while (!Serial);
 Serial.println('U');
 delay(50);
}
```

```
//=====================================================
//-- Reçoit un caractère via la liaison série
//=====================================================
 byte incomingByte;

void setup()

 {
 //--- Création d'un objet liaison série Asynchrone
  Serial.begin(9600);
 }


 void loop()
{
//=====================================================
//--- Attend l'arrivée d'un caractère
//=====================================================
  if (Serial.available() > 0) {
      //--- le caractère est reçu
        incomingByte = Serial.read();
     //--- Envoit vers le terminal série
        Serial.print("Caractère reçu ") ;  Serial.println(incomingByte, DEC);
        }
}
```

Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Le standard RS232
    - Table ASCII (American Standard Code for Information Interchange).

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Le standard RS232
    - Table ASCII (American Standard Code for Information Interchange).

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

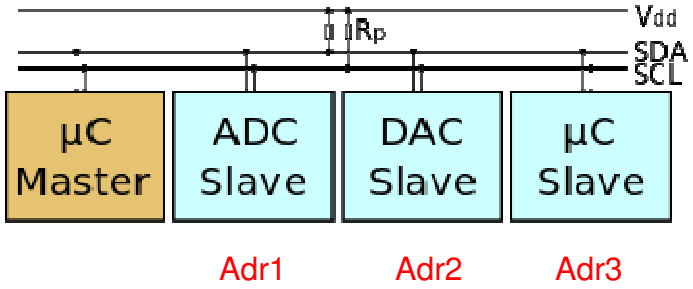
Source: www.LookupTables.com

Les Liaisons Séries Synchrones et Asynchrones

- La liaison série ASYNCHRONE
  - Exemple : Pilotage Tx

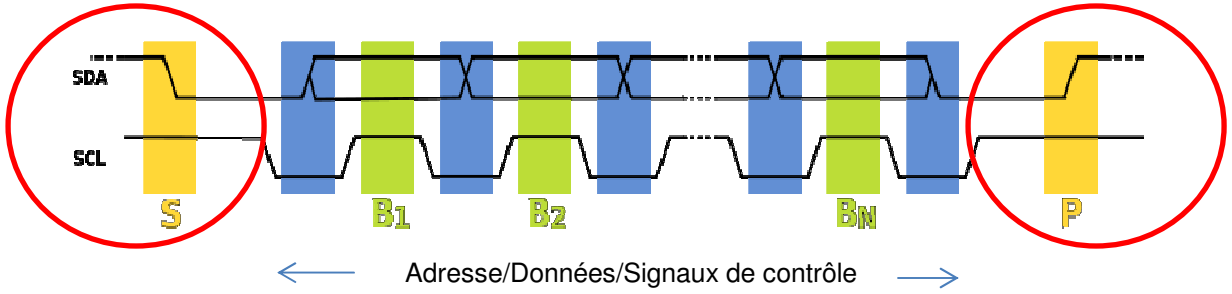
Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Bus de liaison série synchrone de communication entre un ou plusieurs circuits intégrés (ADC, IO, DAC, horloge temps réelle, mémoire…). Chacun des CI est repéré par une adresse Adrn.



- L'horloge est transmise en même temps que les données (registre à décalage).
  - Ligne SDA : Serial Data line
  - Ligne SCL : Serial Clock line

Condition de début de transfert



Condition de Fin de transfert

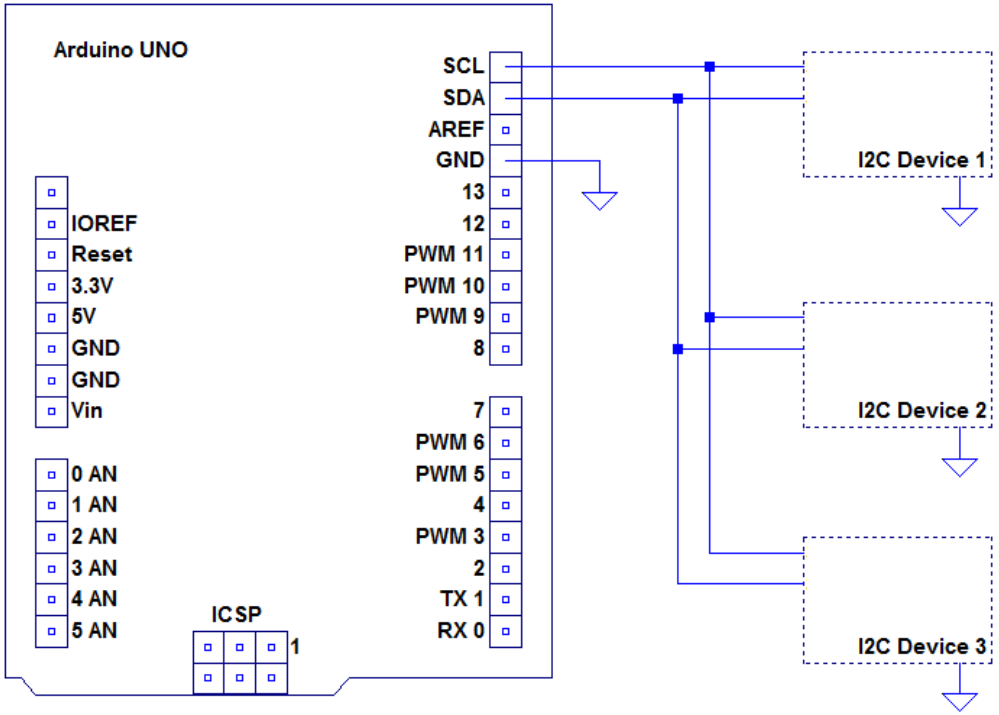
Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Vitesse de transmission

Mode	Vitesse (kBit/s)	TR	Max. Cload (pF)	Rp
Standard	100	1µs	400	2700
Fast	400	300ns	400	820
Fast +	1	120ns	550	270
High-speed	3.4			
Ultra-fast	5			

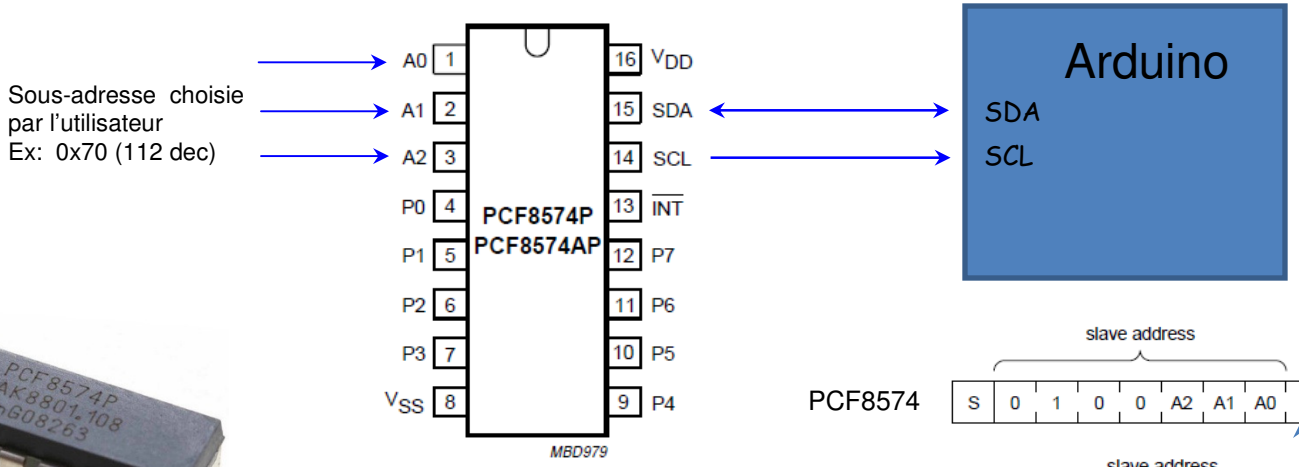
Les Liaisons Séries Synchrones et Asynchrones

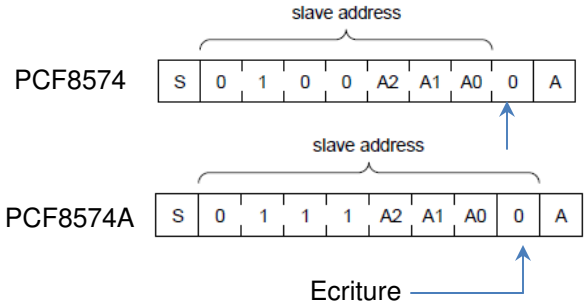
- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - I2C et Aarduino



Les Liaisons Séries Synchrones et Asynchrones

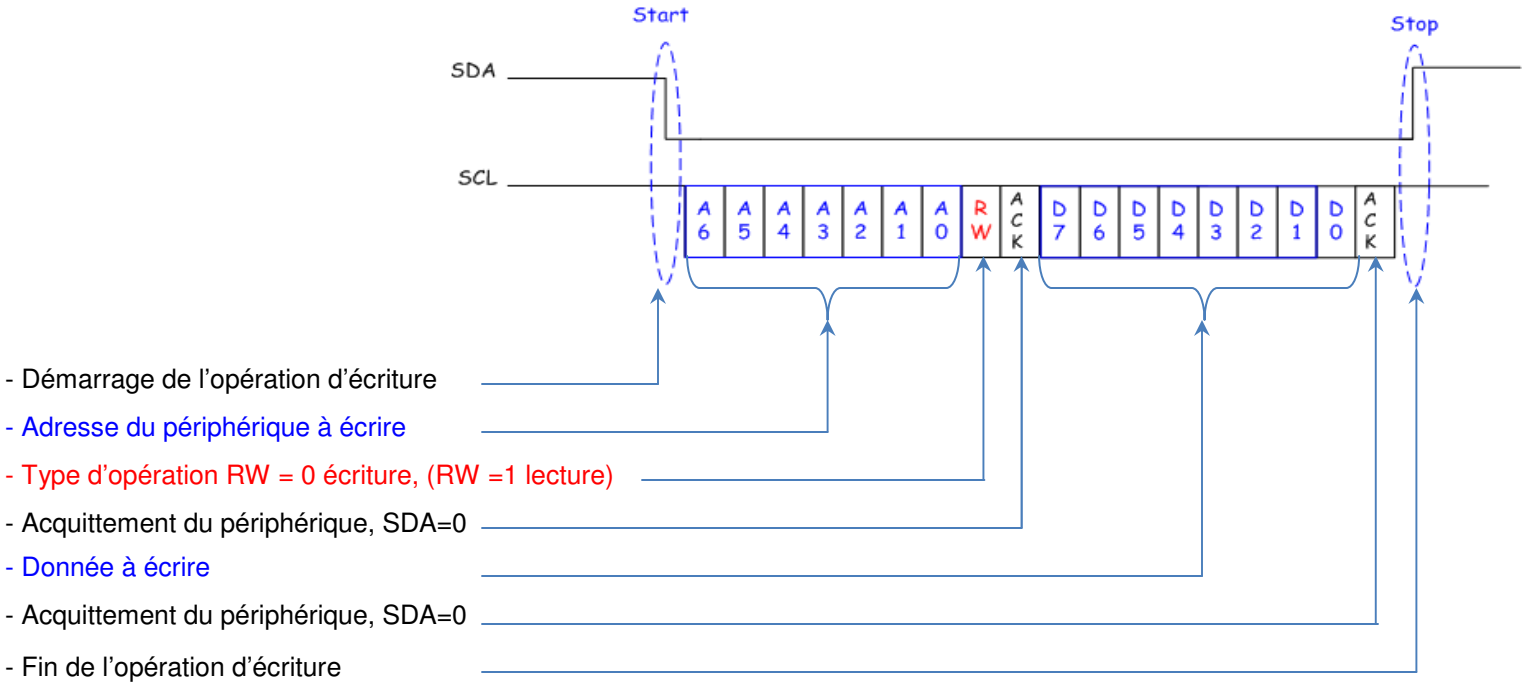
- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Dépend du type d'opération réalisée (cette formation n'envisage que les cas les plus standards).
      - L'écriture d'un octet dans un périphérique (ex: PCF8574A)





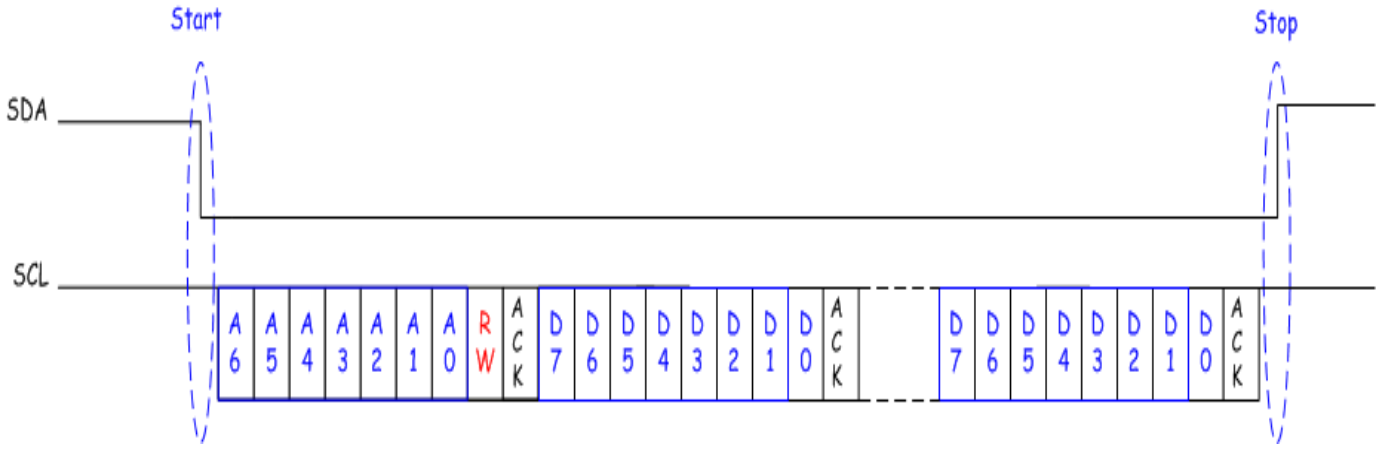
Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C
      - Cas d'une écriture simple



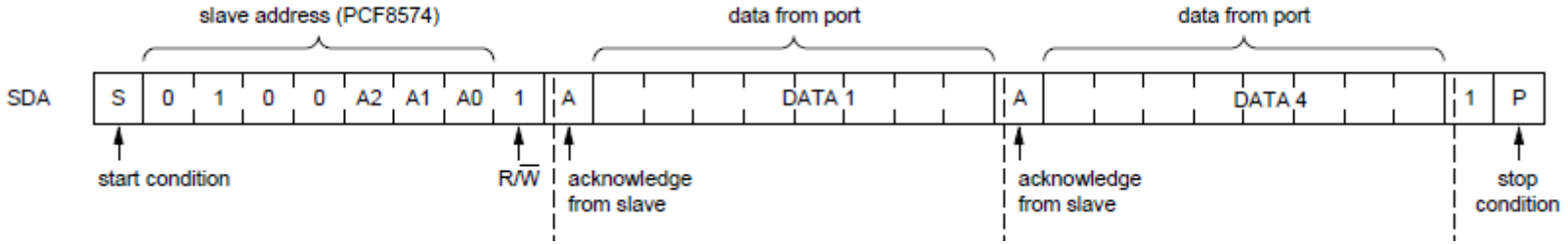
Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Ecriture en rafale (nombre d'écriture illimité)



Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Lecture (nombre d'écriture illimité)



Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Ecriture dans une mémoire (E^2PROM, RAM…)

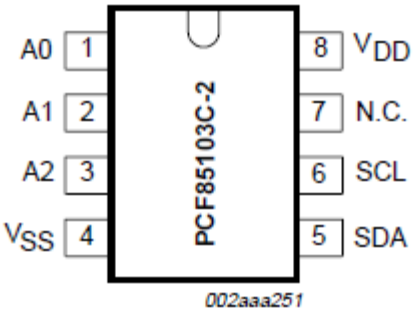
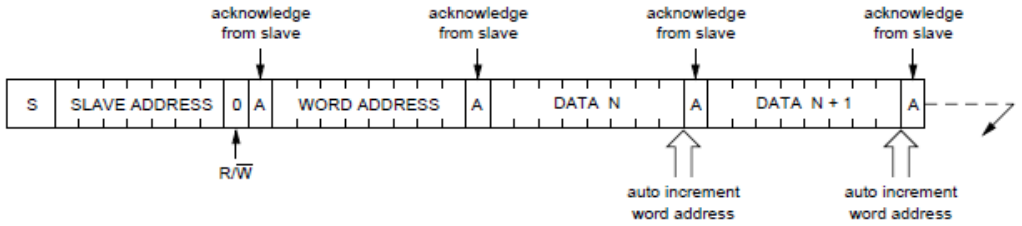
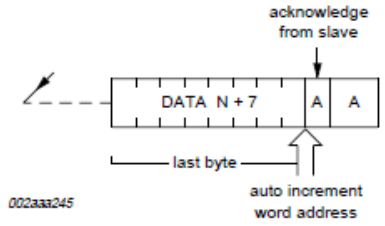


Table 5: Device address code

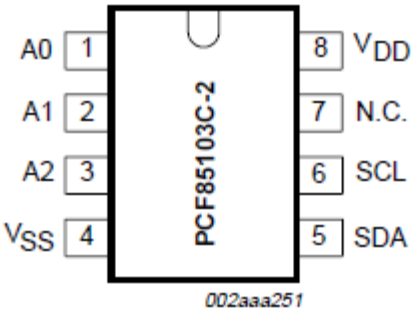
Selection	Device code				Chip Enable			R/$\overline{W}$
Bit	b7[1]	b6	b5	b4	b3	b2	b1	b0
Device	0	0	1	0	A2	A1	A0	R/$\overline{W}$

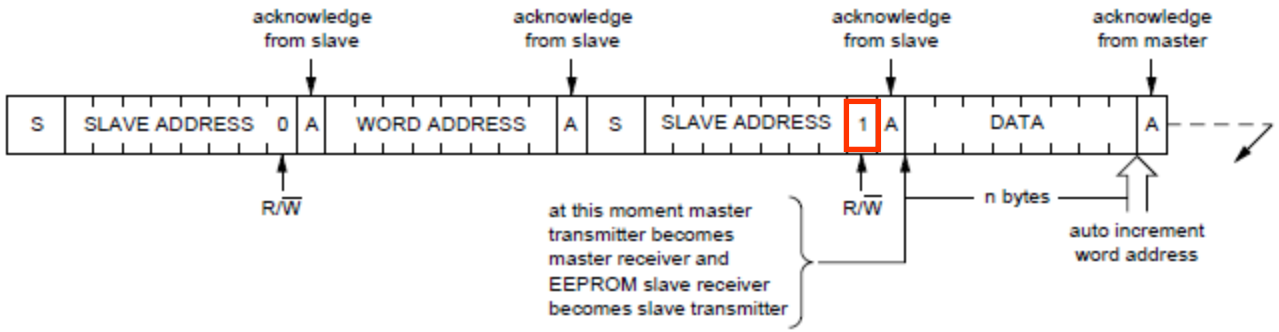


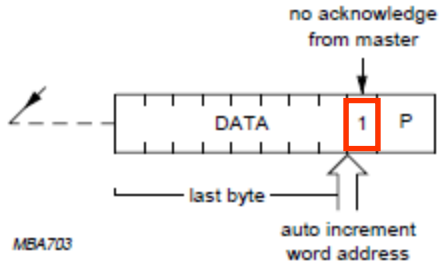


Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Lecture dans une mémoire (E²PROM, RAM…)

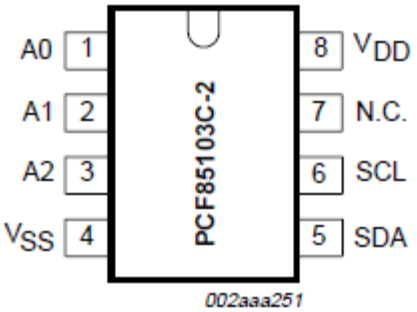


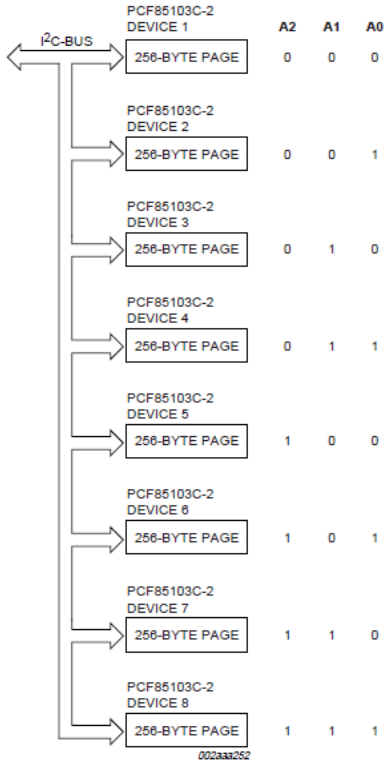




Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Adressage de plus mémoire PCF85103C





Les Liaisons Séries Synchrones et Asynchrones

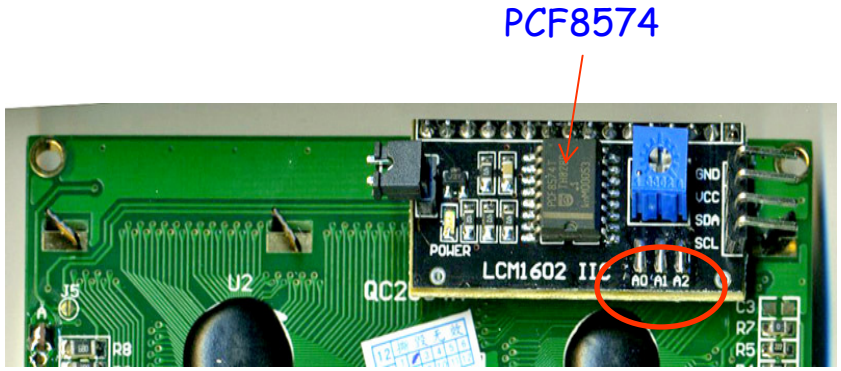
- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Programmation d'un composant I2C

```
//=====================================================
//--- envoie de 64 caractères
//=====================================================
#include <Wire.h>

byte val = 0;

void setup()
{
 //--- Création d'un objet I2C
  Wire.begin();
}

void loop()
{
 //=====================================================
 //--- Début d'une transmission par l'envoie de l'adresse du périphérique
//=====================================================
  Wire.beginTransmission(44);

//=====================================================
//-- envoie de la valeur
//=====================================================
  Wire.write(val);

//=====================================================
//--- Fin de transmission
//=====================================================
  Wire.endTransmission();

  val++;
  if(val == 64)
  {
    val = 0;
  }
  delay(500);
}
```

```
//=====================================================
//-- Transmission vers la liaison série des caractères reçus par la liaison I2C.
//=====================================================
#include <Wire.h>

void setup()
{
 //--- Création d'un objet I2C
  Wire.begin();
//--- Création d'un objet liaison série
  Serial.begin(9600);
}

void loop()
{
 //=====================================================
 //--- Demande de lecture de 6 octets du périphérique d'adresse 0x02
//=====================================================
  Wire.requestFrom(2, 6);
//=====================================================
//--- Attente de la réception d'un caractère
//=====================================================
  while(Wire.available())
  {
   //=====================================================
   //--- Lecture d'un caractère reçu depuis la liaison I2C et envoi vers
   //--- la liaison série
   //=====================================================
    char c = Wire.read();
    Serial.print(c);
  }
  delay(500);
}
```

Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Afficheur LCD 4 x 20 caractères



Les Liaisons Séries Synchrones et Asynchrones

- La liaision série synchrone
  - I2C : Inter Integrated Circuit bus
    - Structure d'une trame I2C.
      - Afficheur LCD

```
#include <Wire.h>

//============================================================
// LCD librairie
// https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
// Configure les connections entre l'afficheur LCD et l'interface I2C
//                  addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
// Adresse I2C = 0x27
//============================================================

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup()
{
  //--- Création d'un objet liaison série
   Serial.begin(9600);
  //--- Création d'un objet LCD comportant 4 lignes de 20 caractères
   lcd.begin(20,4);
  //--- Positionne le curseur sur la première ligne à gauche
   lcd.setCursor(0,0);
  //--- Affiche "Formation ARALA"
   lcd.print("Formation ARALA");
  //--- Positionne le curseur sur la deuxième ligne à gauche
   lcd.setCursor(0,1);
  //--- Affiche "Mai 2014"
   lcd.print
}
```

```
void loop()
{
  {
   //--- Attent l'arrivée d'un caractère
    if (Serial.available()) {
   //--- Tempo pour pour être sur la transmission série est terminée
      delay(100);
   //--- RAZ de l'affichage
      lcd.clear();
   //--- Lecture de tous les caractères reçus et affichage sur le lCD
      while (Serial.available() > 0) {
   //--- Affichage sur le LCD
        lcd.write(Serial.read());
      }
    }
  }
}
```

Les Interruptions

Les Interruptions

- A quoi cela sert?
  - Analogie
    - C'est bien connu:
      - Les Hommes ne savent gérer qu'une seule action à la fois.
      - Les Femmes savent gérer plusieurs actions en même temps.

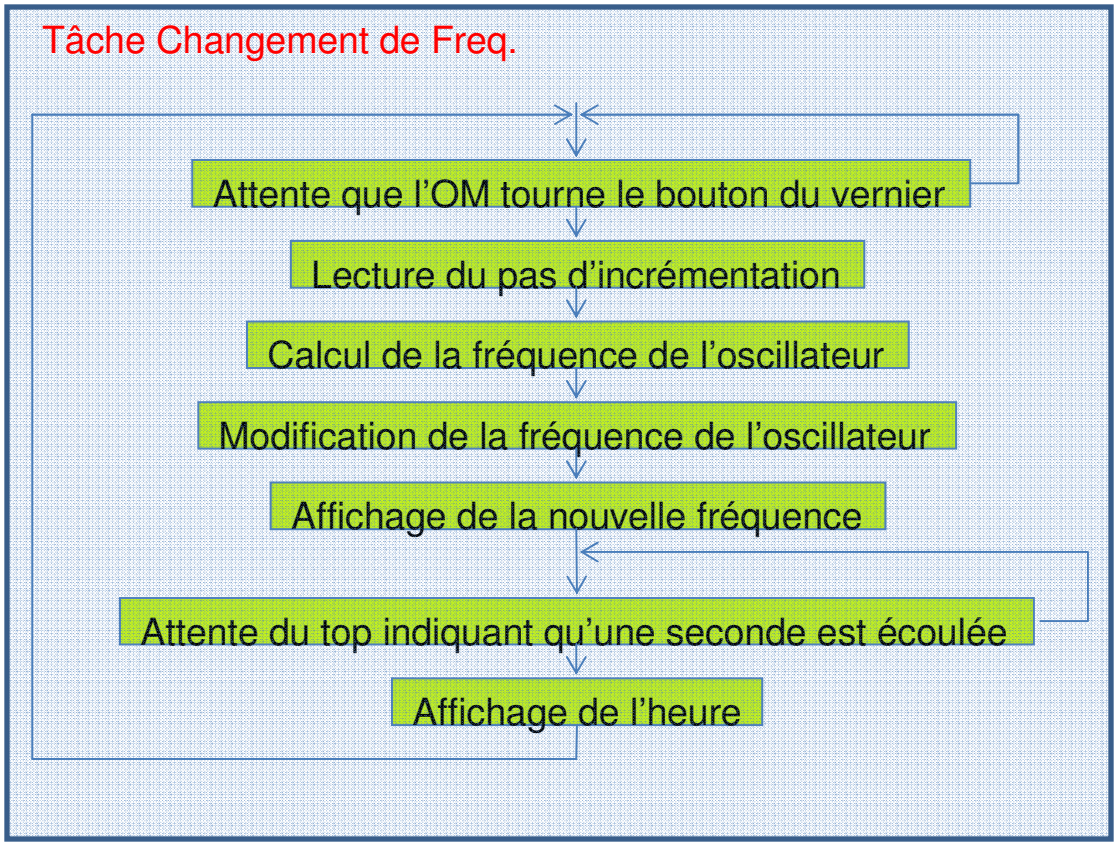
      - Les hommes sont mono tâche et les femmes multitâches.
  - Définition du principe du mécanisme d'interruption
    - Le mécanisme des interruptions permet au microcontrôleur de gérer plusieurs actions quasi en même temps.

    - Ce microcontrôleur n'ayant qu'une unité d'exécution les différentes actions sont réalisées l'unes après l'autres.

    - La notion de « quasi en même temps » n'est vraie que si le temps pris pour gérer les différentes tâches sont compatibles avec la vitesse d'exécution du microcontrôleur.

Les Interruptions

- Exemple de programme nono-tâche
  - Changement la fréquence du transceiver.
    - Le microcontrôleur répète à l'infini la tâche ci-dessous constituée de plusieurs actions.



Attente par scrutation logicielle d'une demande d'action du type :

while !(bouton);

Tant que le bouton n'est pas activé alors attendre.

Attente par scrutation logicielle d'une demande d'action du type :

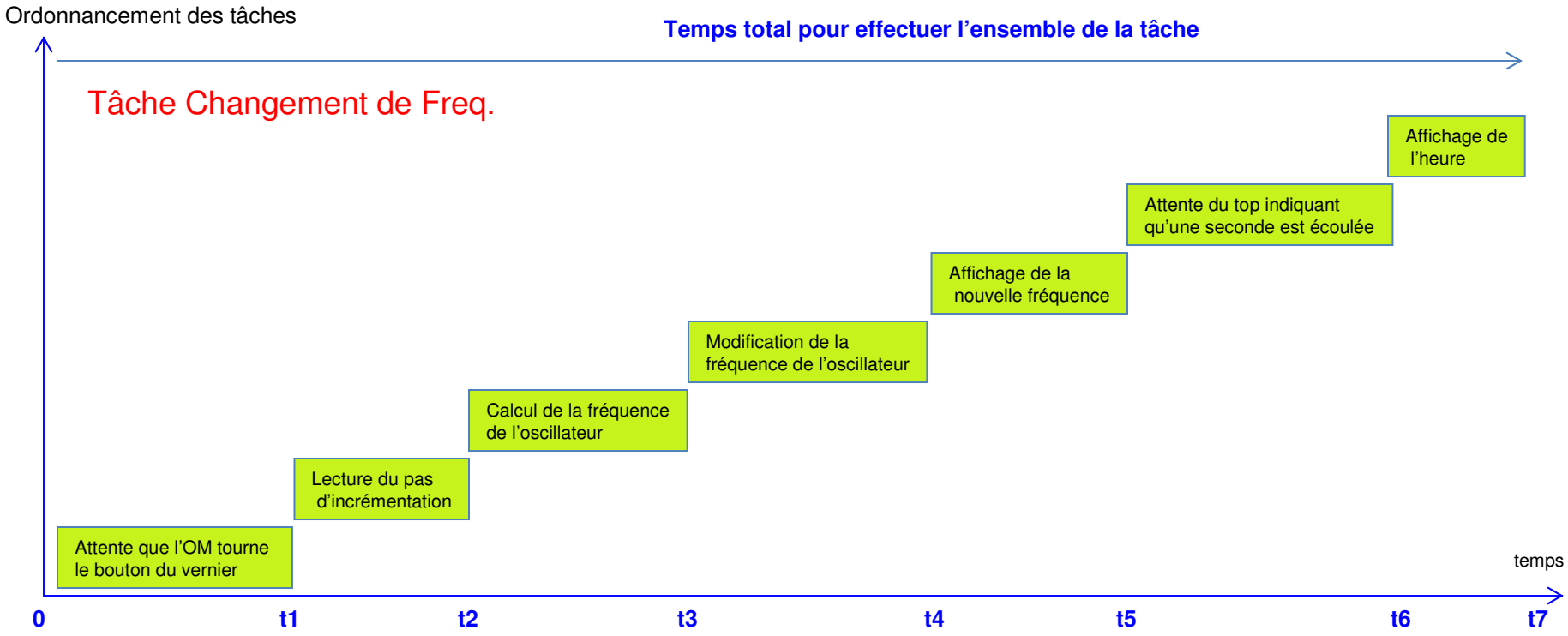
While !(seconde);

Tant que la seconde n'est pas écoulée alors attendre

Le microcontrôleur gère plusieurs actions mais séquentiellement l'une après l'autre.

Les Interruptions

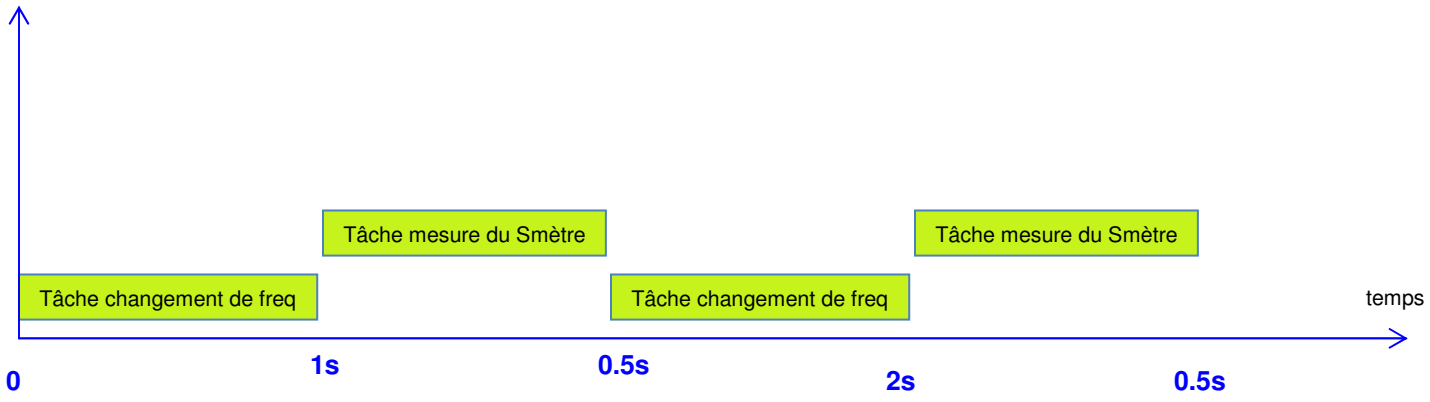
- Exemple de programme nono-tâche
  - Changement la fréquence du transceiver.



L'action d'attente force le processeur à surveiller si l'évènement est arrivé:
- Il ne peut rien faire d'autre.

Les Interruptions

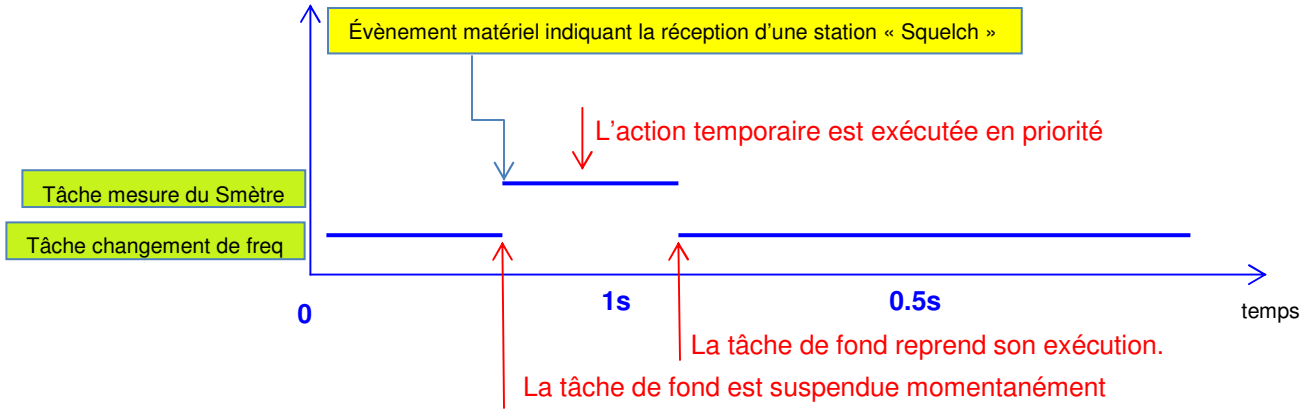
- Exemple de programme multitâche
  - Supposons que le récepteur détecte le déclenchement d'un relais (R7) et qu'il soit nécessaire de mesurer le niveau du signal reçu pour l'afficher.
  - Supposons que la tâche « Tache changement de freq » prend 1 seconde.
  - Il faut donc attendre 1 seconde avant d'aller mesurer la force du signal reçu et l'afficher.



Les Interruptions

- Exemple de programme multitâche
  - Supposons que la tâche « mesure du Smètre » soit la tâche prioritaire et qu'elle doit être exécutée sans délai.

  - D'un point de vue chronogramme cela ressemble à ceci:



- La tâche « Tâche de changement de freq » est une TACHE DE FOND qui est exécutée à l'infini tant que le microcontrôleur n'a rien d'autre à faire.

- La tâche « Tâche de mesure du Smètre » est une TACHE ou ACTION TEMPORAIRE considérée comme étant plus prioritaire que la tâche de fond. Elle est déclenchée par un évènement matériel (ici le « squelch »).

Les Interruptions

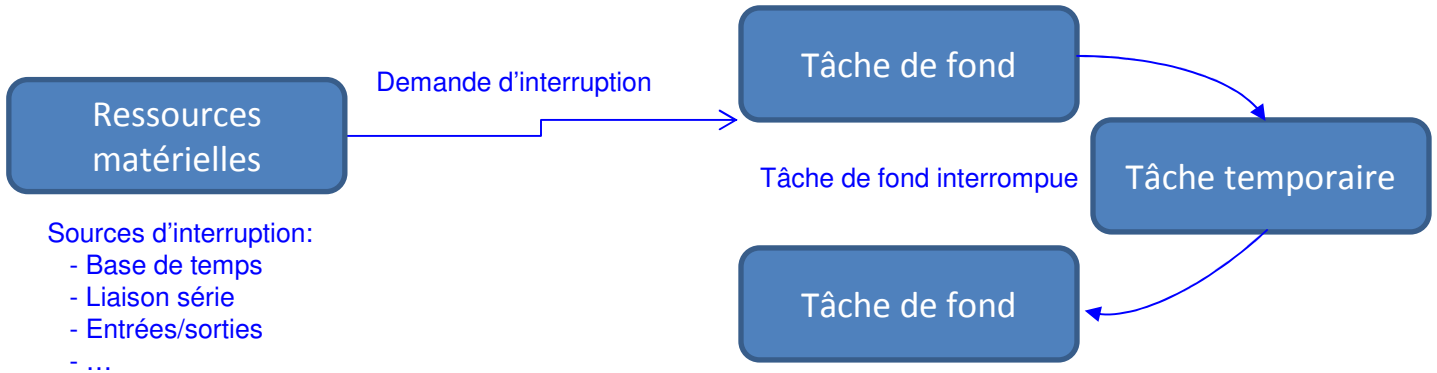
- Exemple de programme multitâche

  - Bien qu'il n'y ait qu'une seule Unité de calcul, il y a bien deux tâches différentes qui sont exécutées et gérées par le matériel → multitâche.

  - La synchronisation est matérielle et non pas logicielle (technique de la scrutation logicielle « polling » qui consomme du temps CPU).

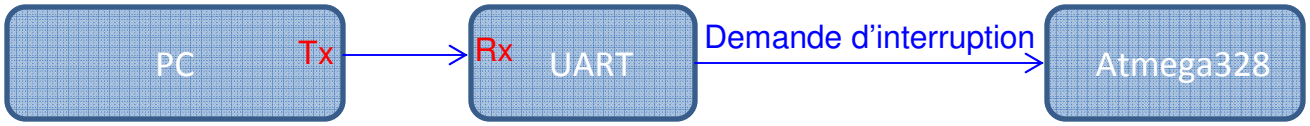
  - Le mode d'ordonnancement des tâches par interruption est une technique de synchronisation matérielle utilisant les ressources périphériques du microcontrôleur ou sources d'interruption.

  - Une ressource périphérique demande à la CPU d'interrompre temporairement la tâche de fond pour lui permettre d'exécuter son traitement.

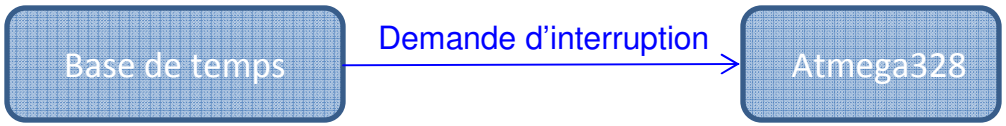


Les Interruptions

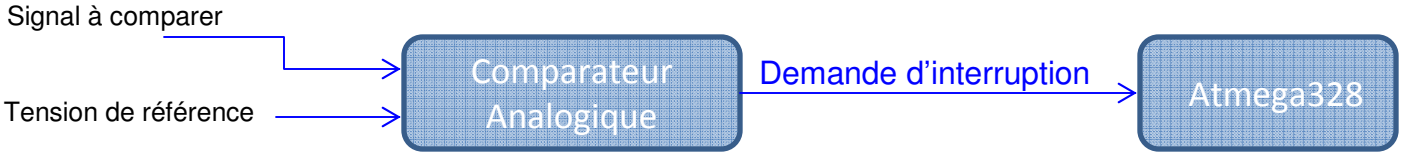
- Exemple de programme multitâche
  - Exemples:
    - UART, liaison série asynchrone
      - Une interruption temporaire de la tâche de fond est demandée par l'UART et indique qu'un caractère a été reçu ou que le caractère courant a été transmis.



    - Base de temps
      - Une interruption temporaire de la tâche de fond est demandée par la base de temps pour indiquer que le temps de référence est écoulé (ex: 1 seconde).



    - Comparateur analogique
      - Une interruption temporaire est demandée par le comparateur analogique pour indiquer que le signal d'entrée à dépasser la tension de comparaison.



Les Interruptions

- Les sources d'interruption de l' ATmega328
  - 24 sources d'interruption :

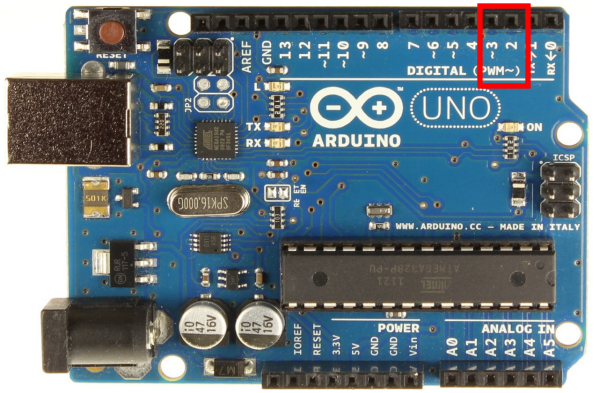
    - **Interruptions externes**:
      - Interruptions liées aux entrées INT0 (PD2) et INT1 (PD3).

    - **Interruptions internes**
      - Base de temps:
        - » Interruptions liées aux Timers 0, 1 et 2 (plusieurs causes configurables).

      - Fonctions analogiques:
        - » Interruption liée au comparateur analogique.
        - » Interruption de fin de conversion ADC.

      - Communication série:
        - » Interruptions du port série USART.
        - » Interruption du bus TWI (I2C) et SPI.

Les Interruptions



- Comment utiliser les interruptions externes avec l'Arduino Uno?
  - Du point de vue logiciel
    - Autoriser une ou les sources d'interruption à interrompre la tâche de fond.
    - Indiquer le mode de déclenchement de la demande d'interruption, exemple activation sur front ou niveau.
    - Décrire l'algorithme de traitement lié à la source d'interruption

Initialise et définie la source d'interruption utilisée.

```
int Squelch = 0; // Interruption externe 0 câblée sur la pin 2 et a comme numéro 0.
void setup()
    {
    .
    .
    //-- L'action temporaire "StationRecue" est exécutée pour chaque détection de squelch (front montant (→
    //-- "RISING".) sur la pin 2.

    attachInterrupt(SQUELCH, StationRecue, RSISING);

    }
```

Mode de déclenchement front ou niveau

Nom de l'action temporaire

Numéro de la Source d'interruption

Les Interruptions

- Comment utiliser les interruptions externes 0 et 1 (pin 2 and 3)?

  – Du point de vue logiciel

    - La fonction attachInterrupt(param1, param2, param3) nécessite trois paramètres :

      • **param1** : Quelle interruption est utilisée.
      - Attention, c'est le numéro de l'interruption et pas le numéro de la pin:
      - Interruption externe 0 → 0 , Interruption externe 1 → 1;

      • **param2** : Le nom de la fonction /algorithme de traitement.
      - Il n'y a pas de paramètre à passer ou à retourner par cette fonction.

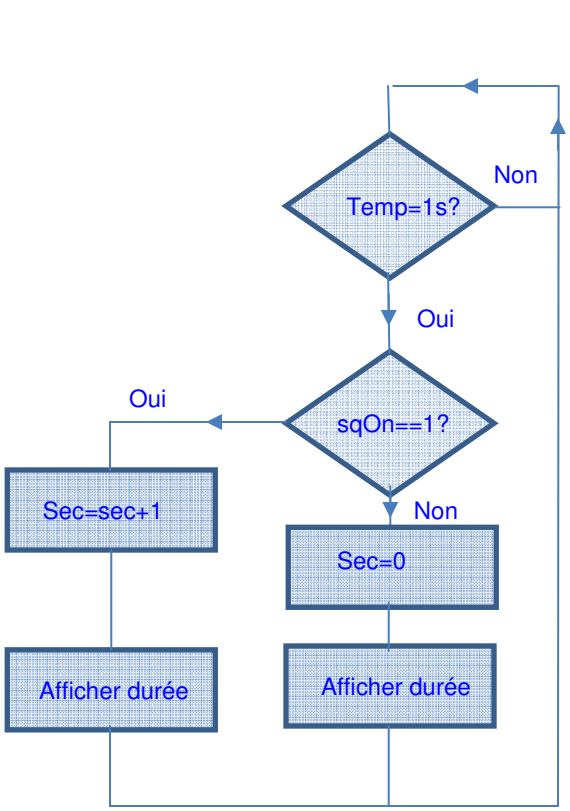
      • **param3** : Quel est le type de déclenchement de l'interruption.

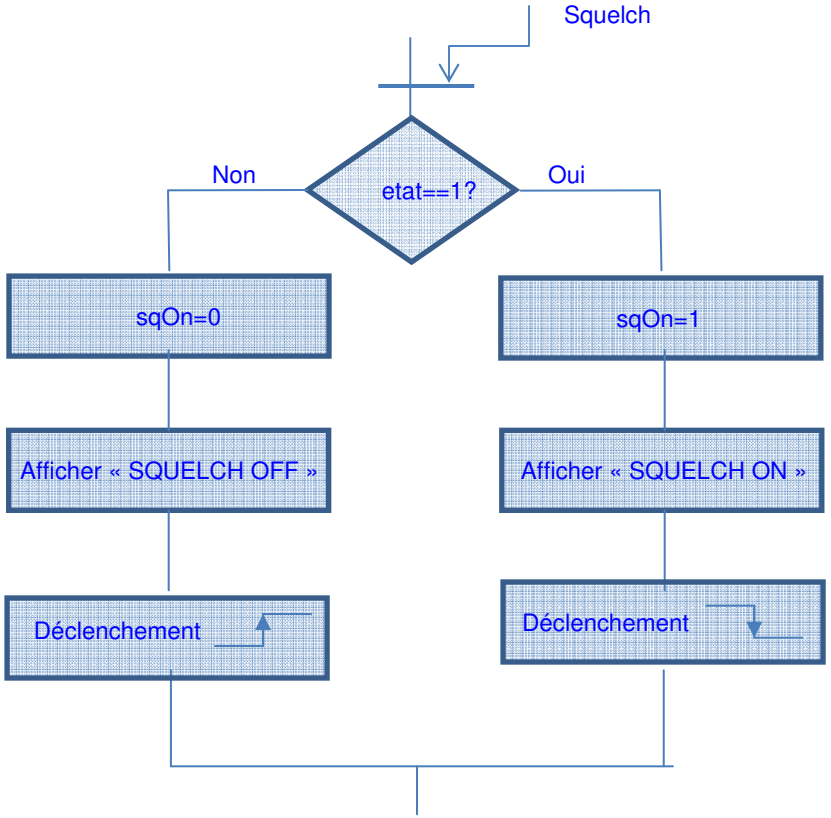
    - Il y a quatre types de déclenchements possible :

      • LOW : L'entrée est au niveau bas
      • RISING : L'entrée passe d'un niveau bas à un niveau haut
      • FALLING : L'entrée passe d'un niveau haut à un niveau bas
      • CHANGE : L'entrée a changé de niveau. Elle est passé d'un niveau bas vers un niveau haut ou inversement.

Les Interruptions

- Comment utiliser les interruptions externes 0 et 1 (pin 2 and 3)?
  - Exemple:
    - Mesurer le temps d'ouverture du squelch d'un récepteur en utilisant l'interruption externe 0 (pin 2).
    - Déclenchement de la mesure de la durée sur le front montant et arrêt sur le front descendant.





Les Interruptions

- Comment utiliser les interruptions externes (pin 2 and 3)?
  - Exemple:

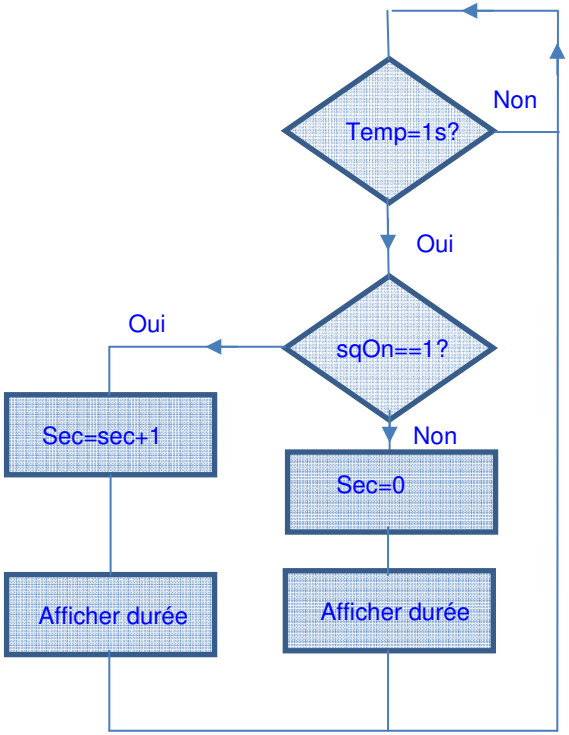
```
#include "LiquidCrystal.h"
LiquidCrystal lcd(8,9,4,5,6,7);

int squelchIn     = 0;
volatile int etat = LOW;
int sqOn;
int sec;

void setup()
{
  lcd.begin(16,2);
  lcd.clear();
  lcd.setCursor(0,0);lcd.print("Interruption ex ");
  attachInterrupt(squelchIn,Squelch,RISING);
}
```

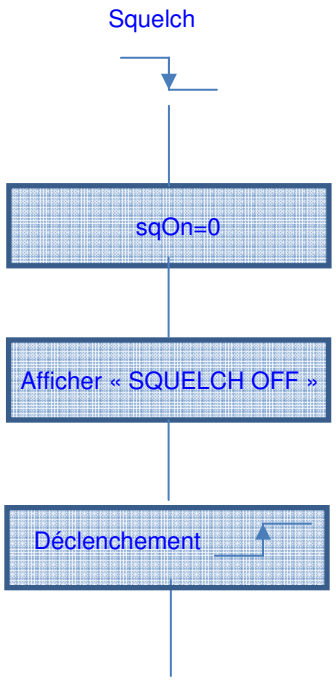
```
void Squelch()
{
  etat=!etat;

  if (etat)
  {
    lcd.setCursor(0,0);lcd.print("             ");
    attachInterrupt(squelchIn,Squelch,RISING);
    lcd.setCursor(0,0);lcd.print("SQUELCH ON ");
    sqOn=1;
    }
  else
  {
     attachInterrupt(squelchIn,Squelch,FALLING);
     lcd.setCursor(0,0);lcd.print("SQUELCH OFF ");
     sqOn=0;
  }
}
```

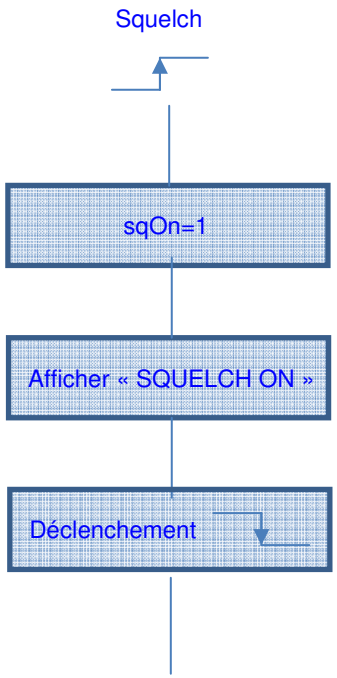
```
void loop()
{
 delay(1000);
 if (sqOn)
 {
  if (sec==60) sec=0;

 sec=sec++;
 lcd.setCursor(0,1);lcd.print("Time= ");
 lcd.setCursor(6,1);lcd.print("       ");
 lcd.setCursor(6,1);lcd.print(sec,DEC);
 }
 else
 {
   sec=0;
   lcd.setCursor(0,1);lcd.print("Time= ");
   lcd.setCursor(6,1);lcd.print(sec,DEC);
 }
}
```

Les Interruptions

- Comment utiliser les interruptions externes (pin 2 and 3)?
  - Exemple:
    - Mesurer le temps d'ouverture du squelch d'un récepteur en utilisant l'interruption externe 0 (pin 2).
    - Déclenchement de la mesure de la durée sur le front montant et arrêt sur le front descendant.
    - Deuxième solution en associant une tâche temporaire pour le front descendant et une autre pour le front montant.







Les Interruptions

- Comment utiliser les interruptions externes 0 et 1 (pin 2 and 3)?
  - Exemple:

```
#include "LiquidCrystal.h"
LiquidCrystal lcd(8,9,4,5,6,7);

int squelchIn = 0;
//-- Numéro de de l'interruption externe:
// 0--> pin 2 , 1--> pin 3

int sqOn   ; //-- Indique si une station est présente.
int sec    ; //-- Variable indiquant la durée

void setup()
{
 Serial.begin(9600);
 lcd.begin(16,2);
 lcd.clear();
 lcd.setCursor(0,0);lcd.print("Interruption ex ");
 attachInterrupt(squelchIn,SquelchOn,RISING);
}
```

```
void SquelchOff()
{
 lcd.setCursor(0,0);lcd.print("SQUELCH OFF    ");
 sqOn=0;
 attachInterrupt(squelchIn,SquelchOn,RISING);
}

void SquelchOn()
{
 lcd.setCursor(0,0);lcd.print("SQUELCH ON     ");
 sqOn=1;
 attachInterrupt(squelchIn,SquelchOff,FALLING);
}
```

```
void loop()
{
 delay(1000);
 if (sqOn)
 {
  lcd.setCursor(0,1);lcd.print("Time= ");
  lcd.setCursor(6,1);lcd.print("        ");
  lcd.setCursor(6,1);lcd.print(sec,DEC);
  sec=sec++;
  if (sec>=60) sec=0;
 }
 else
 {
  sec=0;
  lcd.setCursor(0,1);lcd.print("Time= ");
  lcd.setCursor(6,1);lcd.print(sec,DEC);
 }
}
```

Les Interruptions

- Comment utiliser les interruptions externes 0 et 1 (pin 2 and 3)?
  - Du point de vue logiciel
    - Autres fonctions
      - noInterrupts() : Les demandes d'interruption ne sont pas autorisées et ne seront pas prises en compte.
      - interrupts() : Toutes les demandes d'interruption sont autorisées à condition que les interruptions utilisées soient autorisées individuellement.
      - detachInterrupt(numéro_interruption): La demande d'nterruption "numéro_interruption" n'est plus prise en compte.