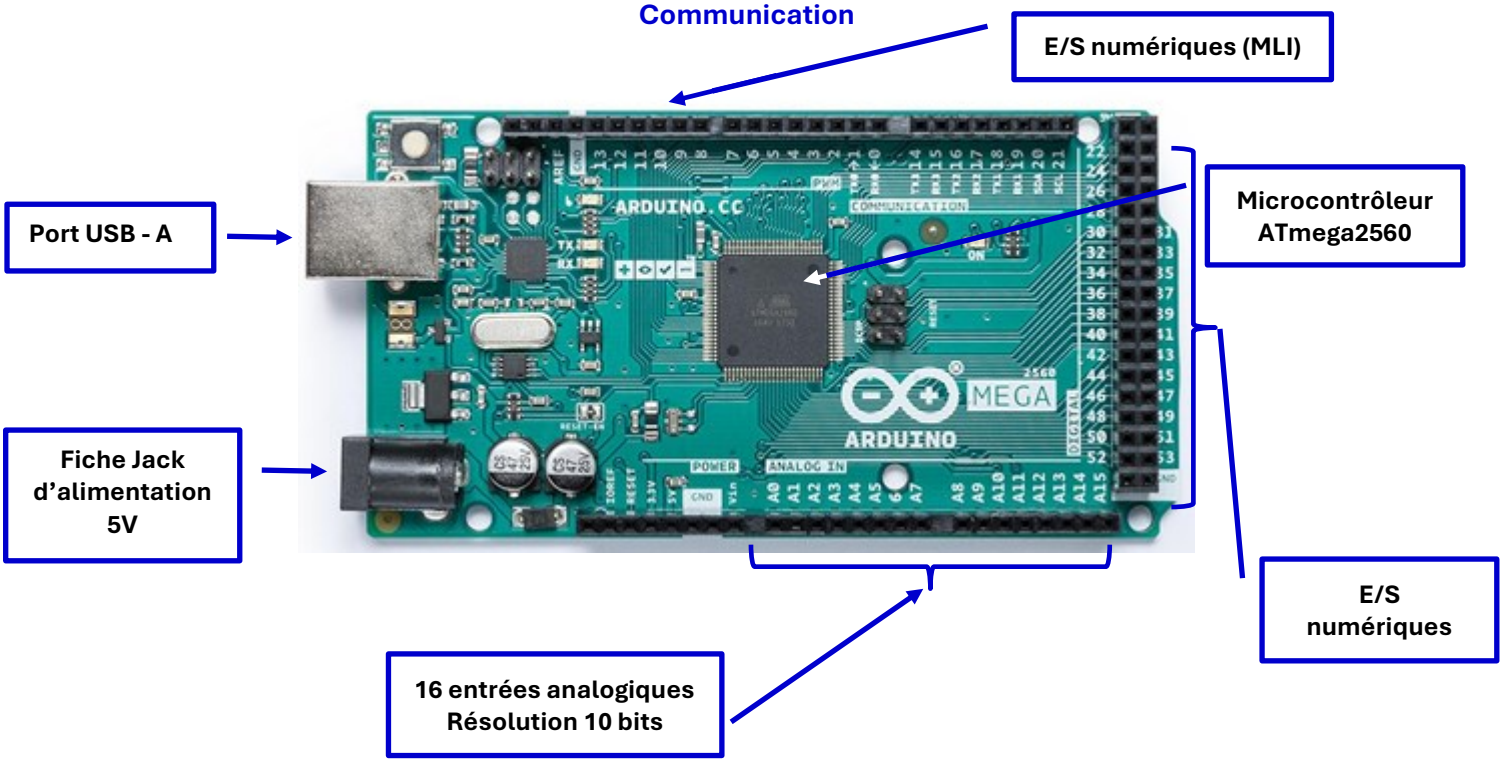
ADESFA
Séminaire de formation des formateurs
Informatique Industrielle et Systèmes Embarqués
INUBIL-Douala – Université Paris-Saclay
28 octobre- 02 novembre 2024

SYSTEME ARDUINO
MEGA 2560

TP2 - E/S, interruptions, Commande d'un moteur, Communication



Objectif : *- Utiliser le système Arduino MEGA2560 R3 et sa programmation*
- Mettre en œuvre des exemples utilisant les ports d'E/S et les interruptions

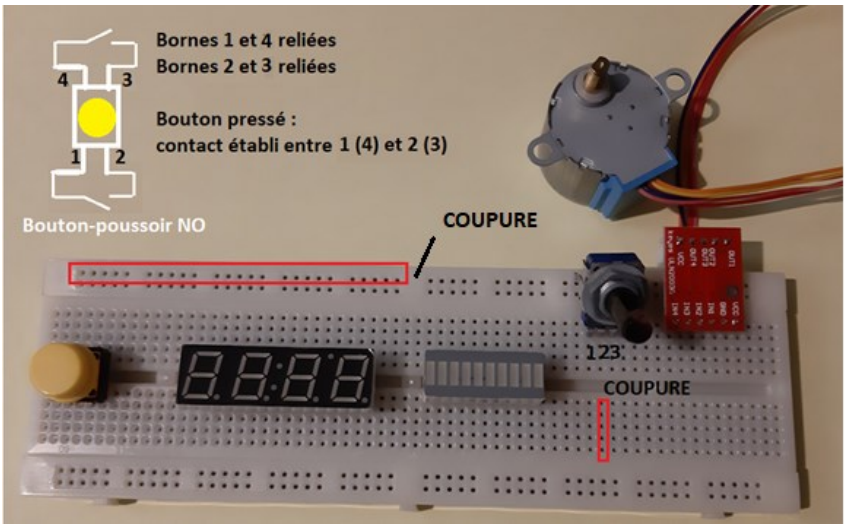
Table des matières

I. Rappel

Composants requis :
Le tableau 1 liste les différents éléments et composants requis pour la mise en œuvre des deux sujets de travaux pratiques.

Tableau 1 : kit de travaux pratiques par poste de TP

Description	***Caractéristiques***	***Quantité***
Carte Arduino	MEGA 2560	1
Câble USB2.0	AM-Micro BM 0,8 mm	1
Platine d'expérimentation	840 points	1
Afficheur 7-segment à 4 chiffres, anode commune	YSD-439AB4B-35	1
Moteur pas-à-pas	28BYJ-48 5VDC	1
Pilote (driver) pour moteur pas-à-pas	ULN2003	1
Afficheur LED Vert/Rouge, bargraphe	2473107	1
Potentiomètre rotatif	2,2 kΩ	1
Bouton-poussoir	NO	4
Diodes électroluminescentes - LED	Blanc	4
	Jaune	4
	Rouge	4
	Vert	4
Résistances	220 Ω ou 270 Ω	1
	330 Ω ou 390 Ω	1
	1 kΩ	2
	4,7 kΩ	4
Câble en nappe prêt à l'emploi	10 fils	1
Fils de connexion prêts à l'emploi différents coloris	12/16/20/25 cm	25



Rappel : La figure 1 illustre le montage du diviseur de tension avec un potentiomètre rotatif.

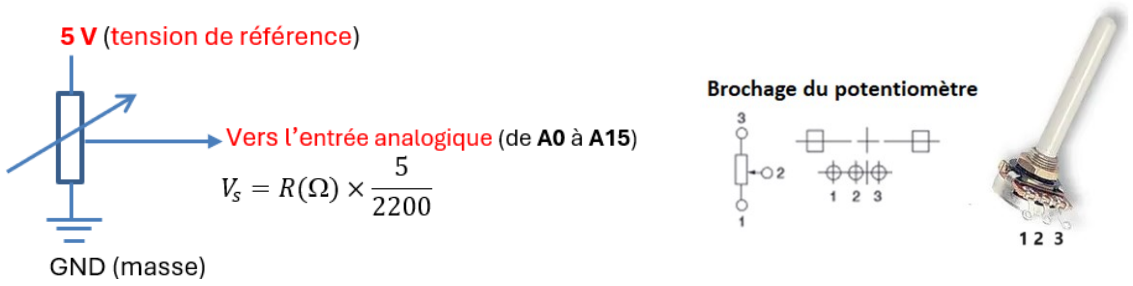
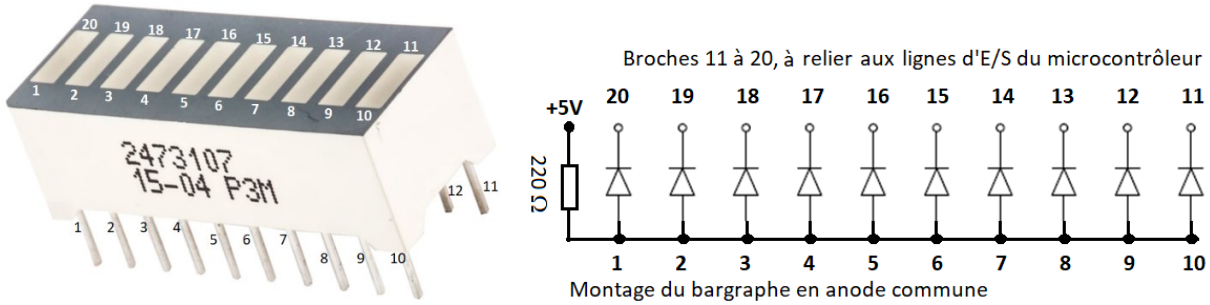


Fig. 1 Montage du potentiomètre rotatif 2,2 KΩ

La Fig. 2 illustre le brochage de l'afficheur bargraphe à 10 segments.



(a) *(b)*

Fig. 2 Image (a) et brochage (b) de l'afficheur bargraphe à 10 segments.

Les Figures 3 et 4 illustrent le montage et le brochage de l'afficheur 4 chiffres à 7 segments.

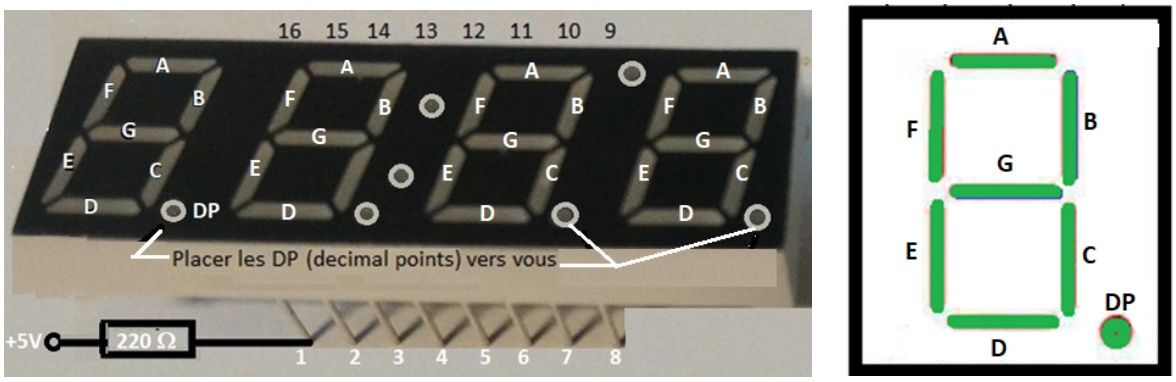


Fig. 3 Afficheur à 7 segments : montage de l'afficheur en anode commune.

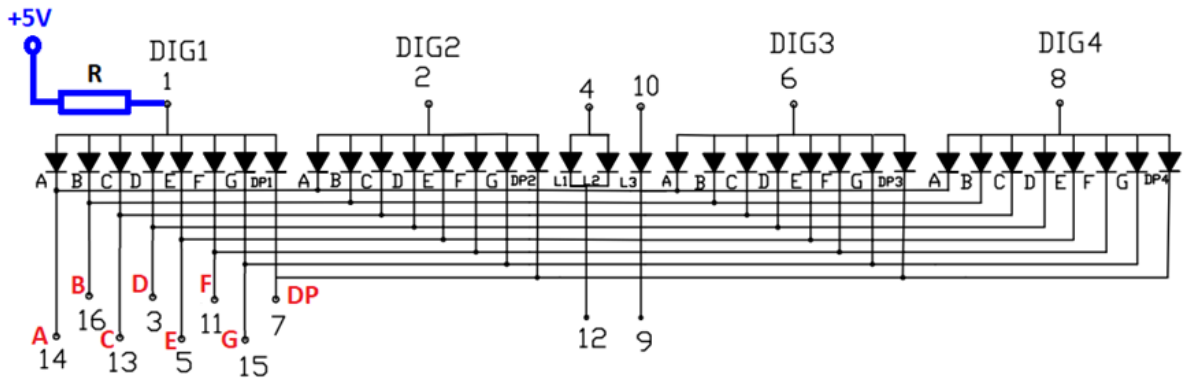


Fig. 4 Afficheur 7-segments : brochage de l'afficheur en anode commune.

Les broches **1, 2, 6** et **8** (anodes communes) permettent d'alimenter les afficheurs **DIGIT1**, **DIGIT2**, **DIGIT3** et **DIGIT4** respectivement.

II. Voltmètre numérique à 1 chiffre

1- Ecrire un programme qui permet de mesurer une tension variable fournie par un potentiomètre câblé en diviseur de tension entre 3,3 V et 0 V (la masse ou GND). Relier la borne du milieu du potentiomètre à une entrée analogique du M0 PRO de votre choix.

a- Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

b- Quelles sont les limites éventuelles de ce voltmètre ?

c- Que proposez-vous pour éventuellement remédier à ces limites ?

2- Calibrer la valeur de sortie pour que « 9 » corresponde à la tension max et « 0 » à la tension min.

a- Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

b- Afficher sur le moniteur la valeur réelle mesurée. En déduire :

i. La valeur de tension analogique qui correspond au minimum (0) ?

ii. La valeur de tension analogique qui correspond au maximum (9) ?

3- Changer votre programme pour afficher la valeur calibrée (valeur entière entre 0 et 9) de la tension mesurée sur un afficheur 7-segment.

a- Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

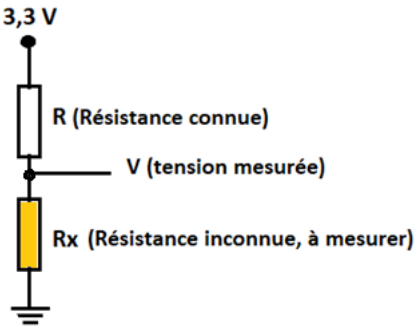
b- Les valeurs affichées sur l'afficheur 7-segments, correspondent-elles aux valeurs obtenues en II-2-b ?

c- Quelles sont les raisons des différences éventuelles.

III. Ohmmètre

Câbler un diviseur de tension avec une résistance de valeur connue et une résistance de valeur inconnue aux entrées appropriées du système MEGA 2560 et écrire un programme qui permet d'afficher sur le moniteur la valeur de la résistance inconnue.
Se basant sur le principe du diviseur de tension, nous pouvons obtenir
(Attention - utiliser 2 résistances au lieu du potentiomètre) :

$$V_{mesurée} = \frac{R_X}{R + R_X} \times 3{,}3 \rightarrow R_X = \frac{V_{mesurée} \times R}{3{,}3 - V_{mesurée}}$$



a- Quelle est la valeur de la résistance inconnue Rx (obtenue à l'aide du code des couleurs ou d'un ohmmètre).

b- Donner la valeur de Rx affichée sur le moniteur.

c- Vérifier si cette valeur mesurée (b) correspond à la valeur de la résistance inconnue (a).

d- Changer éventuellement votre programme pour faire correspondre la valeur réelle de la résistance inconnue à la valeur mesurée. Tester votre programme avec différentes valeurs de résistances.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

IV. Multimètre

Combiner les parties II et III pour mettre en place la possibilité d'utiliser un voltmètre OU un ohmmètre en fonction de l'état d'entrée d'un sélecteur de gamme, fonction assurée par un bouton-poussoir :

High → Voltmètre
Low → Ohmmètre

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

Partie voltmètre _____

Partie ohmmètre _____

V. Bargraphe : indicateur visuel de niveau

L'exemple suivant permet de déclarer un groupe de 10 pins d'E/S numériques en SORTIE à la fois. Dans cet exemple, les broches numériques 2 à 11 sont déclarées en sortie.

```
const int ledCount = 10;    // Nombre de LED du bargraphe
int ledPins[] = {
                2, 3, 4, 5, 6, 7, 8, 9, 10, 11
};
void setup() {
  for (int i = 0; i < ledCount; i++) {
  pinMode(ledPins[i], OUTPUT);
 }
}
```

a- Refaire la question II-2 et II-3 en utilisant l'afficheur bargraphe au lieu de l'afficheur 7-segment.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

Partie voltmètre _____

Partie ohmmètre _____

a- Tester le montage du bargraphe en anode commune et en cathode commune. Avec toutes les LEDs allumées, comparer le fonctionnement du système entre les deux montages (luminosité, alimentation). En pratique, lequel de ces deux montages est le plus approprié ? Justifier votre réponse.

_____

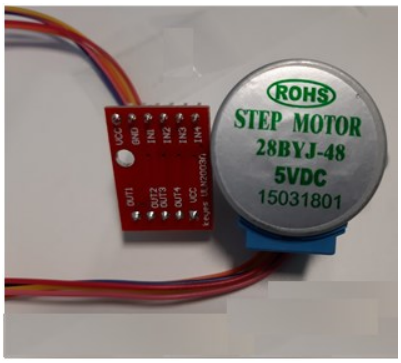
_____

_____

_

VI. Commande d'un moteur pas-à-pas

Dans cette partie, nous allons utiliser le moteur pas-à-pas à 4 phases (4 fils + masse) de référence 28BYJ-48 5VDC, ayant les caractéristiques suivantes :



– Tension d'alimentation 5V continu (DC)
– Engrenages internes de réduction 1/64
– 64 pas par tour, angle de pas =560 °/64 = 5,625°
– Couple de rotation : > 34,3 mN.m (~340 g.cm)

Ce montage fonctionne sans alimentation externe, cependant, il est préférable d'utiliser une alimentation externe de 5V afin de soulager votre carte Arduino.

Fig. 5 Le moteur pas-à-pas 28BYJ-48

Les 4 phases du moteur pas-à-pas sont reliées via un connecteur dont les 5 broches du coté moteur, sont présentées dans la Fig. 6.

Numéro Pin	***Phase***	***Couleur du fils***
1	4	Bleu
2	2	Rose
3	3	Jaune
4	1	Orange
5	Vcc	Rouge

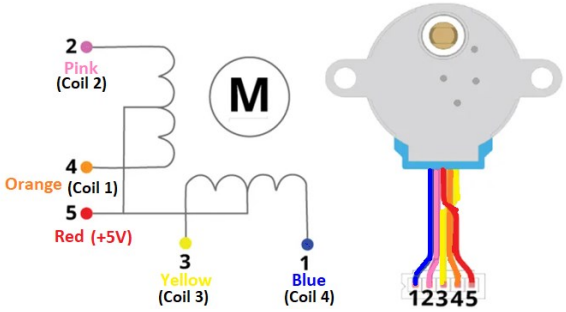


Fig. 6 Brochage du connecteur moteur-pas-à-pas et driver ULN2003

Le microcontrôleur ne peut pas piloter directement ce moteur pas-à-pas. Il est nécessaire de le relier à travers une interface de puissance, le driver ULN2003 dont le brochage est illustré dans la Fig. 7.

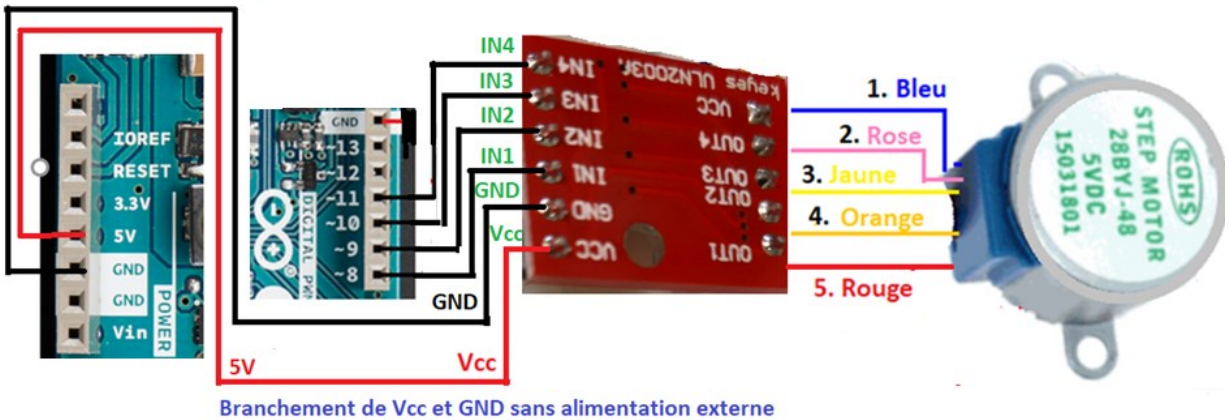


Fig. 7 Circuit de commande du moteur pas-à-pas

a- Réaliser le montage de la Fig. 7 en respectant le brochage et les polarités indiqués. Les phases du moteur sont reliées en anode commune.

b- Ecrire un programmer pour faire tourner le moteur pas-à-pas dans le sens des aiguilles d'une montre. Remarquer l'utilisation du « code Grey ». Prévoir **un délai de 5 ms** entre chaque pas.

Entrée	***Pas 1***	***Pas 2***	***Pas 3***	***Pas 4***	***Pas 5***	***Pas 6***	***Pas 7***	***Pas 8***
IN4	0	0	1	1	1	1	1	0
IN3	1	0	0	0	1	1	1	1
IN2	1	1	1	0	0	0	1	1
IN1	1	1	1	1	1	0	0	0

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

c- Changer le programme précédent pour faire tourner le moteur dans le sens antihoraire.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

d- Ajouter un bouton poussoir, changer le programme précédent pour faire tourner le moteur dans le sens horaire lorsque le bouton-poussoir est en position de repos et dans le sens antihoraire lorsque le bouton est pressé. Prévoir **un délai d'arrêt de 20 ms** avant chaque changement de sens de rotation.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

VII. Interruption

Quelques recommandations pour utiliser les interruptions :

- La routine de service d'interruption (ISR : Interrupt Service Routine) doit être aussi courte que possible
- La fonction delay( ) utilise elle-même une interruption, donc elle ne fonctionnera pas à l'intérieur d'une ISR.
- Une variable globale, partagée entre le programme interrompu et l'ISR doit être déclarée comme VOLATILE.

Syntaxe d'une ISR (sur une M0 PRO 32-bits) :

attachInterrupt(pin, ISR, mode) ;

pin : numéro de la broche source de l'interruption
ISR : nom de la routine de service d'interruption
Mode : type du signal d'interruption,
Exemple : CHANGE → changement de niveau

Saisir l'exemple suivant d'un programme, utilisant l'entrée numérique 7 comme source d'interruption. A l'exception de l'E/S numérique n° 4, vous pouvez également utiliser toute autre ligne d'E/S comme source d'interruption de votre choix.

```
const int interruptPin7 = 7;   // numéro de la source d'interruption
const int ledPin5 =  5;   // numéro de la broche à relier à la LED
// variable globale à déclarer comme VOLATILE
volatile int buttonState = 0;     // variable pour lire l'état du bouton-poussoir

void setup() {
 Serial.begin(9600);
 // initialiser la broche reliée à la LED en sortie
 pinMode(ledPin5, OUTPUT);

 // initialiser la broche reliée au bouton-poussoir en entrée
 pinMode(interruptPin7, INPUT);

// Déclarer l'ISR et la relier à son vecteur d'interruption
 attachInterrupt(7, pin_ISR1, CHANGE);
}
void loop() {
// programme qui ne fait rien !
 while(1)
 {
 Serial.println("Fonctionnement normal 1");
 delay(500);
 digitalWrite(ledPin5, LOW);
 Serial.println("Fonctionnement normal 2");
 delay(500);
 digitalWrite(ledPin5, HIGH);
 }
}
void pin_ISR1() {
 //buttonState = digitalRead(interruptPin7);
 //digitalWrite(ledPin5, buttonSate);
 digitalWrite(ledPin5, HIGH);
 Serial.println("Source d'interruption 1");
}
```

a- Votre programme fonctionne-til normalement ? Expliquer les raisons de tout dysfonctionnement éventuel.

b- Proposer et mettre en œuvre une solution pour éventuellement remédier au problème de rebonds.

Appeler votre enseignant de TP pour valider le fonctionnement de votre proposition.

c- Lister les autres types d'interruptions que vous pouvez utiliser avec M0 PRO

d- Changer le programme précédent pour rajouter une seconde source d'interruption. Le programme doit afficher sur le moniteur le numéro de la source d'interruption utilisée et allume une LED différente pour chaque source.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

e- Combiner VI-e et II-3 pour afficher le numéro de la source d'interruption sur un afficheur 7-segment.

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

VIII. Communications avec un terminal d'ordinateur

Rappel de quelques instructions du langage C

E/S numériques
pinMode() : permet de configurer une ligne d'E/S en entrée ou en sortie.
digitalRead() : lire une valeur d'une broche numérique d'entrée spécifique.
digitalWrite() : écrire un « HIGH » ou un 'LOW' vers broche de sortie numérique spécifique.

E/S analogiques
analogRead() : lire une valeur d'une broche analogique d'entrée spécifique.
analogWrite() : écrire une valeur analogique (signal MLI) vers une brioche analogique de sortie.

Communication
Serial.begin(speed) : permet de définir la vitesse en baud de communication avec un terminal série.
Serial.read() : permet de lire les données série en entrée.
Serial.print(value, format) ou Serial.println(value,format) : permet d'écrire une valeur avec un format spécifique (optionnel) vers un périphérique série. « ln » permet d'injecter un retour à la ligne en fin d'impression.
Serial.peek() : permet d'obtenir une copie du prochain octet (caractère) d'une trame de données série.
Serial.available() : donne le nombre d'octets (caractères) disponibles pour lecture à partir du port.

1. Liaison Terminal – Arduino.

Durant le premier TP, nous avons pu afficher sur l'écran de l'ordinateur, les résultats de mesure de tensions analogiques via la liaison USB en utilisant les fonctions **Serial.begin(9600)** et **Serial.print()** ou **Serial.println()**.

Sur l'écran, nous avons une fenêtre pour saisir les commandes et un bouton pour envoyer la commande saisie vers la carte Arduino, voir Fig. 8.

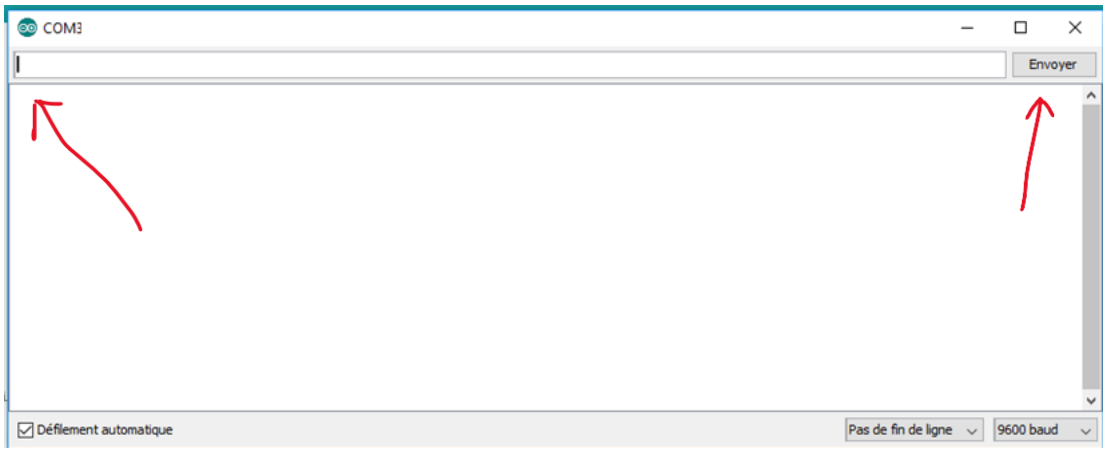


Fig. 8 Fenêtre de commande du moniteur

Saisir le programme suivant :

```
int led = 7;
void setup(){
 pinMode(led, OUTPUT);
 Serial.begin(9600);
}
void loop(){
 if (Serial.available()){
    char commande = Serial.read();
    // Agir selon la valeur reçue
 if (commande == 'e'){
 // Eteindre la LED
 digitalWrite(led, LOW);
}
 else if (commande == 'a'){
 // Allumer la LED
    digitalWrite(led, HIGH);
}
 else if (commande == 'c'){
 // Clignoter avec un délai correspondant à la valeur qui suit la lettre 'c'
       if (Serial.peek() > '0' && Serial.peek() <= '9'){
        //for (int i=1; i<=(commande-48); i++) {
        digitalWrite(led, HIGH);
        //delay (10);
        delay((Serial.read() - 48) * 100); // convertir le code ASCII en décimal
        digitalWrite(led, LOW);
       //}
}
}
}
}
```

a- Décrire la fonction de ce programme.

_____

b- Quelle est la différence entre **Serial.read()** et **Serial.peek()**

_____

_____

c- Changer ce programme pour afficher sur le moniteur série la commande reçue telle illustrée dans la figure 9.

Fig. 9 Copie écran après exécution du programme modifié

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

d- Modifier encore ce programme pour allumer 4 LED selon le code binaire correspondant à la valeur en décimal, qui suit la commande 'c'.

Par exemple pour des LED reliées en anode commune :

	LED3	*LED2*	*LED1*	*LED0*
Code reçu : c**3**	1	1	**0**	**0**
Code reçu : c**5**	1	**0**	**1**	**0**
Code reçu : c**8**	**0**	**1**	**1**	**1**

Les quatre LED doivent s'allumer si le code reçu est 'a' (allumer) et s'éteindre si le code reçu est 'e' (éteindre).

Appeler votre enseignant de TP pour valider le fonctionnement de votre programme.

IX. Conclusions

Quelles sont vont conclusions par rapport à :

- La programmation des E/S.
- La communication avec les E/S.
- La gestion des interruptions sur un système Arduino.
- Le développement d'une application basée sur un microcontrôleur.