TP de synthèse

Plan :

I. Rappel du cahier des charges

On souhaite mesurer la valeur moyenne et la valeur efficace de tension de forme quelconque. Pour ne pas perturber le fonctionnement de l'appareil électrique sur lequel on souhaite mesurer cette tension, il faut que l'impédance d'entrée soit supérieure à 100kΩ (idéalement 1MΩ). La tension à mesurer aura une amplitude de ±100Vmax. On souhaite avoir une précision de l'ordre de 5%. Et enfin on veut afficher ces mesures sur un écran LCD toutes les secondes.

II. Principe de fonctionnement

Le figure 1 montre le principe général d'un tel appareil.

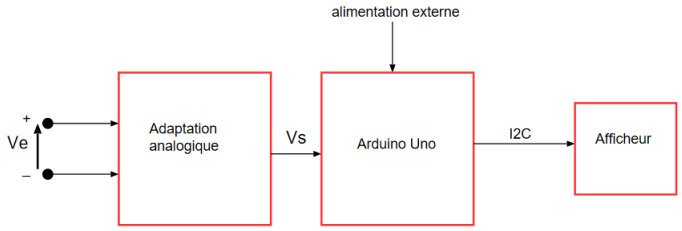

FIGURE 1 – principe général du voltmètre

La partie analogique sert à adapter la tension à mesurer car celle-ci peut être alternative et ainsi avoir une composante négative que l'Arduino ne sait pas gérer. L'Arduino Uno gère la partie numérique. Elle doit ainsi récupérer le signal de sortie de la partie analogique et effectue des calculs afin d'afficher les valeurs moyenne et efficace de la tension d'entrée.

a. Partie analogique

La partie analogique sert comme dit précédemment d'adaptation ainsi la tension de sortie de cette partie analogique doit être comprise entre 0 et Vref. Il faut que l'impédance d'entrée soit élevée et celle de sortie faible. Idéalement il faudrait apporter plusieurs calibres et une protection en cas de surtension.

Ainsi notre but est donc d'avoir :

$Vs = \alpha * Ve + \beta$

Avec :

α : le coefficient du pont diviseur de tension ;

β = Vref/2 : on choisit de prendre Vref = 5V ;

Vs : la tension de sortie de la partie analogique ;

Ve : la tension d'entrée à mesurer

Nous avons donc choisi d'utiliser un montage à AOP combiné à un pont diviseur de tension par calibre (3 calibres : 3V, 30V et 100V). Ce qui donne le montage suivant :

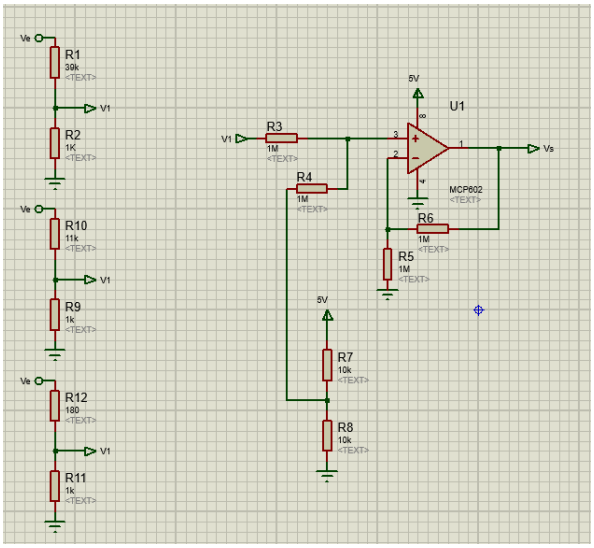

Figure 2 – montage de la partie analogique

On voit bien ici les 3 calibres. N'ayant pas les résistances adéquates les calibres font : 100V, 27,5V et 2,95V. Pour changer de calibres nous n'avons pas réussi à trouver de solution, ainsi il faudra débrancher puis rebrancher un câble entre le pont diviseur et l'entrée du montage à AOP. Ce montage à AOP sert à rehausser la tension afin de n'avoir qu'une composante positive sans supprimer la composante négative de la tension d'entrée. Ainsi l'additionneur additionne la tension de sortie du pont diviseur de tension (V1 dans le schéma qui sera de maximum 2,5V si le calibre est bien choisi) avec la tension Vref/2 qui sera de 2,5V. La tension de sortie de la partie analogique Vs sera envoyé à la broche A0 de la carte Arduino. On aura donc en sortie de la partie analogique un signal diminué et réhaussé comme suit :

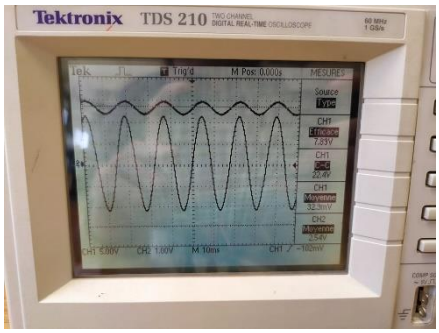

Figure 3 – Signal de sortie de la partie analogique avec un signal d'entrée sinusoïdal

b. Partie numérique

Les objectifs de cette partie sont de pouvoir faire une acquisition de la tension via le CAN de l'Arduino UNO toute les Te=1ms puis au bout de 1000 acquisitions calculer la valeur moyenne et efficace et enfin afficher les valeurs sur un écran LCD.

Afin de réaliser la partie numérique nous nous sommes d'abord intéressés à la période d'échantillonnage. Pour cela nous avons décidé d'utiliser les interruptions et le TIMER1. Le principe est que le TIMER1 va compter à une fréquence que nous allons lui imposer et à chaque fois que le TIMER1 sera en overflow donc quelle aura atteint sa valeur max de comptage cela déclenchera une interruption qui remettra le TIMER1 à son état initial avant le comptage.

Le CPU de l'Arduino UNO a une fréquence de 16Mhz et le TIMER1 et un timer sur 16 bit donc il peut compter au maximum jusqu'à 65536. Nous voulons une période Te de 1ms donc une fréquence de 1Khz. Comme nous voulons des interruptions sur overflow nous devons pré remplir le TIMER1 avec la valeur que nous allons calculer en utilisant la formule suivante.

$$X = 65536 – 16Mhz/Prescaler/FreqVoulu$$

Avec un prescaler de 1 et une fréquence de 1Khz nous obtenons 49536.

Nous avons la configuration suivante :

```
noInterrupts(); //Désactivation des interruptions
  TCCR1A = 0;//Remise a 0 des bits de réglage TCCR1A
  TCCR1B = 0;//Remise a 0 des bits de réglage TCCR1B

  TCNT1 = 49536;// Prérempli le timer 65536-16MHz/1/1KHz
  TCCR1B |=(1<<CS10);//Prescaler de 1
  TIMSK1 |= (1 << TOIE1);//Activation des interruptions sur overflow
  interrupts();// Activation des interruptions

}
  ISR(TIMER1_OVF_vect)//Service d'interruption sur overflow
  {
    if(digitalRead(ledPin)){ //Clignotement LED
      digitalWrite(ledPin,LOW);
    }
    else{
      digitalWrite(ledPin,HIGH);
    }

    TCNT1 = 49536;// Prérempli le timer (etat inital)

  }
```

Figure 3 - Code interrupts et timer

Nous avons testé à l'aide d'un oscilloscope la période d'inversion du signal nous avons obtenu le signal suivant qui nous montre bien une période Te de 1ms.

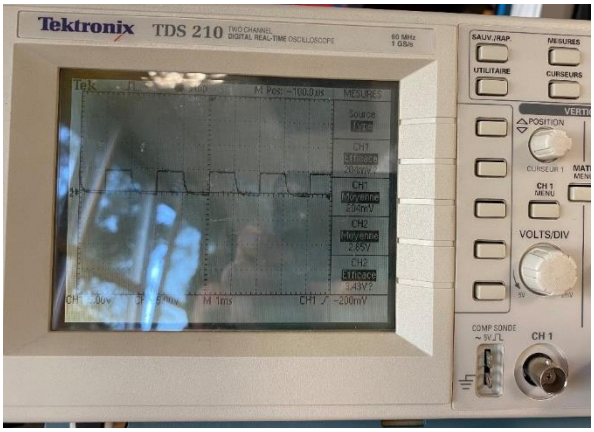

Figure 4 - Vérification du temps de 1ms

Pour réaliser les calibres nous avons utilisé des boutons déclarés en entrée et des if. Lorsque le programme détecte qu'un bouton est appuyé cela modifie la valeur des coefficients appliqués à la tension mesurée.

```
//TEST des boutons pour les calibres
if(digitalRead(BUTTON_PIN1)== HIGH){
  R1 = 180.0;//Calibre 3V
  str3[0]='1';//Valeur a afficher a côté de calibre
}
if(digitalRead(BUTTON_PIN2)== HIGH){
  R1 = 11000.0;//Calibre 30V
  str3[0]='2';//Valeur a afficher a côté de calibre

}
if(digitalRead(BUTTON_PIN3)== HIGH){
  R1 =39000.0;//Calibre,100V
  str3[0]='3';//Valeur a afficher a côté de calibre
}
```

Figure 5 – Code pour gérer les calibres

Pour l'acquisition de la valeur de la tension nous avons utilisé une variable flag qui est à 0 et qui passe à 1 à chaque interruption. Quand le flag est à 1 nous utilisons la fonction analogRead() dans le programme principal pour lire la valeur numérique sur la broche analogique. On utilise ensuite cette formule afin d'avoir la valeur de la tension

$$V_s = \frac{V_{num}}{2^N} V_{ref}$$

Une fois Vs obtenu on applique le bon coefficient par rapport au calibre et on applique la formule de la moyenne ou de la valeur efficace on ajoute ensuite le résultat obtenu dans la bonne variable ResultatMoy ResultatEff

Pour l'affichage nous avons simplement repris le programme de base fournie dans le sujet du TP et l'avons adapté afin d'afficher les

2 résultats. Lorsque le nombre d'acquisition atteint 1000 on divise ResultatMoy et ResultatEff par 1000 et on affiche le résultat sur le LCD.

```
//Une interruption a eu lieu le flag est a 1
if(flag ==1){
  if(nbr_Aquisition<1000){
    //Aquisition de Vnum et conversion en Vs
     mesure = analogRead(ANALOG_PIN)*(VREF/1023.0);
     //On applique les coefficients
     mesure = ((mesure - VREF/2)*Req);
     //Ajout du resultat pour la moyenne et la valeur efficace
     resultatMoy = resultatMoy+ mesure;
     resultatEff = resultatEff+(mesure*mesure);
     nbr_Aquisition ++;
  }
  else{

    //Calcule de la valeur efficace et de la valeur moyenne
    resultatMoy = resultatMoy/nbr_Aquisition;
    resultatEff = sqrt(resultatEff/nbr_Aquisition);

    //Conversion des resultat en chaine de caractere
    dtostrf(resultatMoy,5,5,str1);
    dtostrf(resultatEff,5,5,str2);

     //Affichage
    u8g.firstPage();
  do {
    u8g.drawStr( 0, 10, "Valeur moyenne:");
    u8g.drawStr( 0, 25, str1);
    //u8g.drawStr(55,25,"V");
    u8g.drawStr( 0, 40, "Valeur eff:");
    u8g.drawStr( 0, 55, str2);
    //u8g.drawStr(55,55,"V");
    u8g.drawStr(95, 55, "Cal");
    u8g.drawStr(118, 55, str3);
  } while( u8g.nextPage());

    nbr_Aquisition = 0;
  }
}
flag =0;//Remise a 0 du flag

}
```

Figure 6 – Code du calcul de la mesure et de l'affichage

III. Résultats

Ainsi nous avons obtenus le résultat voulu (affichage de la valeur moyenne et efficace de la tension d'entrée). Cependant tout le cahier des charges n'a pas été respecté : il n'y a pas de protection contre la surtension et les calibres ne sont pas automatiques, il faut changer un câble.

IV. Annexe

a. Montage final et PCB

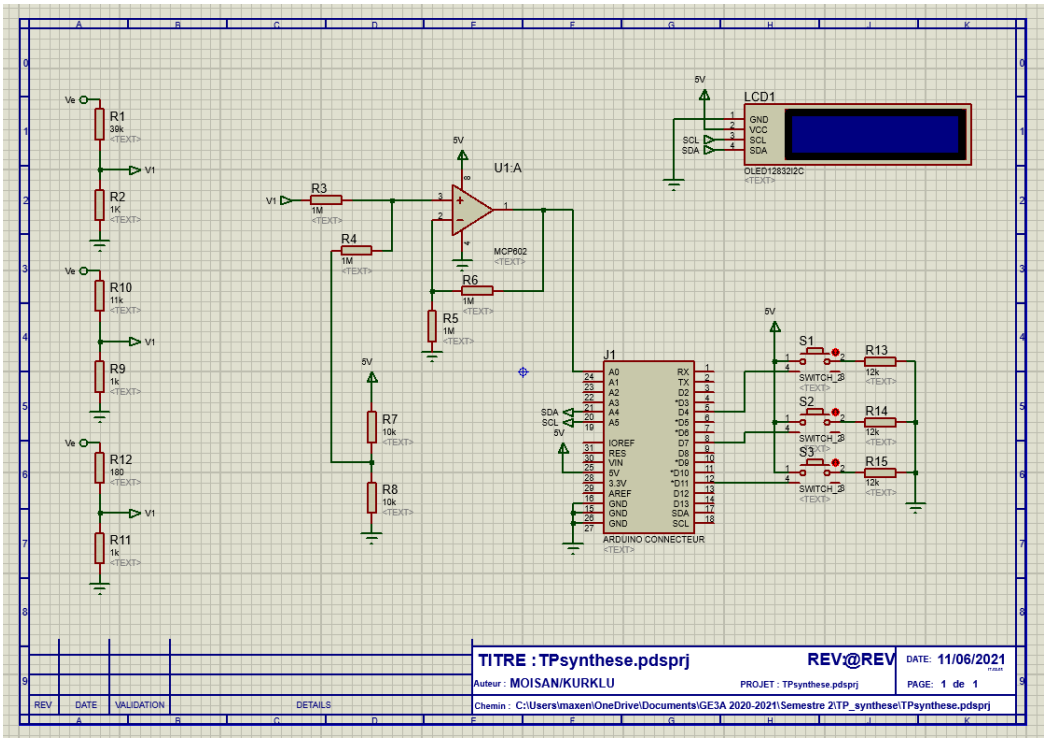

Figure 7 – Montage complet

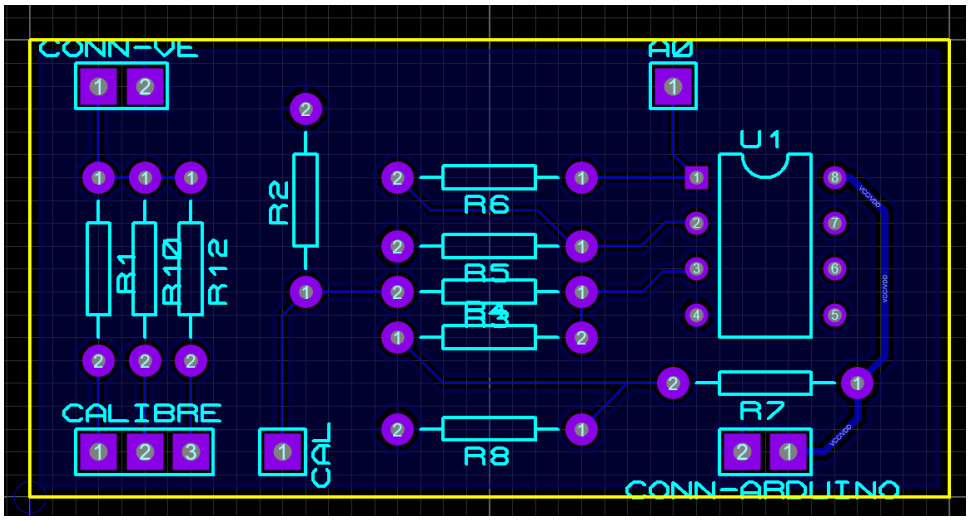

Figure 8 – PCB de la partie analogique

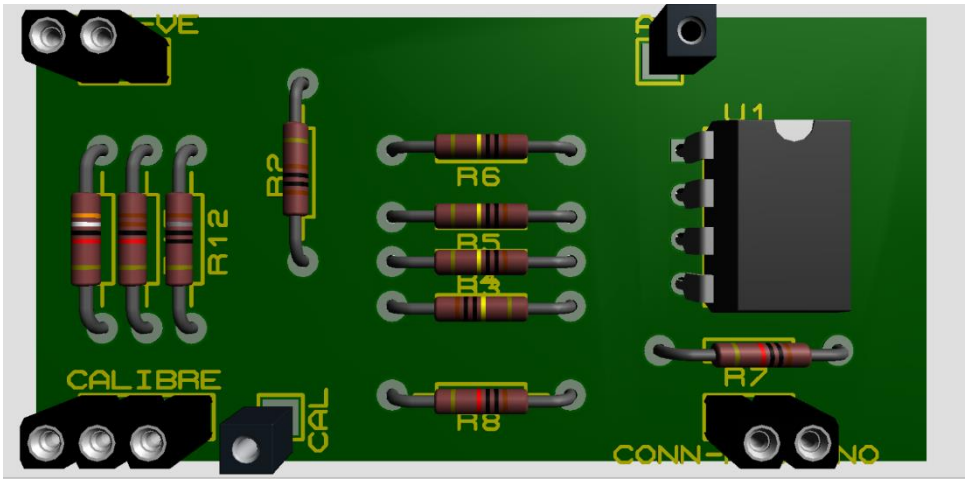

Figure 9 – Vue 3D de la face composant

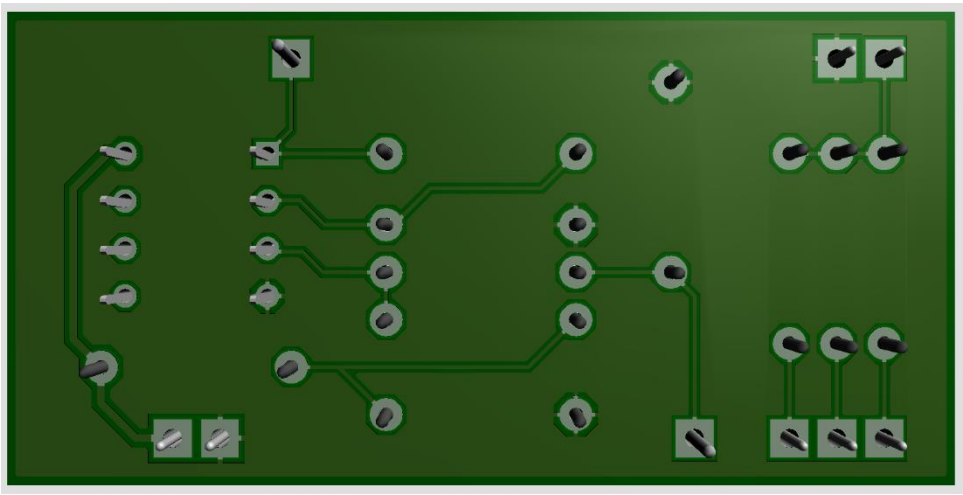

Figure 10 – Vue 3D de la face de dessous

b. Code

```
#include "U8glib.h"
#include <math.h>
#define VREF 5.0
#define ANALOG_PIN 0

#define BUTTON_PIN1 12
#define BUTTON_PIN2 8
#define BUTTON_PIN3 5

float Req;
float R1;
float R2=1000.0;
int flag =0;

//CONFIGURATION
U8GLIB_SH1106_128X64 u8g(U8G_I2C_OPT_DEV_0|U8G_I2C_OPT_FAST);

void setup() {
  /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```

```
  u8g.setColorIndex(255);// white
  u8g.setFont(u8g_font_unifont);
  analogReference(DEFAULT);
  pinMode(BUTTON_PIN1,INPUT);
  pinMode(BUTTON_PIN2,INPUT);
  pinMode(BUTTON_PIN3,INPUT);

  /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

  noInterrupts(); //Désactivation des interruptions
  TCCR1A = 0;//Remise a 0 des bits de réglage TCCR1A
  TCCR1B = 0;//Remise a 0 des bits de réglage TCCR1B

  TCNT1 = 49536;// Prérempli le timer 65536-16MHz/1/1KHz
  TCCR1B |=(1<<CS10);//Prescaler de 1
  TIMSK1 |= (1 << TOIE1);//Activation des interruptions sur overflow
  interrupts();// Activation des interruptions



}
  ISR(TIMER1_OVF_vect) //Service d'interruption sur overflow
  {
    flag = 1;
    TCNT1 = 49536;// Prérempli le timer (etat inital)
  }

void loop() {

  //Déclaration des variables
  static unsigned short nbr_Aquisition =0 ;
  static float resultatMoy = 0;
  static float resultatEff = 0;
  static float mesure =0;
  static char str1[11];
  static char str2[11];
  static char str3[1];

  //TEST des boutons pour les calibres
  if(digitalRead(BUTTON_PIN1)== HIGH){
    R1 = 180.0;//Calibre 3V
    str3[0]='1';//Valeur a afficher a côté de calibre
  }
  if(digitalRead(BUTTON_PIN2)== HIGH){
    R1 = 11000.0;//Calibre 30V
    str3[0]='2';//Valeur a afficher a côté de calibre

  }
  if(digitalRead(BUTTON_PIN3)== HIGH){
    R1 =39000.0;//Calibre,100V
    str3[0]='3';//Valeur a afficher a côté de calibre
  }

  Req = (R1+R2)/R2;

  //Une interruption a eu lieu le flag est a 1
  if(flag ==1){
    if(nbr_Aquisition<1000){
      //Aquisition de Vnum et conversion en Vs
       mesure = analogRead(ANALOG_PIN)*(VREF/1023.0);
       //On applique les coefficients
       mesure = ((mesure - VREF/2)*Req);
       //Ajout du resultat pour la moyenne et la valeur efficace
       resultatMoy = resultatMoy+ mesure;
       resultatEff = resultatEff+(mesure*mesure);
       nbr_Aquisition ++;
    }
    else{

      //Calcule de la valeur efficace et de la valeur moyenne
      resultatMoy = resultatMoy/nbr_Aquisition;
      resultatEff = sqrt(resultatEff/nbr_Aquisition);

      //Conversion des resultat en chaine de caractere
```

```
      dtostrf(resultatMoy,5,5,str1);
      dtostrf(resultatEff,5,5,str2);

       //Affichage
      u8g.firstPage();
    do {
      u8g.drawStr( 0, 10, "Valeur moyenne:");
      u8g.drawStr( 0, 25, str1);
      //u8g.drawStr(55,25,"V");
      u8g.drawStr( 0, 40, "Valeur eff:");
      u8g.drawStr( 0, 55, str2);
      //u8g.drawStr(55,55,"V");
      u8g.drawStr(95, 55, "Cal");
      u8g.drawStr(118, 55, str3);
    } while( u8g.nextPage());

      nbr_Aquisition = 0;
    }
  }
  flag =0;//Remise a 0 du flag

}
```