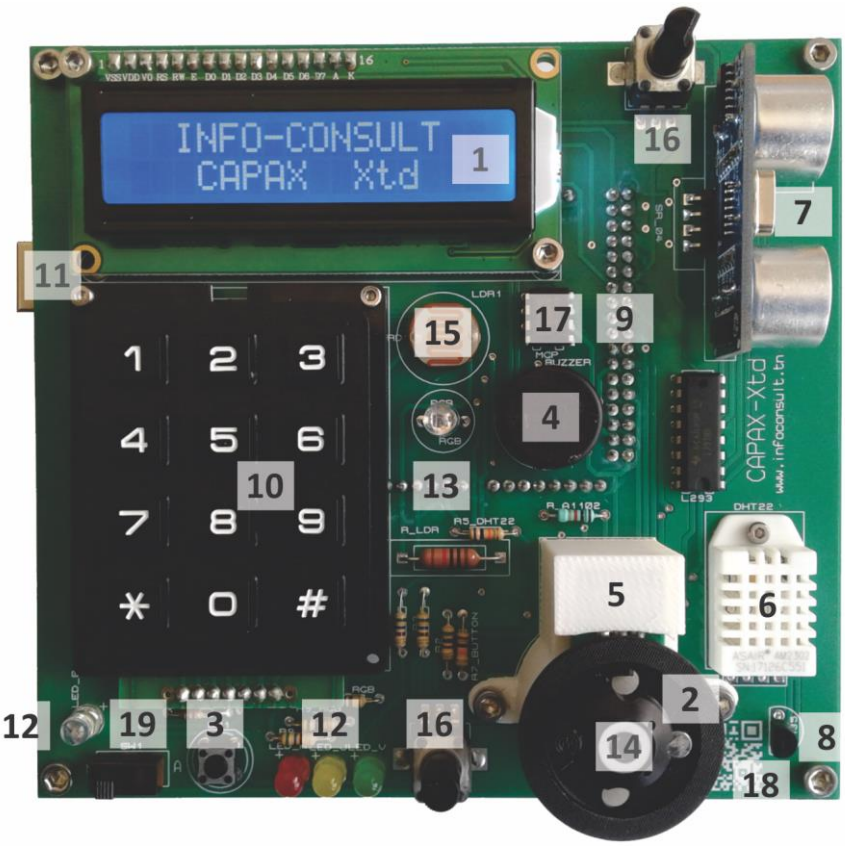
CAPAX-Xtd® pour les débutants

Plan

- Présentation de la carte CAPAX-Xtd®
- Correspondance des capteurs-actionneurs sur la carte ARDUINO MEGA
- Présentation de l'IDE ARDUINO.
- Sortie numérique.
- Sortie analogique.
- Entrée numérique.
- Entrée analogique.
- Affichage sur l'écran LCD.
- Lecture d'une touche du clavier.

1. Présentation de la carte CAPAX-Xtd®



1. Afficheur LCD 16*2
2. Aimant
3. Bouton poussoir
4. Buzzer
5. Capteur à effet HALL
6. Capteur DHT22
7. Capteur distance US
8. Capteur LM35
9. Carte ARDUINO MEGA
10. Clavier 12 Touches
11. Connecteur USB
12. Diodes LED
13. Diode RGB
14. Moteur MCC
15. Photorésistance LDR
16. Pot. analogique
17. Pot. numérique (SPI)
18. QR code
19. Sélecteur

Fig.1 : Nomenclature des composants de la carte CAPAX-Xtd®

2. Correspondance des capteurs-actionneurs sur la carte ARDUINO MEGA

Réf. carte CAPAX-Xtd®	Légende	Broche (Arduino)	E/S	Type
16	**Potentiomètre analogique**	A4	Entrée	Analogique
14	**Photorésistance LDR**	A3	Entrée	Analogique
8	**Capteur LM35**	A15	Entrée	Analogique
6	**Capteur DHT 22**	A6	Entrée	Numérique
3	**Bouton poussoir**	D47	Entrée	Numérique
10	**Clavier 12 Touches 4*3** (Colonne 1)	D37	Entrée	Numérique
	Clavier 12 Touches 4*3 (Colonne 2)	D35	Entrée	Numérique
	Clavier 12 Touches 4*3 (Colonne 3)	D41	Entrée	Numérique
	Clavier 12 Touches 4*3 (Ligne 1)	D36	Entrée	Numérique
	Clavier 12 Touches 4*3 (Ligne 2)	D39	Entrée	Numérique
	Clavier 12 Touches 4*3 (Ligne 3)	D40	Entrée	Numérique
	Clavier 12 Touches 4*3 (Ligne 4)	D38	Entrée	Numérique
5	**Capteur à effet Hall**	D18	Entrée	Numérique (INT)
12	**Diode RGB** (Bleu)	D11	Sortie	Numérique
	Diode RGB (Vert)	D12	Sortie	Numérique
	Diode RGB (Rouge)	D8	Sortie	Numérique
13	**Diode LED** Verte	D42	Sortie	Numérique
	Diode LED Jaune	D44	Sortie	Numérique
	Diode LED Rouge	D43	Sortie	Numérique
1	**Afficheur LCD 16*2** (RS)	D34	Sortie	Numérique
	Afficheur LCD 16*2 (E)	D33	Sortie	Numérique
	Afficheur LCD 16*2 (D1)	D32	Sortie	Numérique
	Afficheur LCD 16*2 (D2)	D31	Sortie	Numérique
	Afficheur LCD 16*2 (D3)	D30	Sortie	Numérique
	Afficheur LCD 16*2 (D4)	D29	Sortie	Numérique
17	**Potentiomètre numérique (SPI)** (CS)	D53	Sortie	Numérique
	Potentiomètre numérique (SPI) (SI)	D51	Sortie	Numérique
	Potentiomètre numérique (SPI) (SCK)	D52	Sortie	Numérique
12	**Diode LED blanche**	D9	Sortie	Analogique
4	**Buzzer**	D10	Sortie	Analogique
7	**Capteur de distance US** (TRIG)	D23	Sortie	Numérique
	Capteur de distance US (ECHO)	D25	Entrée	Numérique
14	**Moteur MCC** (En) : vitesse de rotation	D2	Sortie	Analogique
	Moteur MCC (In1) : sens de rotation*	D3	Sortie	Numérique
	Moteur MCC (In2) : sens de rotation*	D5	Sortie	Numérique

* Le sens de rotation du moteur est défini selon le tableau suivant :

In1	In2	Sens de rotation
HIGH	LOW	Horaire
LOW	HIGH	Trigonométrique

3. Structure de la programmation sur l'IDE ARDUINO

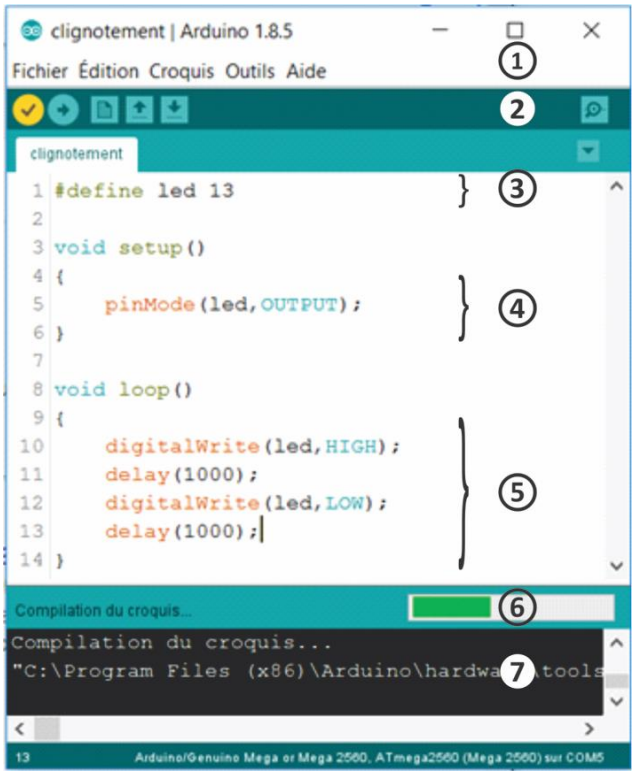
3.1 Généralités sur l'IDE ARDUINO

L'IDE Arduino a pour fonctions principales **:**

- Ecrire et compiler des programmes pour la carte Arduino.
- Se connecter avec la carte Arduino pour y transférer les programmes.

Cet environnement de développement intégré (EDI) dédié au langage `Arduino` et à la programmation des cartes Arduino comporte :

- Une `barre de menu`.
- Une `barre des boutons` qui donne un accès direct aux fonctions essentielles du logiciel.
- Un `éditeur` pour écrire le code du programme.
- Une `zone de message` qui affiche l'état des actions en cours.
- Une `console texte` qui affiche les messages concernant le résultat de la compilation du programme.



Légende :

1. Barre de menu.
2. Barre des boutons.
3. Zone de déclaration des constantes, des variables et des libraires.
4. Zone `setup()`.
5. Zone `loop()`.
6 Zone d'affichage des actions en cours.
7. Zone d'affichage des messages de compilation.

Fig.2 : Structure d'un programme ARDUINO

La structure d'un programme Arduino se compose de 3 zones :

- Zone de déclaration des différentes variables, des constantes ou encore des librairies qui seront utilisées dans la suite du programme.
- Zone `setup()` : permet de configurer les paramètres qui seront utilisées dans la suite du programme tel que l'initialisation des variables, définition des entrées et des sorties. Cette fonction n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino.
- Zone `loop()` : contient les instructions sur lesquelles le programme va boucler en permanence et delà contrôler activement la carte Arduino.

Remarque : Les fonctions `setup()` et `loop()` même vide sont obligatoires.

L'organigramme suivant illustre le fonctionnement d'un programme Arduino :

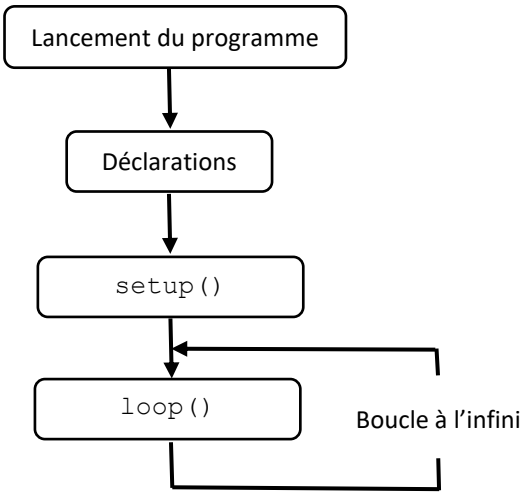
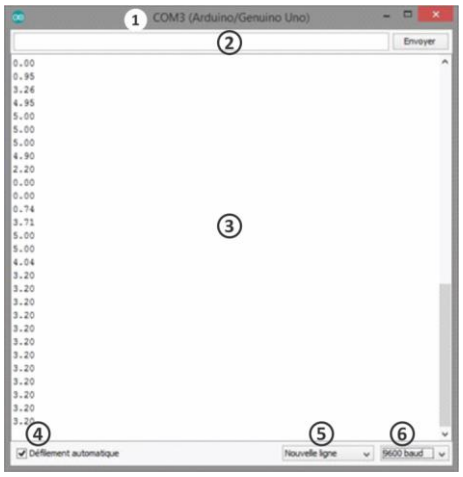


Fig.3 : Principe de fonctionnement de la structure de base d'un programme Arduino

3.2 Exploitation du moniteur série

Le moniteur série est exploitable pour la lecture ou l'écriture des données quand la carte Arduino est liée au port série USB. Pour ouvrir le moniteur série, cliquer sur l'icône de la loupe :
La fenêtre du moniteur série se présente comme suit :



Légende :

1. Le titre de la fenêtre qui indique le port de connexion et le type de la carte Arduino.
2. Zone de saisie ainsi qu'un bouton "`envoyer`" pour l'envoie des données vers la carte Arduino.
3. Zone d'affichage des données reçues par le moniteur à travers les fonctions suivantes : `Serial.print()` ou `Serial.println()`.
4. Une case à cocher pour arrêter le défilement des informations affichées.
5. Un menu déroulant qui permet le réglage du mode de défilement des informations via une liste déroulante de choix.
6. Un menu déroulant qui permet de choisir la vitesse de transmission des données du moniteur série de 300 à 2500000 bauds.

Fig.4 : Moniteur série

4. Sortie numérique

4.1 Allumer une LED

4.1.1 Créer un nouveau projet

Pour créer un nouveau programme, on doit respecter les étapes suivantes :

- Ouvrez le logiciel Arduino.
- Allez dans le menu `Fichier`.
- Choisissez l'option `enregistrer sous...`
- Saisir le nom du programme (comme exemple `LED`)
- Cliquer sur `enregistrer`.

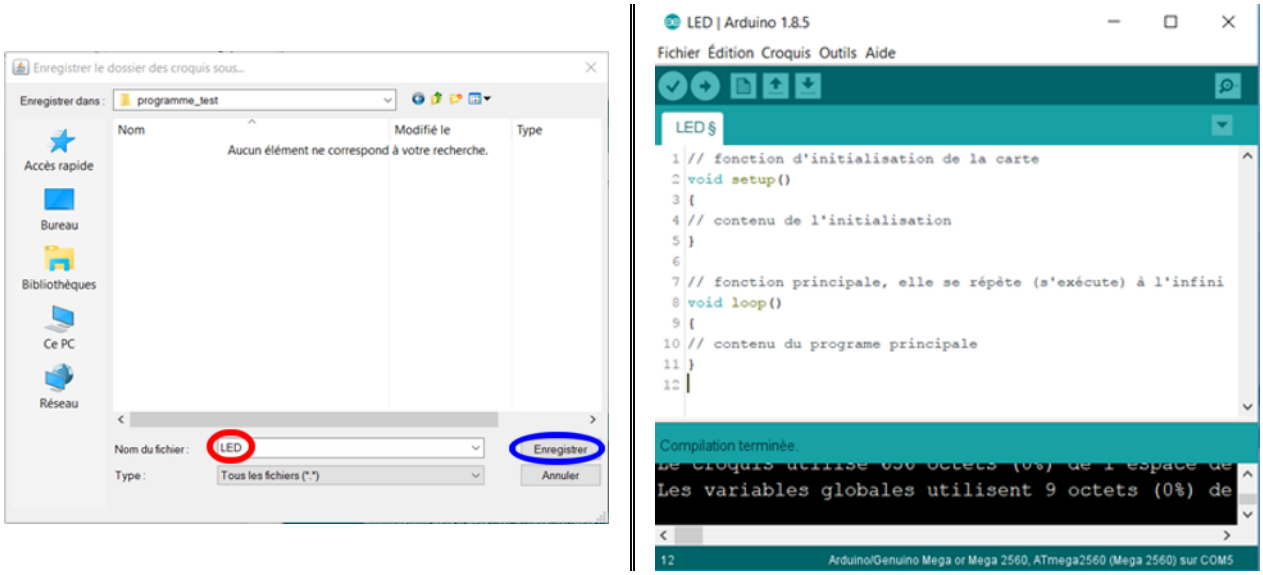


Fig.5 : Création d'un nouveau projet

4.1.2 Déclaration des variables

Cette étape définie les broches à utiliser au sein de la carte. Comme exemple, on veut allumer la `led_rouge` câblée sur la broche `43`. La déclaration peut se faire de deux manières :

- En utilisant l'instruction `int` :

```
// Déclaration de la broche 43 de la carte ARDUINO en tant que variable de type 'integer' sous le nom de led_rouge
int led_rouge 43;
```

- En utilisant l'instruction `#define` :

```
// Déclaration de la broche 43 de la carte ARDUINO en tant que variable de type 'E/S' sous le nom de led_rouge
#define led_rouge 43
```

La ligne précédée par `'//'` est une ligne de commentaire. Elle est facultative.

4.1.3 Déclaration des entrées/sorties

Déclarer si la broche est une **entrée** (lecture de la donnée) ou une **sortie** (envoi de la donnée). On déclare les **entrées/sorties** dans la fonction `setup()` en utilisant la fonction `pinMode(nom,mode);` :

- `nom` : Variable affectée à la broche.
- `mode` : Broche en entrée (`INPUT`) ou sortie (`OUTPUT`).

```
void setup() {
// Déclaration de la broche led_rouge en tant que sortie
  pinMode(led_rouge,OUTPUT);
}
```

4.1.4 Code principal

Cette étape consiste à créer le contenu du programme dans la fonction `loop()` pour allumer la diode LED. En électronique numérique, un niveau "haut" ou "`HIGH`" correspond à une tension de +5V et un niveau "bas" ou "`LOW`" correspond à une tension de 0V (généralement la masse). Pour cette opération, on utilise la fonction `digitalWrite(nom,etat)`:

- `nom` : Variable affectée à la broche.
- `etat` : état logique "`HIGH`" ou "`LOW`".

Le programme final devient alors comme suit :

Fig.6 : `Exemple4` : `Sortie_numerique_LED_Rouge.ino`

4.2 Exercices

Ex4.2.1 Ecrire un programme qui permet de clignoter la diode LED Rouge (sur la broche D43 de la carte Arduino) selon la procédure suivante : elle s'allume pendant 1 seconde et s'éteint pendant 2 secondes, en utilisant la fonction `delay(T)` or `T` est en millisecondes.

Ex4.2.2 Ecrire un programme qui permet de faire un jeu de lumière comme suit :

- LED verte allumée, LED jaune éteinte, LED rouge éteinte, pendant 700 millisecondes.
- LED verte éteinte, LED jaune allumée, LED rouge éteinte, pendant 800 millisecondes.
- LED verte éteinte, LED jaune éteinte, LED rouge allumée, pendant 1000 millisecondes.

5. Sortie analogique

5.1 Commande du signal en MLI (PWM)

Contrairement à la sortie numérique, une sortie analogique est une tension continue variant de 0 à 5 Volts. L'IDE Arduino traite les sorties analogiques en tant que sortie MLI (Modulation par Largeur d'Impulsion) ou PWM (Pulse Wave Modulation). Ce type de sortie dispose de 255 paliers de tension. Le palier élémentaire étant égal à Q = 5/255 = 0.0196 V.

Pour cela, on utilise l'instruction `analogWrite(broche,val)` qui utilise deux arguments :

- `broche` : est la broche sur laquelle le signal MLI sera généré.
- `val` : représente un nombre entier compris entre 0 et 255.

La tension générée à la sortie de la broche Vs est calculée comme suit :

Vs = `val` * Q = `val` * 0.0196

Exemple : si `val` = 83; on aura une tension de sortie Vs = 83 * 0.0196 = 1.627 V.

Si on veut une tension de sortie Vs = 2.6 V, `val` est calculée comme suit :

`val` = 51 * Vs = 51 * 2.6 = 132,6. Cette valeur est arrondie à 133.

Donc pour une tension de sortie égale à 2.6V qui correspond à allumer la diode LED blanche à 52% de sa valeur maximale, le programme est le suivant :

Fig 7 : Exemple5 : `Sortie_analogique_LED_Blanche.ino`

Attention : Toutes les sorties se comportent comme sortie numérique. Uniquement celles marquées comme sortie PWM sont des sorties analogiques. Se référer au manuel de la carte ARDUINO MEGA sur le site https://store.arduino.cc/arduino-mega-2560-rev3

5.2 Exercices

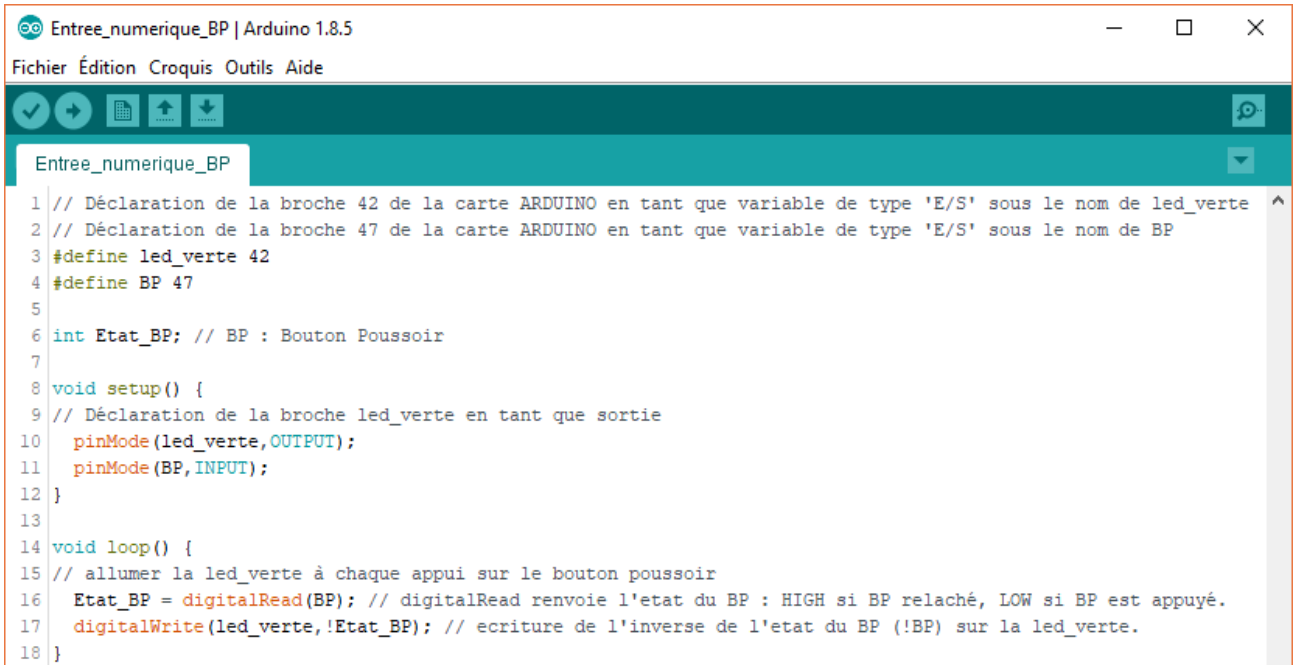
Ex5.2.1 Ecrire un programme Arduino qui permet de faire une augmentation progressive du niveau de luminosité de la diode blanche pendant 3 secondes et une diminution progressive pendant 4 secondes.

Ex5.2.2 Ecrire un programme qui permet allumer la diode RGB en jaune (rouge = 90 %, vert = 80 %, bleu = 0 %) pendant 2 secondes, cyan (rouge = 0 % ; vert = 90 %, bleu = 80 %) pendant 3 secondes, magenta (rouge = 85 %, vert = 0 %, bleu = 95 %) pendant 2 secondes.

6. Entrée numérique

6.1 Introduction

La lecture des données numériques se fait par la fonction `digitalRead(broche)`, elle renvoie l'état logique `HIGH` (1 logique) ou `LOW` (0 logique).
Le programme suivant illustre le fonctionnement de l'entrée numérique "`bouton poussoir BP`". A chaque appui, la `led_verte` s'allume.




```
1  // Déclaration de la broche 42 de la carte ARDUINO en tant que variable de type 'E/S' sous le nom de led_verte
2  // Déclaration de la broche 47 de la carte ARDUINO en tant que variable de type 'E/S' sous le nom de BP
3  #define led_verte 42
4  #define BP 47
5
6  int Etat_BP; // BP : Bouton Poussoir
7
8  void setup() {
9  // Déclaration de la broche led_verte en tant que sortie
10   pinMode(led_verte,OUTPUT);
11   pinMode(BP,INPUT);
12 }
13
14 void loop() {
15 // allumer la led_verte à chaque appui sur le bouton poussoir
16   Etat_BP = digitalRead(BP); // digitalRead renvoie l'etat du BP : HIGH si BP relaché, LOW si BP est appuyé.
17   digitalWrite(led_verte,!Etat_BP); // ecriture de l'inverse de l'etat du BP (!BP) sur la led_verte.
18 }
```

Fig. 8 : Exemple6 : `Entree_numerique_BP.ino`

6.2 Exercice

Ex6.2.1 Ecrire un programme Arduino qui réalise les fonctions suivantes :

- `BP` relaché : `led_jaune` allumée, `led_rouge` éteinte.
- `BP` appuyé : `led_jaune` éteinte, `led_rouge` allumée.

Ex6.2.2 Ecrire un programme Arduino qui, en cliquant sur le bouton, permet d'augmenter le niveau de la luminosité de la diode `led_blanche` de 10 % jusqu'à atteindre la valeur maximale et ainsi de refaire le même travail indéfiniment.

7. Entrée analogique

7.1 Introduction

La lecture des données analogiques se fait par la fonction `analogRead(broche)` et retourne une valeur entre 0 et 1023 : `val = analogRead(broche)` où

- `broche` est le numéro de l'entrée analogique à lire.
- `val` est le résultat de la conversion analogique→numérique.

La carte Arduino MEGA contient un convertisseur analogique-numérique (CAN) 10 bits à 16 canaux. Pour chaque canal, la tension comprise entre 0 et 5V est convertie en une valeur comprise entre 0 et 1023 ce qui définit la résolution du convertisseur Q = 5/1023 = 4.888 mV.
Il faut environ 100 microsecondes pour lire une entrée analogique, de sorte que le taux de lecture maximum est d'environ 10 000 fois par seconde.

L'exemple suivant permet de lire la valeur de la tension aux bornes du potentiomètre P2 (en bas, à côté du moteur) et de l'afficher sur le moniteur série. (Mettre le sélecteur (19) sur la position "`M`").

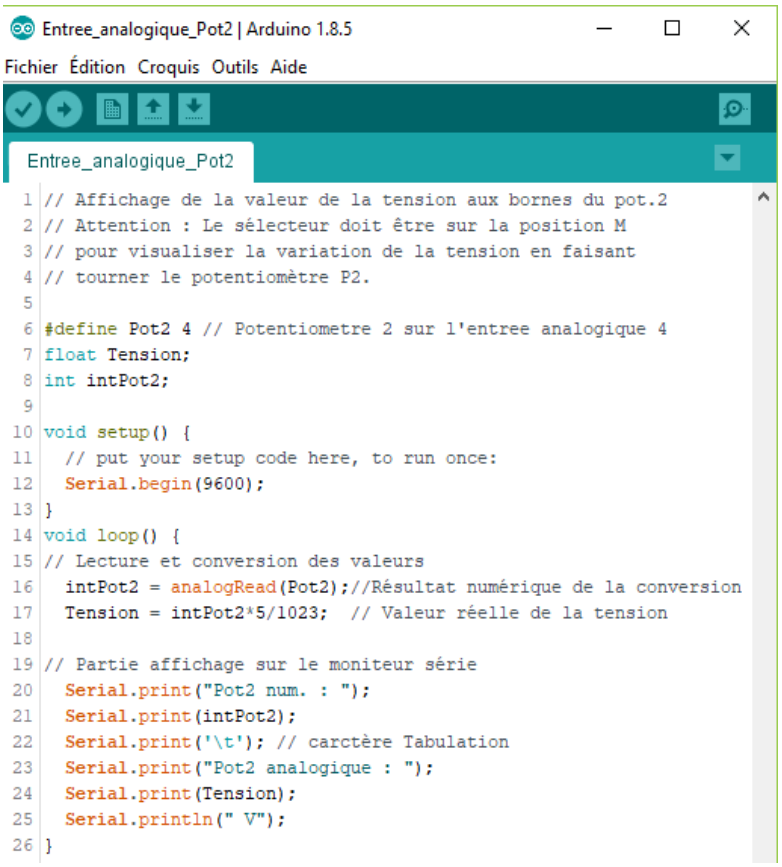


Fig 9 : Exemple7 : Entree_analogique_Pot2.ino

7.2 Exercices

Ex7.2.1 Ecrire un programme Arduino qui permet d'afficher la valeur mesurée Vm du capteur LM35 et ensuite d'afficher la température T en utilisant la formule suivante : **T = Vm * 0.488** où T : Température [Celsius] et Vm : valeur mesurée [0 - 1023].

Ex7.2.2 Ecrire un programme Arduino qui, en mettant la position du sélecteur sur "M" et en faisant tourner le potentiomètre P2 (en bas, à côté du moteur), permet d'afficher la valeur mesurée de la tension en ses bornes sur le moniteur série, puis de faire varier la luminosité de la led_blanche en fonction de cette valeur en utilisant l'instruction map de l'IDE ARDUINO.

8. Affichage sur l'écran LCD

LCD signifie "Liquid Crystal Display" et se traduit, en français, par "Écran à Cristaux Liquides". L'écran utilisé permet l'affichage de 16 caractères par ligne sur 2 lignes.

Pour l'utilisation de l'écran LCD on doit initialiser la bibliothèque <LiquidCrystal.h> comme suit :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(rs,en,d4,d5,d6,d7).
```

Avec :

lcd: une variable de type LiquidCrystal.

rs : broche de sélection de registre.

en : broche « enable » qui permet d'activer la lecture dans les registres de l'écran LCD.

d4, d5, d6, d7 : quatre lignes, utilisées pour le transfert de données entre la carte ARDUINO et l'écran LCD.

8.1 Calibrage de l'écran LCD

`Téléverser` le programme suivant dans la carte Arduino et tourner `le potentiomètre P1` (en haut, à côté de l'écran) pour régler le contraste de l'écran jusqu'à ce que la chaine de caractère « `Bonjour !` » soit affichée correctement.

Test_LCD | Arduino 1.8.5

Fichier Édition Croquis Outils Aide

Test_LCD

```
#include <LiquidCrystal.h>
const int rs=34;
const int en=33;
const int d4=32;
const int d5=31;
const int d6=30;
const int d7=29;
LiquidCrystal lcd(rs,en,d4,d5,d6,d7);

void setup() {
// Configuration du nombre de ligne et de colonne de l'écran LCD : 16 colonnes et 2 lignes
// Les colonnes sont numérotées de 0 à 15 et les lignes sont numérotées de 0 à 1

  lcd.begin(16,2);
// Placer le curseur sur la colonne 0, ligne 1
  lcd.setCursor(0,1);
// Ecrire le message suivant "Bonjour !"
  lcd.print("Bonjour !");
  }

void loop() {
}
```

Fig10 : Exemple8 : `Test_LCD.ino`

8.2 Exercices

Ex.8.2.1 Ecrire un programme Arduino qui permet le décalage du caractère "`a`" chaque 1 s sur la première ligne de l'écran LCD.

Ex.8.2.2 Ecrire un programme qui permet d'afficher l'état du bouton poussoir (appuyé ou relâché).

Ex.8.2.3 Reprendre l'exercice 7.2.1 mais en utilisant l'afficheur LCD : 1ère ligne valeur mesurée du capteur, 2ème ligne : valeur de la température.

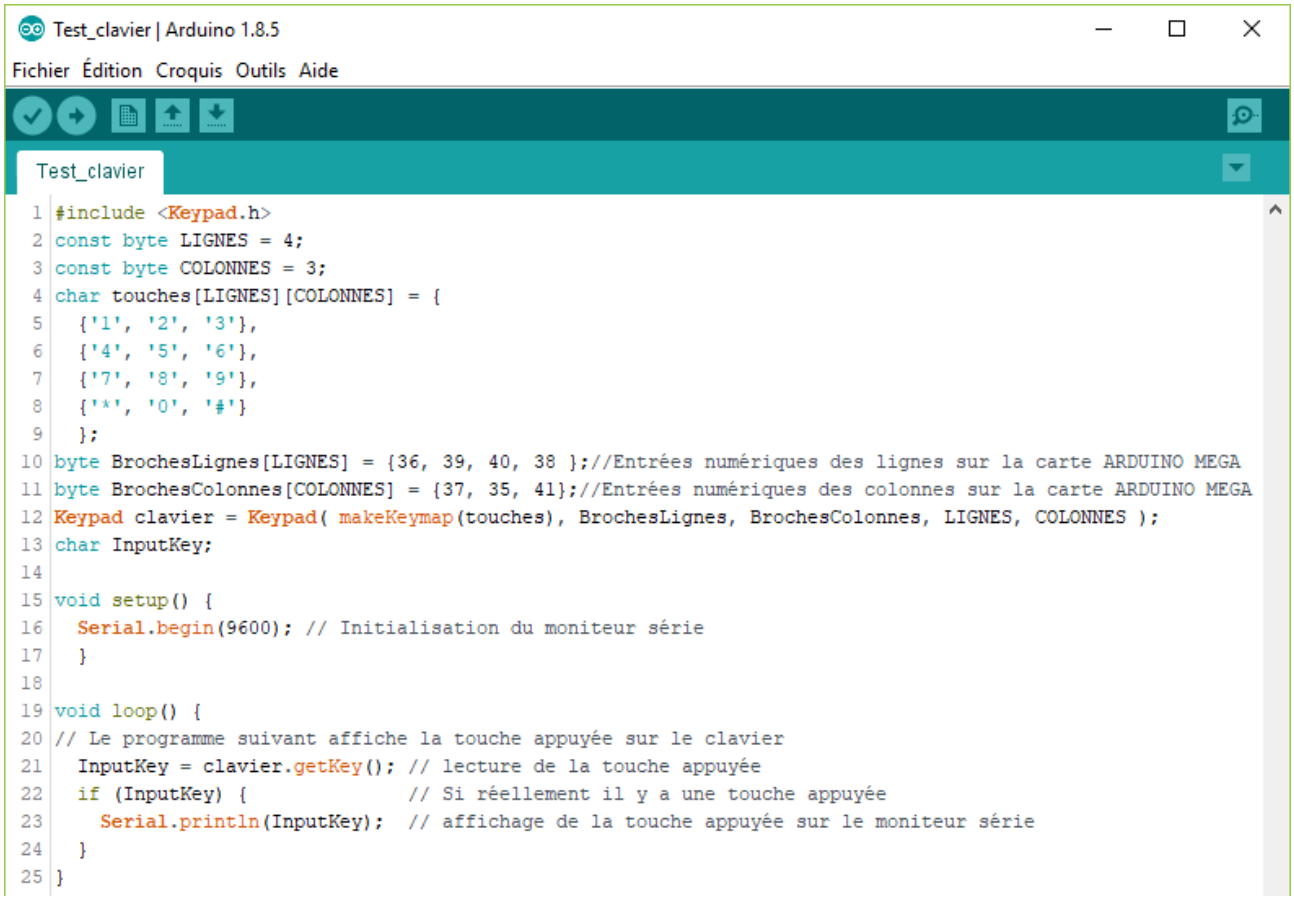
9 Lecture d'une touche du clavier

9.1 Présentation du clavier

Le clavier utilisé est un clavier rigide à base de 12 touches. Chaque touche est configurée dans un format de matriciel de 3 colonnes par 4 lignes.

Fig. 11 : Clavier matriciel 12 Touches

9.2 Exemple de code pour utiliser le clavier




```ino
#include <Keypad.h>
const byte LIGNES = 4;
const byte COLONNES = 3;
char touches[LIGNES][COLONNES] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
  };
byte BrochesLignes[LIGNES] = {36, 39, 40, 38 };//Entrées numériques des lignes sur la carte ARDUINO MEGA
byte BrochesColonnes[COLONNES] = {37, 35, 41};//Entrées numériques des colonnes sur la carte ARDUINO MEGA
Keypad clavier = Keypad( makeKeymap(touches), BrochesLignes, BrochesColonnes, LIGNES, COLONNES );
char InputKey;

void setup() {
  Serial.begin(9600); // Initialisation du moniteur série
  }

void loop() {
// Le programme suivant affiche la touche appuyée sur le clavier
  InputKey = clavier.getKey(); // lecture de la touche appuyée
  if (InputKey) {              // Si réellement il y a une touche appuyée
    Serial.println(InputKey);  // affichage de la touche appuyée sur le moniteur série
  }
}
```

Fig. 12 : Exemple9 : Test_clavier.ino

9.3 Exercices

Ex9.3.1 Ecrire un programme ARDUINO qui en cliquant sur la touche :

- ‘*’ affiche sur l'écran LCD les chiffres pairs.
- ‘#’ affiche sur l'écran LCD les chiffres impairs.
- un délai d’une demi-seconde entre chaque affichage des chiffres.
- Un écran d'accueil au début précisera le choix entre la touche '*' ou la touche '#'.

Ex9.3.2 Ecrire un programme ARDUINO qui en cliquant sur une touche du clavier, affichera si le chiffre est pair ou impair. L'appui sur '*' ou '#' affichera le message "indefini".

Ex9.3.3 Ecrire un programme Arduino qui permet de réaliser la fonction de multiplication de deux chiffres selon le scénario suivant :

- Appui sur la première touche et son affichage (seuls les chiffres de 0 à 9 sont acceptés).
- L'appui sur "*" efface la touche de l'écran LCD.
- L'appui sur "#" mémorise la touche comme étant le premier opérande.
- Appui sur la deuxième touche et son affichage. (seuls les chiffres de 0 à 9 sont acceptés).
- L'appui sur "*" efface la touche de l'écran LCD.
- L'appui sur "#" mémorise la touche comme étant le premier opérande.
- Affichage de l'équation complète avec le résultat.