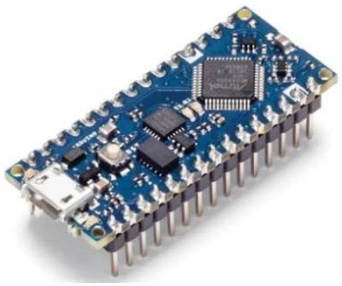
2ème année
Filière ASI
PASSERA Léon G3
POTEAU Mathis G4



Rapport final : Conception Et Réalisation de Systèmes Électronique (CERSE)

Sujet : Voltmètre numérique à calibre automatique (basé sur une carte microcontrôleur Arduino-Nano)

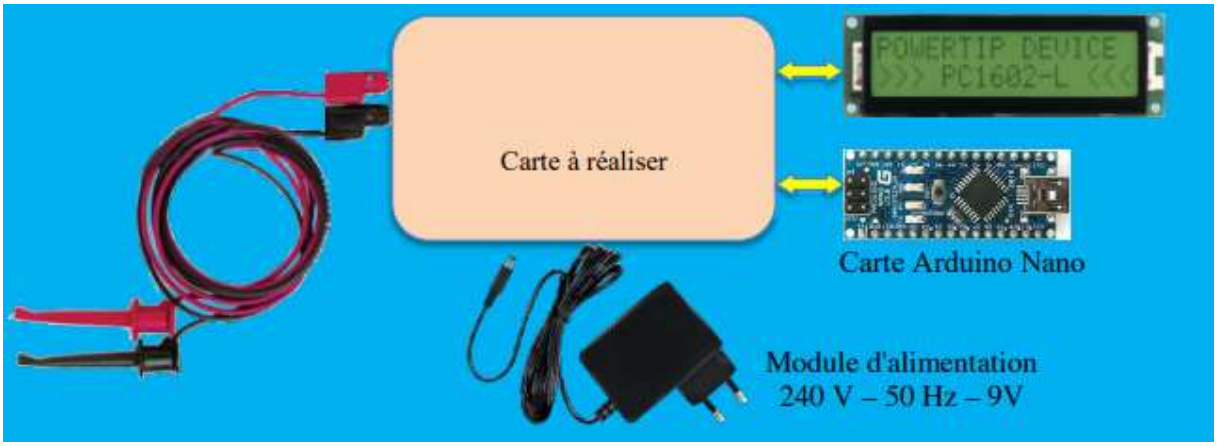


Table des matières

Présentation sujet

Objectif

L'objectif de ce bureau d'études est de réaliser un voltmètre numérique à calibre automatique basé sur une carte microcontrôleur Arduino-Nano. Nous devrons donc mesurer et afficher sur un écran LCD des valeurs efficaces et moyenne de tension d'un signal quelconque, avec adaptation automatique du calibre de mesure.

Cahier des Charges

Cet appareil dans le cas idéal devra respecter les exigences suivantes :

- ➔ Tension d'entrée à mesurer :
  - Alternative ou unidirectionnelle
  - De valeur absolue inférieure à 35 V

- ➔ Affichage sur écran LCD de la valeur moyenne du signal mis en entrée.

- ➔ Adaptation automatique à la valeur de la tension mise en entrée (changement de calibre automatique, par exemple en dessous de 3.5V)

- ➔ Impédance d'entrée : supérieure à 200 kΩ (constante et identique sur tous les calibres)

- ➔ Alimentation par un adaptateur secteur 9 V

- ➔ Le dispositif doit pouvoir supporter sans dommage une tension supérieure au calibre choisi, par exemple 35 V sur le calibre 3.5 V

- ➔ En bonus : ajout de l'affichage de la valeur "efficace vrai" du signal mis en entrée

Déroulement des séances

Premières approches Arduino

Partie 1 : AnalogRead()

Dans cette première partie, nous allons simplement essayer d'afficher sur l'écran LCD une valeur de tension obtenue avec la fonction AnalogRead(). Pour cela, nous utilisons la tension de sortie de l'Arduino 5V couplée à un potentiomètre afin de pouvoir modifier la tension.

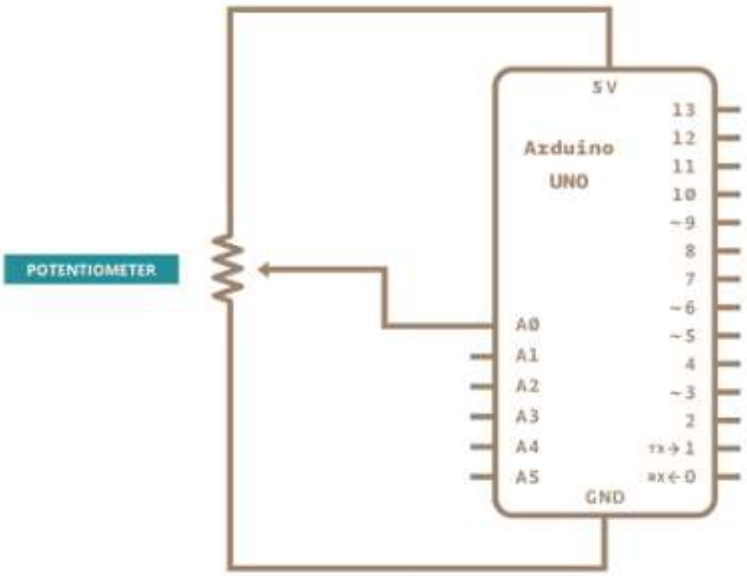


Fig1 : Schéma 1ère approche fonction AnalogRead()

*Remarque : La fonction **analogRead()** lit les valeurs analogiques renvoyées par l'arduino sur 10 bits et renvoie une valeur entre 0 et 1023 il est donc nécessaire de retransformer la valeur lue en valeur de Tension avant de l'afficher sur l'écran LCD.*

La tension aux bornes du potentiomètre est bien affichée sur l'écran LCD. Nous pouvons donc passer à la seconde approche.

PS : Le schéma de câblage de l'écran LCD à l'Arduino est disponible en Annnexe I.

Partie 2 : AnalogRead() avec source extérieure

Désormais, nous allons réutiliser le programme précédent pour cette fois-ci mesurer une tension extérieure (source : GBF). On envoie un signal sinusoïdal de 1 à 2V (voir fig2). On enlève le potentiomètre de la partie 1 et connectons la source de tension à la borne A0 et la masse de l'Arduino.

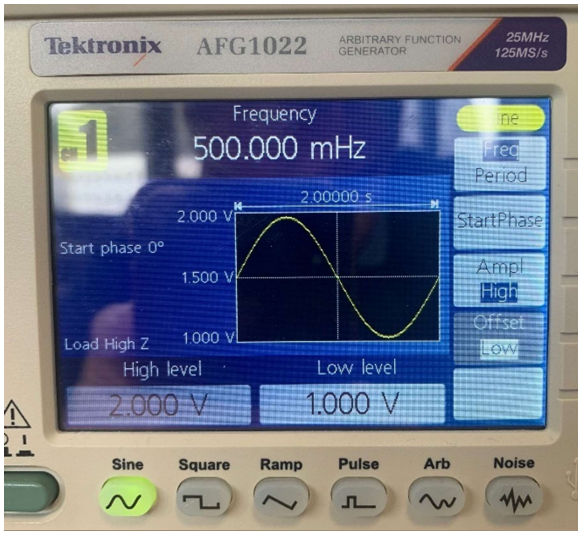


Fig2 : Signal envoyé sur le GBF

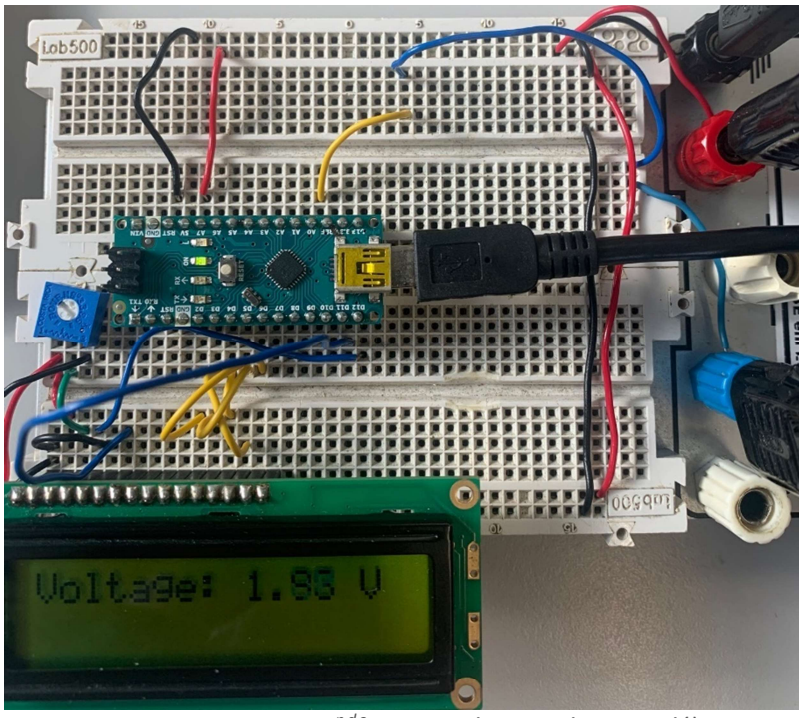


Fig3 : Montage 2nde approche AnalogRead()

La valeur de tension est correctement affichée sur l'écran LCD. Afin de respecter le CDC qui exige une mesure de tension pour un signal quelconque, on fait varier la forme des signaux (sinus, créneaux, triangle, …) du GBF. La mesure de tension s'effectue bien pour toute forme de signaux. Le test est concluant, nous allons donc conserver cette méthode de mesure pour le voltmètre.

Conditionnement Tension

Nous avons pour le moment effectué des mesures de tension sur la plage 0 à 5V. En effet, l'Arduino ne permet pas de réaliser des mesures en dehors de cette plage de tension.

Cependant, notre voltmètre est censé mesure des valeurs de tension allant de -35 à 35V. Nous allons donc devoir conditionner cette tension pour respecter la plage de tension admise en entrée de l'Arduino.

Pont diviseur de tension

L'objectif est dans un premier temps de désamplifier le signal afin de pouvoir passer d'une plage de tension en entrée de -35 <-> 35 V à -2.5 <-> 2.5 V en sortie.

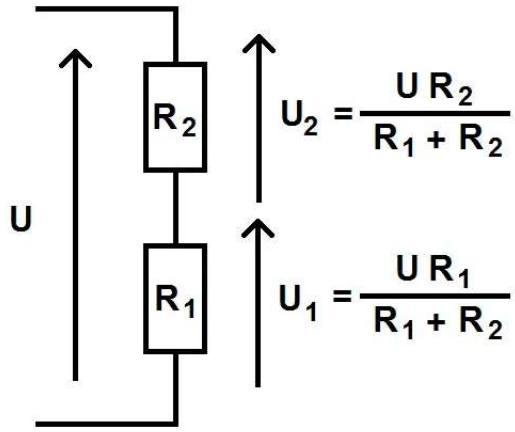


Fig4 : Schéma pont diviseur de tension

On souhaite récupérer U2=U/14 donc on choisit nos R tels que (R1+R2)/R2 =14.

Ce qui équivaut à R1=13*R2.

Nous disposons désormais d'une tension dans la plage -2.5<->2.5V.

Principe de masse virtuelle

Nous souhaitons désormais appliquer le concept de masse virtuelle afin de pouvoir passer d'une échelle de tension en entrée de -2.5 <-> 2.5 V à 0 <-> 5 V en sortie.

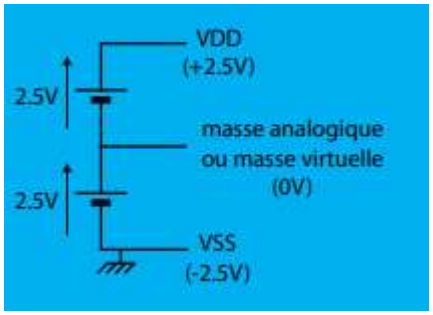


Fig5 : Schéma principe masse virtuelle

Ce schéma permet d'instaurer une nouvelle masse virtuelle 0 V qui vaut réellement -2.5 V. On aura donc une tension de 0V en entrée de l'Arduino pour une tension initiale de -2.5 V et une tension de 5V en entrée de l'Arduino pour une tension initiale de 2.5 V. Les tensions aux bornes des résistance étant égales, les résistances du modèle doivent être égales ni trop élevées ni trop faibles afin de ne pas perturber le courant.

Le schéma est donc composé d'un pont diviseur de tension, d'une « variation de masse » et d'un ampli-op suiveur. Nous avons rajouté un ampli-op suiveur afin de limiter le courant issu de la source de tension. Nous avons opté pour l'ampli Op LF356.

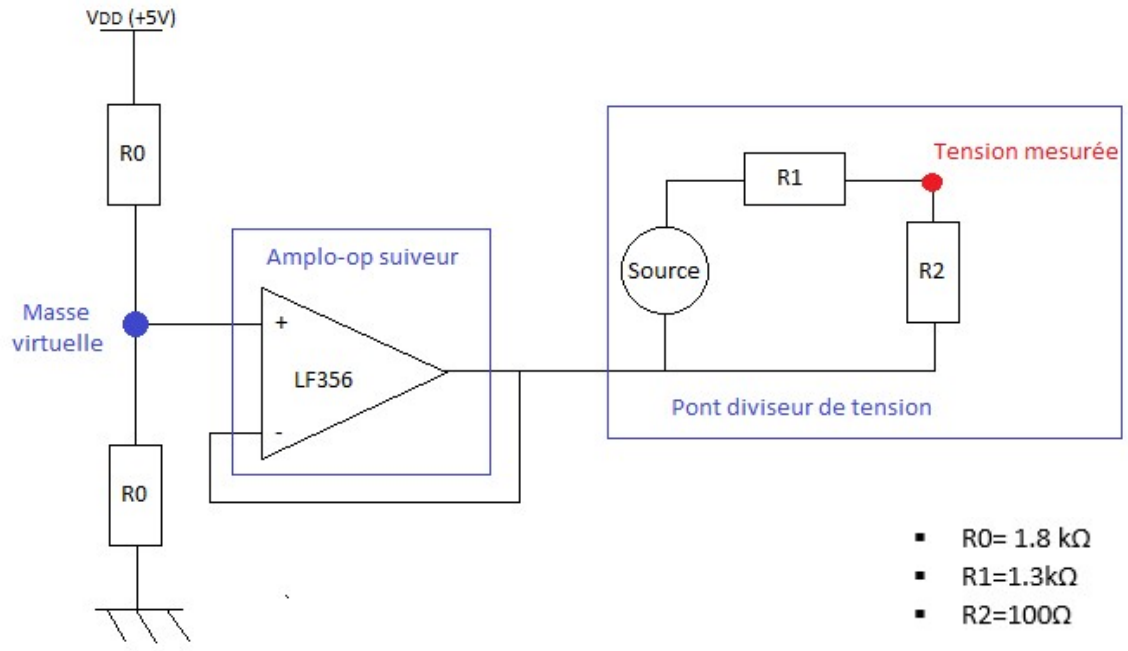


Fig6 : Schéma conditionnement Tension (calibre 35V)

PASSERA Léon
POTEAU Mathis

Ce schéma (fig. 6) devrait donc nous permettre de transformer une plage de tension -35<->35V issue de la source en une plage de tension 0-5V qu'on pourra mesurer.

Insertion des calibres

Afin d'améliorer l'incertitude de mesure de notre voltmètre pour des faibles tensions, nous insérons 2 calibres supplémentaires. Le premier mesurera des tensions comprises dans la plage -3.5 <-> 3.5V. Le conditionnement de ce calibre reposera sur le même principe que le premier avec un pont diviseur de tension.

Il n'est bien évidemment pas possible d'envoyer des tensions supérieures aux calibres sur les différents conditionnements. Pour cela, nous intégrons au début de chaque calibre des interrupteurs commandables par Arduino (OPTO-MOS). Ces composants sont pilotés par les sorties digitales de l'Arduino. Ils sont donc reliés aux sorties par une résistance (800 Ohms) et à la masse.

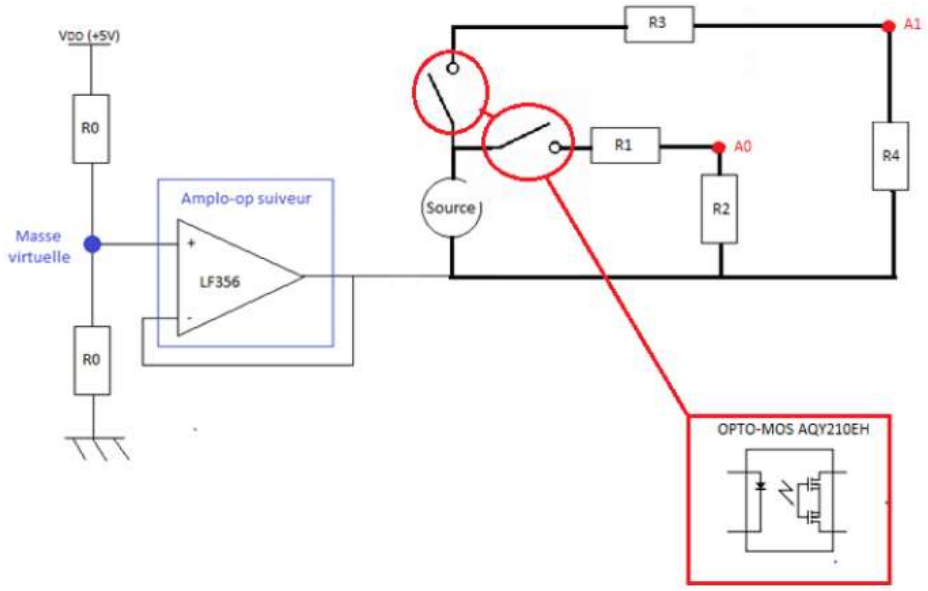
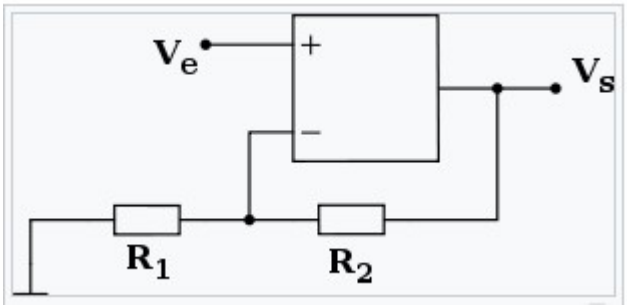


Fig 7: Schéma conditionnement tension (calibres 3.5V et 35V)

Le second mesure quant à lui des tensions dans la plage -350mV <-> 350mV. Son conditionnement requiert donc de multiplier la tension d'entrée cette fois-ci. Nous allons donc utiliser pour son conditionnement un ampli-op amplificateur.



$$V_e = V_s \times \frac{R_1}{R_1 + R_2}$$

Fig 8: Schéma ampli-op amplificateur non-inverseur

En regroupant les 3 calibres, nous obtenons le conditionnement suivant :

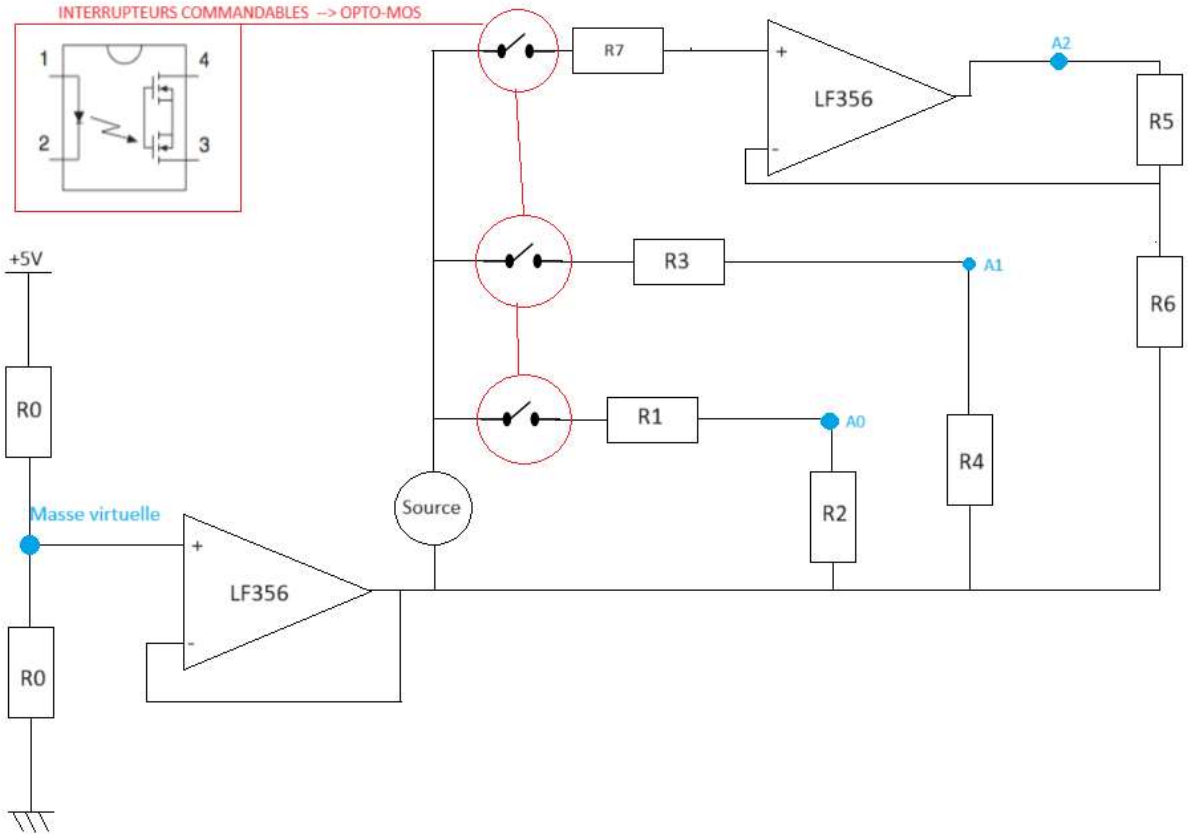
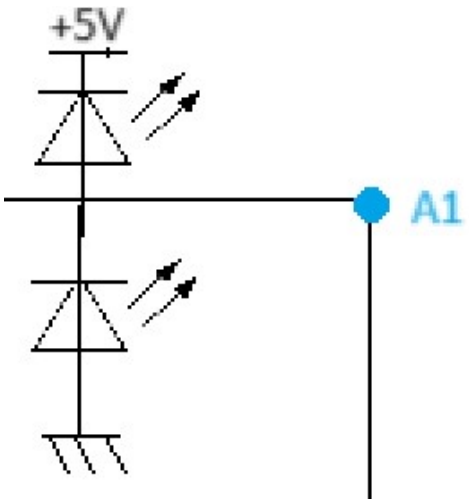


Fig 9: Schéma conditionnement tension (calibres 350 mV, 3.5V et 35V)

Protection montage

La protection du montage consiste à protéger les entrées analogiques de l'Arduino et réorienter le courant en cas de surtension. Pour cela, nous utilisons 2 diodes Zener pour chaque entrée. L'une est reliée à la masse et l'autre au potentiel 5V. Ainsi, le courant sera redirigé dès lorsque la tension aux bornes d'une entrée est supérieure à 5.6V ou inférieure à -0.6V.

Fig 10: Schéma protection entrée analogique



Conditionnement final

Afin de pouvoir router notre carte électronique nous devons penser à limiter le nombre de composants trop nombreux dans la figure précédente. Nous simplifions donc notre conditionnement mais ces fonctionnalités restent les mêmes :

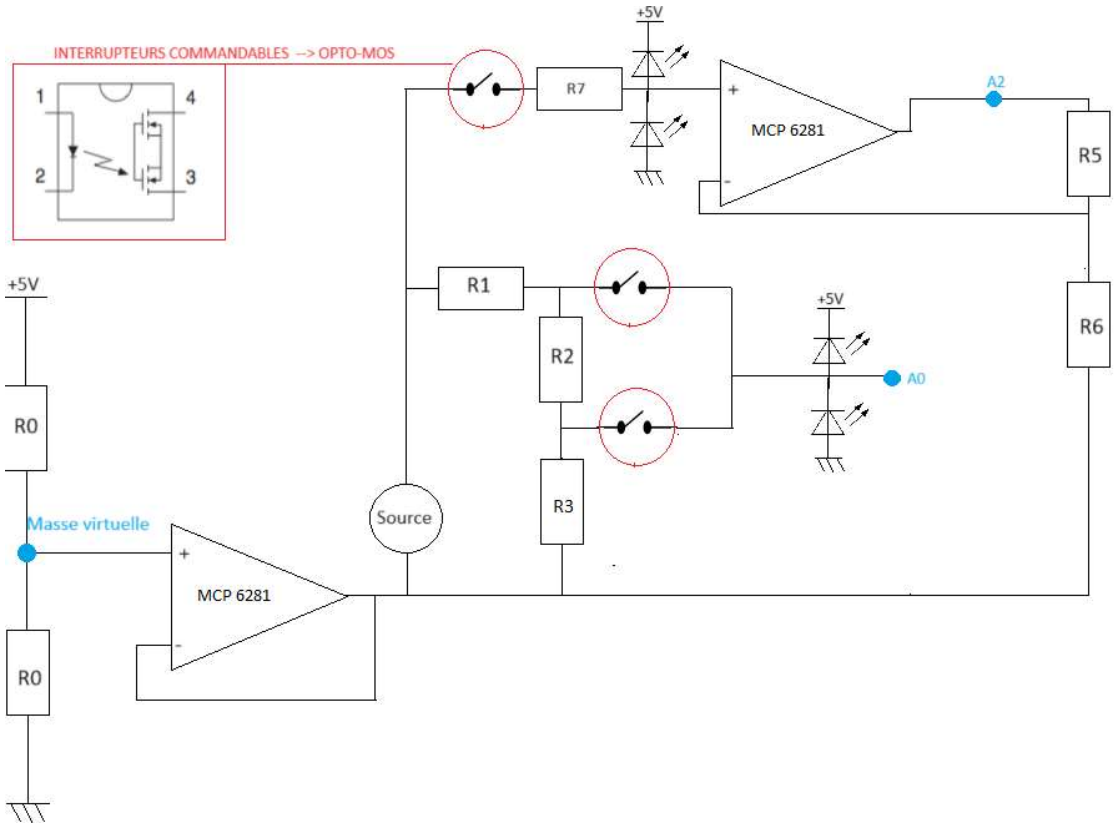


Fig11 : Schéma conditionnement final

PASSERA Léon
POTEAU Mathis

22/12/2023

Les valeurs des composants sont les suivantes :

R0= 1.8 kΩ
R1= 15kΩ
R2=120kΩ
R3=150kΩ
R5= 180 kΩ
R6= 33 kΩ
R7=5.8 kΩ

Résultat final

Une fois le conditionnement optimisé, nous pouvons désormais router le montage sur CADSTAR, l'imprimer et souder les composants. Voici le résultat final :

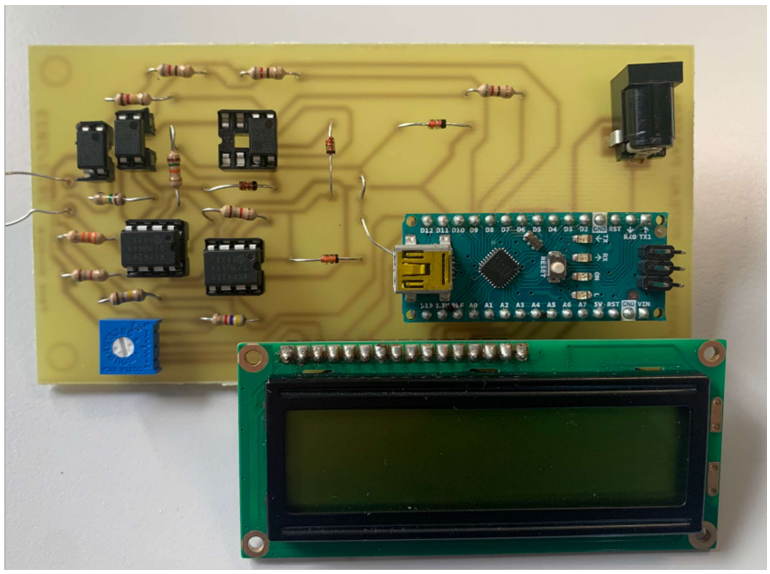


Fig 12: Plaque finale après impression et soudage des composants

Organigramme de programmation

Le code va nous permettre de restituer la vraie valeur de tension initiale sur l'écran LCD. De plus il nous permet de jongler entre les différents calibres utilisables.

Le code complet et l'organigramme de programmation sont disponibles en annexes III et IV .

Simulation sous TinkerCad

Nous simulons notre schéma sous TinkerCad. Voici le lien d'accès au circuit :

[Circuit design Voltmetre PASSERA POTEAU | Tinkercad]

Ps : Le conditionnement effectué sur TinkerCad n'est pas identique au conditionnement final, il s'agit de celui effectué avant le routage. Il n'est pas optimisé mais a les mêmes fonctionnalités que celui final.

Test final

Nous testons dans un premier temps chaque calibre dans sa plage de tension respective :

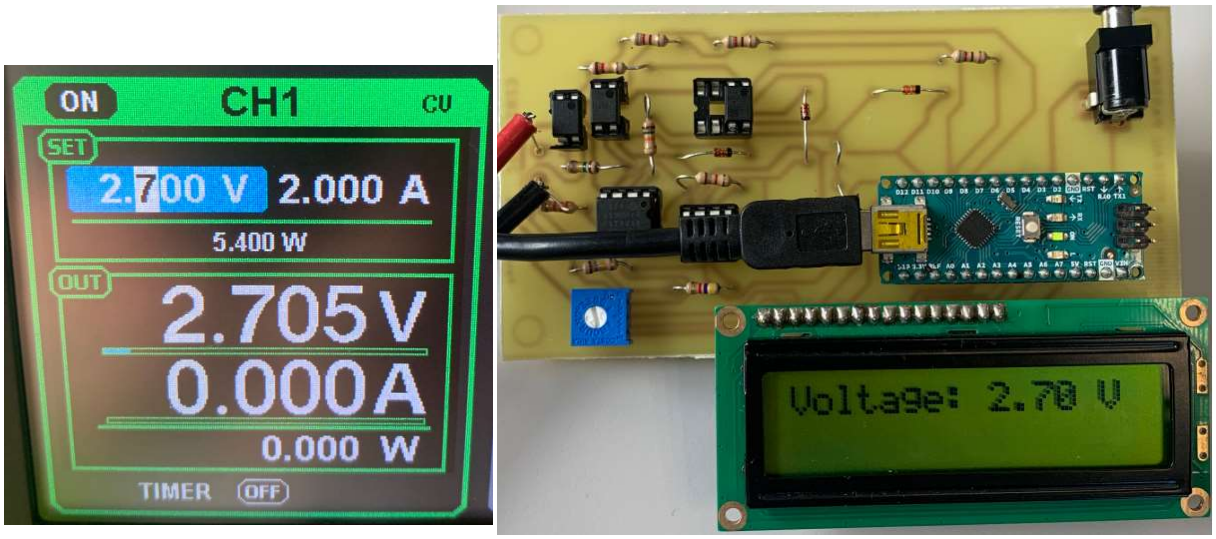


Fig 13: Test pour calibre 3.5 V (V_{in}=2.705V)

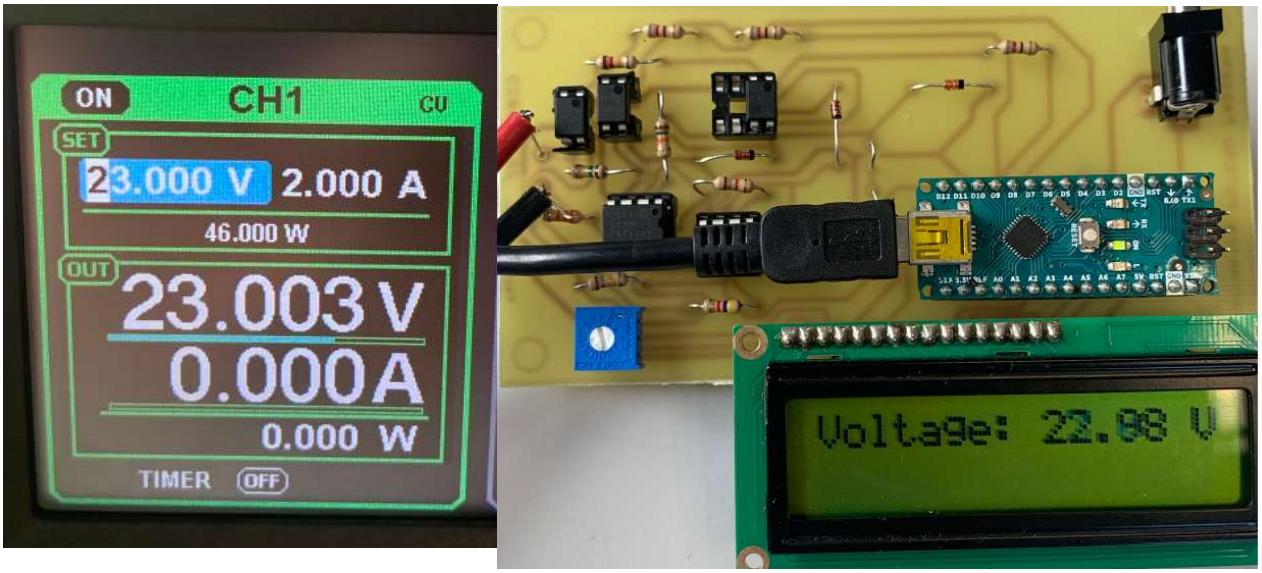


Fig 14: Test pour calibre 35 V (V_{in}=23V)

PASSERA Léon
POTEAU Mathis

22/12/2023

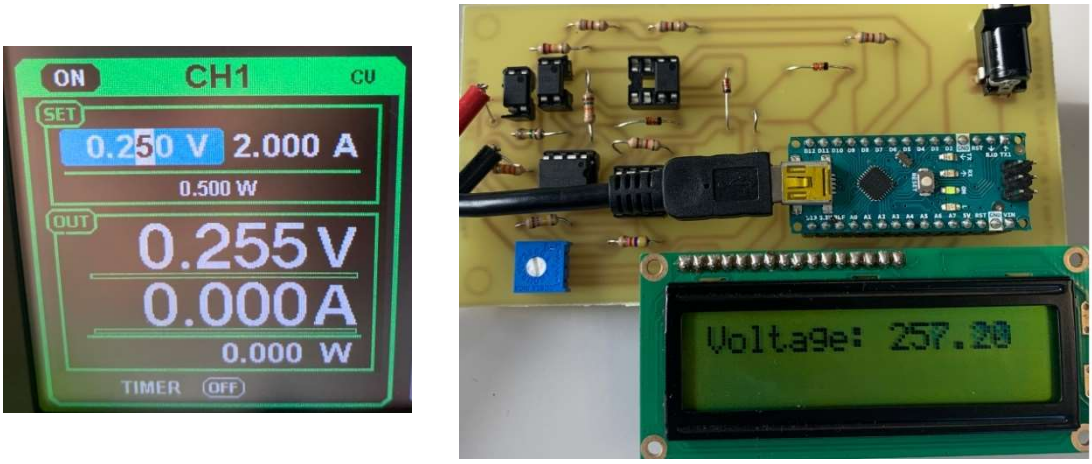
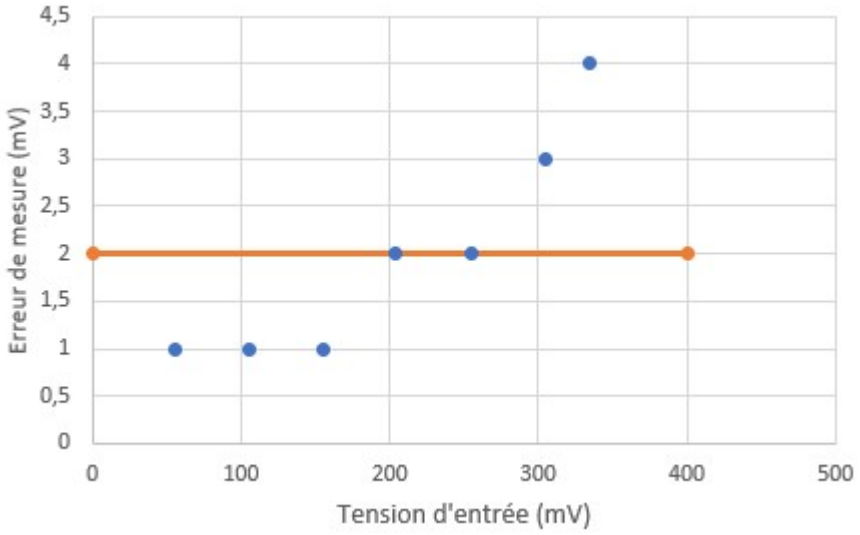


Fig 15: Test pour calibre 350 mV (V_{in}=255mV)

Nous remarquons que la valeur restituée sur l'écran LCD est bien fidèle aux valeurs de tension que l'on souhaite mesurer pour les 3 calibres.

Ensuite, nous traçons pour chaque calibre l'erreur de mesure en fonction de la valeur de tension :

Fig 16: Evolution erreur de mesure calibre 350 mV



PASSERA Léon
POTEAU Mathis

Fig 17: Evolution erreur de mesure calibre 3.5 V

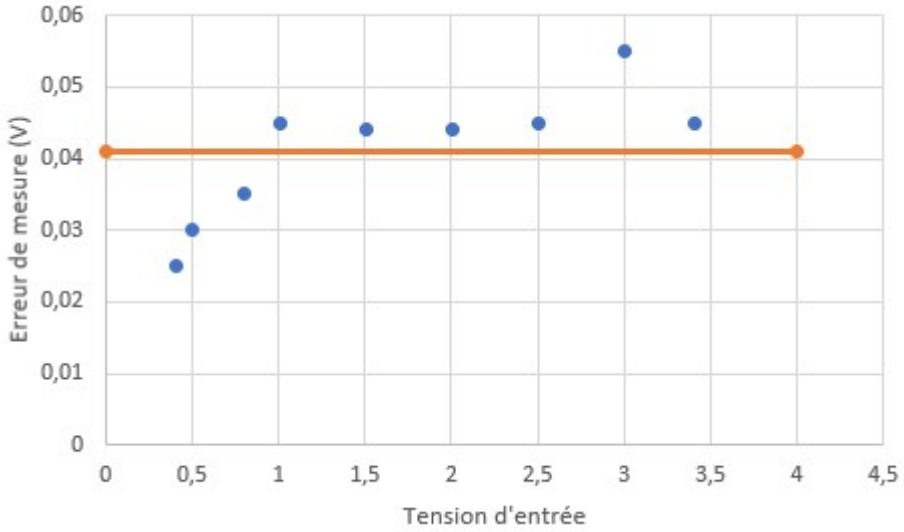
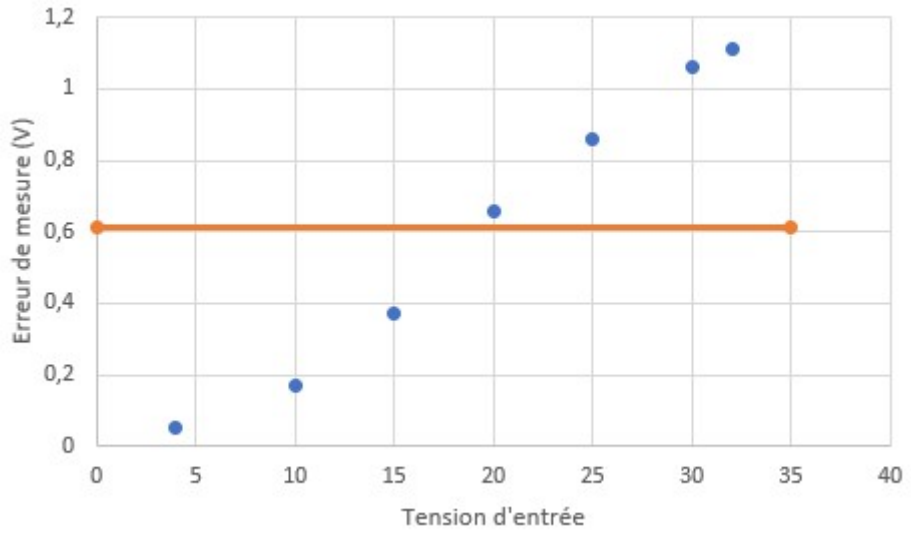
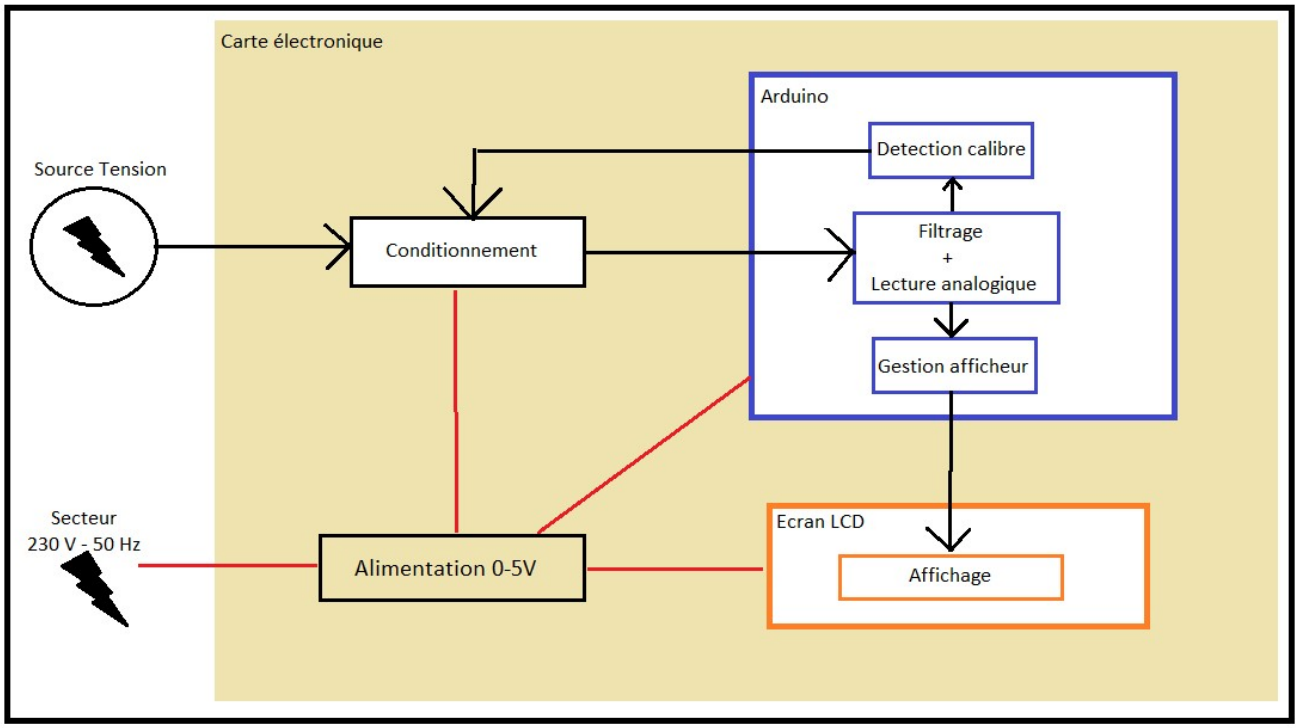


Fig 18: Evolution erreur de mesure calibre 35 V



Ces graphes montrent une logique augmentation de l'erreur de mesure lorsque la tension d'entrée augmente pour les 3 calibres. Il est néanmoins intéressant de comparer les erreurs moyennes (en orange) de chaque calibre. L'erreur moyenne du calibre 350 mV (2 mV) est bien plus faible que celle du calibre 3.5V (40 mV), elle-même bien plus faible que celle du calibre 35V (600 mV). Ces valeurs montrent l'intérêt des petits calibres bien plus fidèles que les gros pour des mesures de faible tension.

Synoptique montage



Conditionnement :

But : Générer une grandeur électrique dépendant du mesurande et exploitable dans Arduino.

Fonction : Modifier la plage de tension en entrée de -35<->35V en plage de tension 0-5V.

Entrée : Tension continue dans la plage -35<->35V.

Sortie : Tension continue dans la plage 0<->5V.

Alimentation : 0-5V

Filtrage + Lecture analogique:

But : S'affranchir des bruits et restituer la valeur numérique d'une grandeur électrique.

Fonction : Effectuer moyenne glissante sur un certain temps et restituer valeur numérique de la tension en entrée.

Entrée : Grandeur électrique (tension) dans la plage 0-5V.

Sortie : Valeurs numériques dans la plage 0-5V.

Alimentation : 0-5V

Détection calibre :

But : Adapter calibre de mesure en fonction de la tension.

Fonction : Orienter la source de tension à mesurer vers le bon conditionnement.

Entrée : Tension mesurée

Sortie : Signaux de commande Opto-Mos

Alimentation : 0-5V

Gestion afficheur

But : Donner l'information d'affichage à l'écran LCD

Fonction : Restituer la tension initiale et les modalités d'affichage

Entrée : Tension lue par lecture analogique dans la plage 0<->5V.

Sortie : Tension initiale dans la plage -35<->35V et modalités d'affichage

Alimentation : 0-5V

Affichage:

But : Restituer à l'utilisateur la valeur souhaitée

Fonction : Affichage d'informations

Entrée : Modalités d'affichage et tension à afficher

Sortie : Affichage final constitué de la valeur numérique et l'unité de mesure.

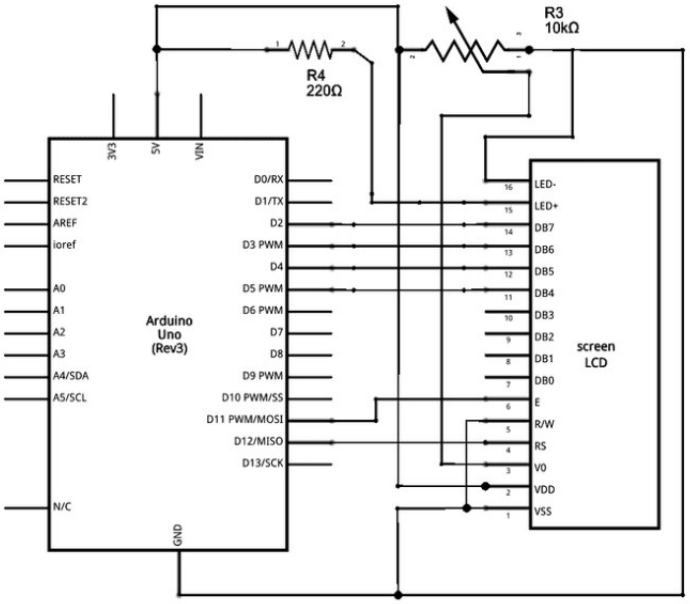
Alimentation : 0-5V

Améliorations possibles

Pour conclure, nous pouvons désormais traiter et conditionner selon 3 calibres (350 mV, 3.5V, 35V) une tension correspondant à la plage exigée par le Cahier des Charges (-35 <-> 35V) afin de l'afficher sur un lecteur LCD. Afin d'améliorer notre projet nous pourrions développer notre code pour traiter les tensions alternatives, ce que nous n'avons pas pu faire par manque de temps.

Annexes

Annexe I : Schéma câblage écran LCD



Annexe II : Matériel utilisé

Logiciels / Sites internet :

TinkerCad (modélisation, simulation montages)
CadStar (modélisation, routage et impression circuits électroniques)
Arduino IDE

Composants électroniques :

Carte microcontrôleur Arduino Nano
Module afficheur 2 lignes de 16 caractères (connectique 1x16)
Ampli-op rail to rail : MCP 6281 (X2)
Relais statiques OptoMos : AQY210EH (X3)
Résistances
Diodes Zener simple (X4)

Autres :

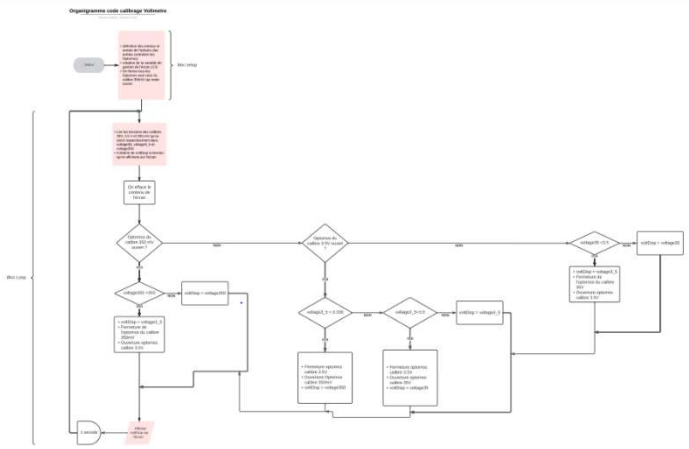
Alimentation stabilisée 9V
Multimètre
Générateur Basses Fréquences
Alimentation Tension continue
Plaque électronique à trous
Oscilloscope

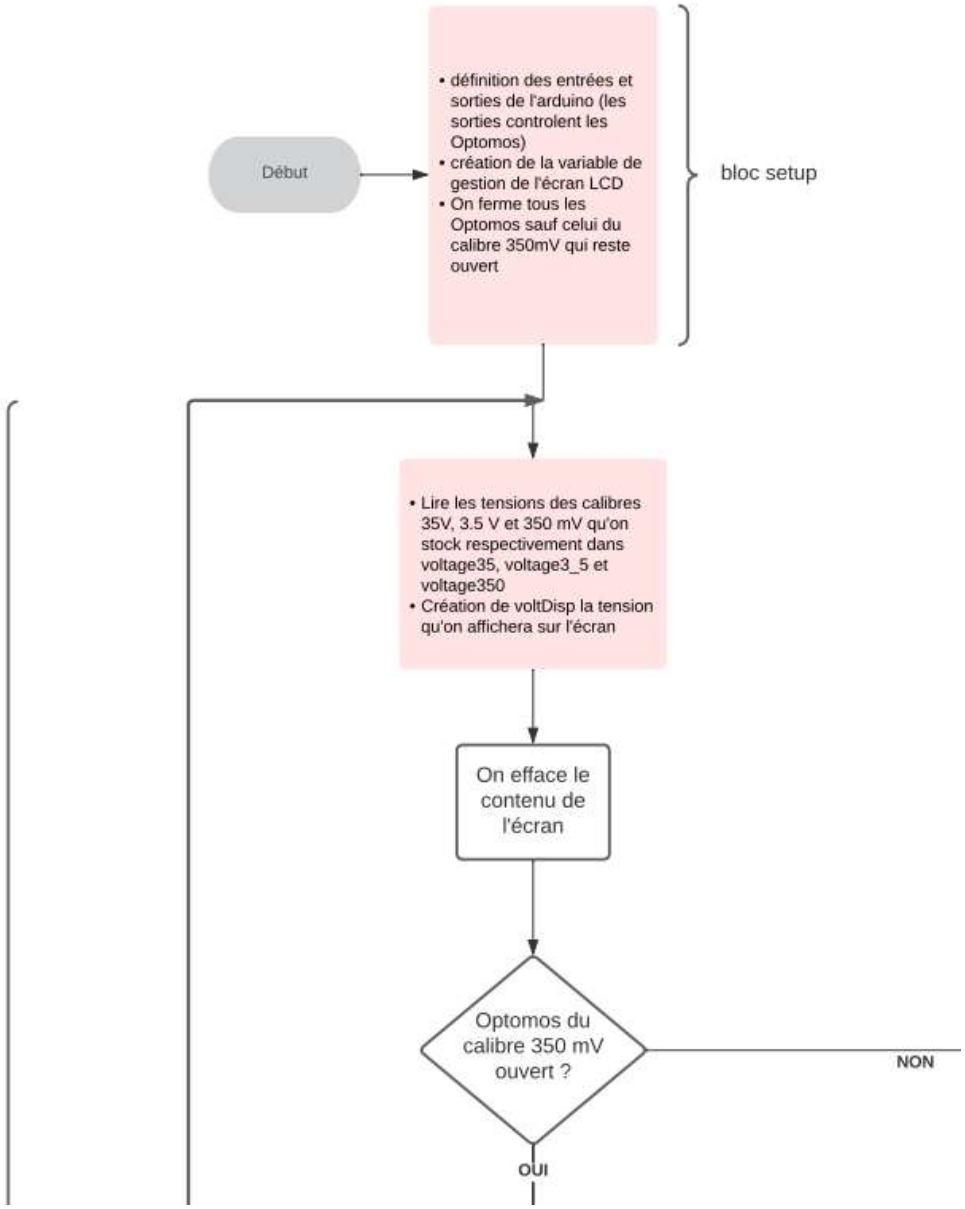
Annexe III : Code Arduino montage final

```
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  digitalWrite(9, LOW);
  digitalWrite(8, LOW);
  digitalWrite(7, HIGH);

}

void loop() {
  // start
  float voltDisp = 0.0;
  String unite = "V";
  int sensorValue3_5 = analogRead(A1);
  int sensorValue35 = analogRead(A0);
  int sensorValue350 = analogRead(A2);
  float voltage3_5 = (sensorValue3_5 * (5.0 / 1023.0)-2.5)*2.11;
  float voltage35 = (sensorValue35 * (5.0 / 1023.0)-2.5)*19 ;
  float voltage350 = (sensorValue350 * (5.0 / 1023.0)-2.5)*0.155*1000;

//Serial.println(unite);
lcd.clear();



if ((digitalRead(7) == HIGH)) {
  if (voltage350 > 350) {
    digitalWrite(7, LOW);
    digitalWrite(8, HIGH);
    voltDisp = voltage3_5;
    unite = "V";
  } else {
    voltDisp = voltage350;
    unite = "mV";
  }
} else if ((digitalRead(8) == HIGH)) {
  if (voltage3_5 < 0.330) {
    digitalWrite(8, LOW);
    digitalWrite(7, HIGH);
    voltDisp = voltage350;
    unite = "mV";
```

```
  } else if (voltage3_5 > 3.5) {
    digitalWrite(8, LOW);
    digitalWrite(9, HIGH);
    voltDisp = voltage35;
    unite = "V";

  } else {
    voltDisp = voltage3_5;
    unite = "V";

  }
} else {
  if (voltage35 < 3.5) {
    digitalWrite(9, LOW);
    digitalWrite(8, HIGH);
    voltDisp = voltage3_5;
    unite = "V";
  } else {
    voltDisp = voltage35;


    unite = "V";
  }


}

lcd.print("Voltage: ");
lcd.print(voltDisp);
lcd.print(unite);


delay(1000);
```

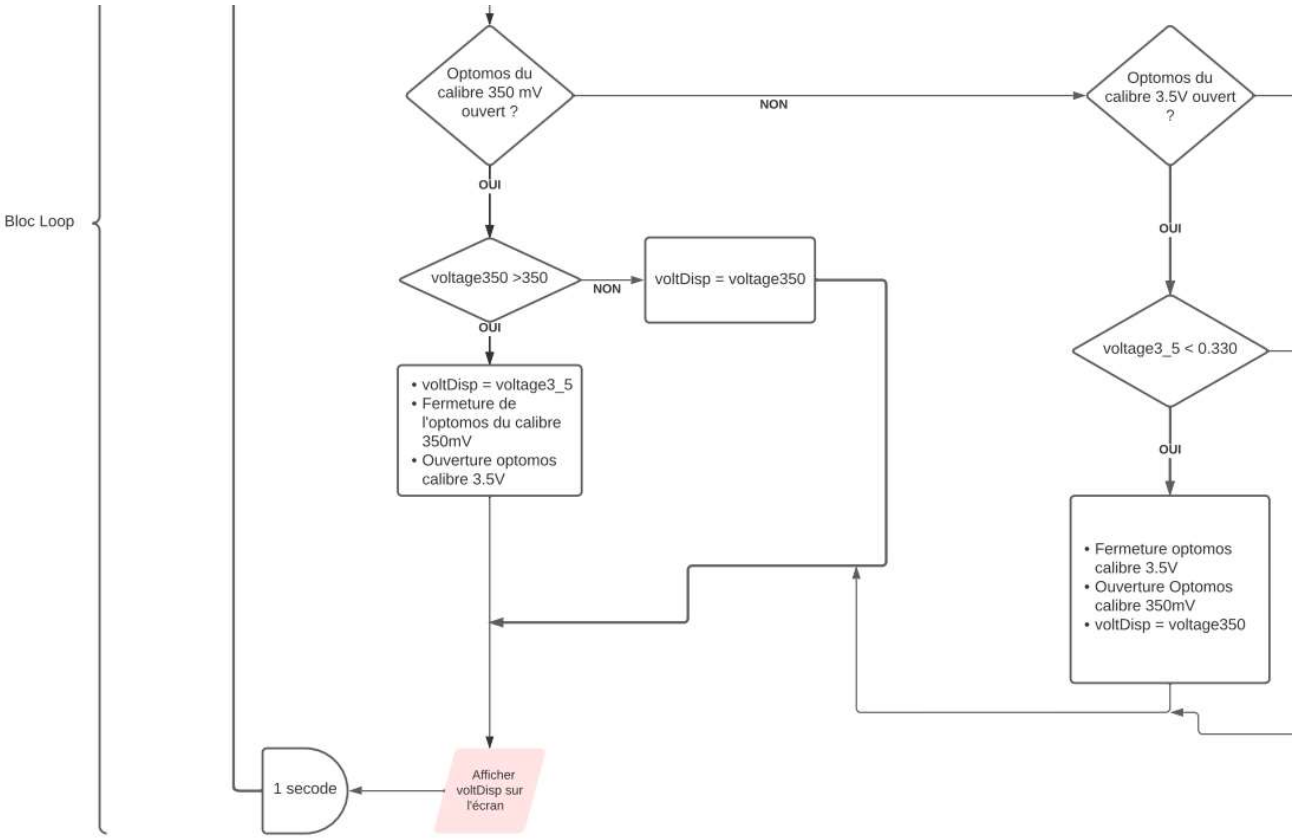
Annexe IV : Organigramme de programmation

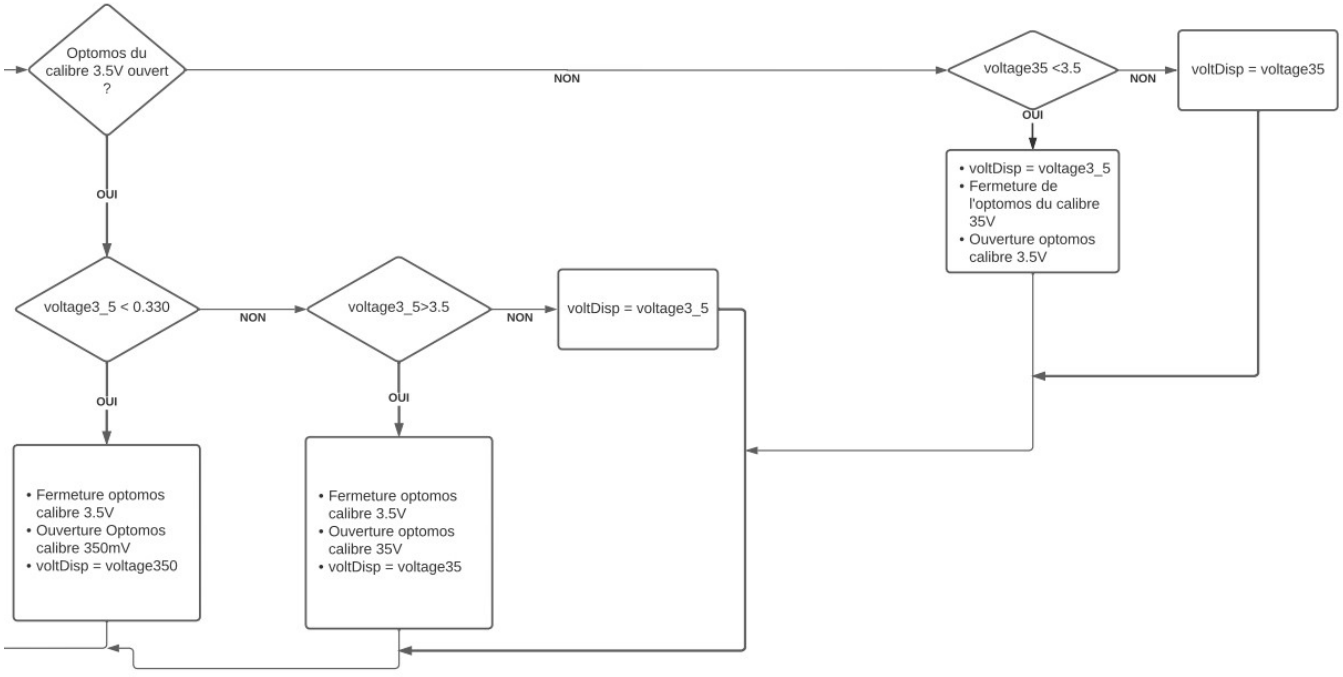




PASSERA Léon
POTEAU Mathis

22/12/2023





PASSERA Léon
POTEAU Mathis

22/12/2023