```
/* ************************************************************************** */
/** Descriptive File Name

  @Company
    Company Name

  @File Name
    filename.c

  @Summary
    Brief description of the file.

  @Description
    Describe the purpose of this file.
 */
/* ************************************************************************** */

/* ************************************************************************** */
/* ************************************************************************** */
/* Section: Included Files                                                    */
/* ************************************************************************** */
/* ************************************************************************** */

/* This section lists the other files that are included in this file.
 */

/* TODO:  Include other files here if needed. */

#include "app.h"
#include "Gest_LED.h"
#include "Gestion_Filtre.h"
/* ************************************************************************** */
/* ************************************************************************** */
/* Section: File Scope or Global Data                                         */
/* ************************************************************************** */
/* ************************************************************************** */

//fonction :Value_search
//EntrÃ©e: uint16_t t_Values [NOMBRE_ECH]
//sortie : uint16_t *Value_max,uint16_t *Value_min, uint16_t *Val_Zero
//description :permet de lire l'entiÃ¨retÃ© du signal pour dÃ©terminer
//la valeur min et la valeur max et l'offset

void Value_search(uint16_t t_Values [NOMBRE_ECH],uint16_t *Value_max,
                    uint16_t *Value_min,uint16_t *Val_Zero)
{
     uint16_t Ech_mem = 0;
    uint32_t Index = 0;

     //mise d'une valeur min pour Ã©viter de tomber sur 0
     *Value_min = t_Values[Index];
     //calcul pour avoir la valeur du 0 pour mesurer la tension ainsi que
```

```
     //la fréquence
    for (Index = 0;Index < NOMBRE_ECH; Index++)
    {

        if (t_Values[Index] > Ech_mem)
        {
            if (t_Values[Index]> *Value_max)
            {
                //recherche de la valeur max du signal
                *Value_max = t_Values[Index];
            }
        }
            if (t_Values[Index]< *Value_min)
            {
                //protexion si erreur de lecture
                if(t_Values[Index] != 0)
                {
    //recherche de la valeur min du signal
                    *Value_min = t_Values[Index];
                }
            }
        //mÃ©moire Ã©chantillons prÃ©cÃ©dent
        Ech_mem = t_Values[Index];
    }
    //calcul de la valeur du centre
    *Val_Zero = ((*Value_max - *Value_min)/2)+*Value_min;
}



//fonction :Conv_Values
//EntrÃ©e: uint16_t t_Values [NOMBRE_ECH],uint16_t Val_Zero
//sortie :  uint16_t *Nb_ech_pos
//description : permet de compter le nombre d'Ã©chantillons lors du
//deuxiÃ¨me demi-sinus positif

void Conv_Values (uint16_t t_Values [NOMBRE_ECH],
                    uint16_t *Nb_ech_pos,uint16_t Val_Zero )
{
    uint32_t Index = 0;

    //flag une frÃ©quence est mesurÃ©e
    bool Frequ_Mes = false;
    //flag les premier Ã©chantillons sont pousitif ou negatid
    bool first_Value = false;
    //flag pour lire que une frÃ©quence
    bool One_Time = false;

    for (Index = 0;Index < NOMBRE_ECH; Index++)
    {
        //si la valeur du tableau est plus grande que le zero theorique
        if (t_Values[Index] >= Val_Zero)
```

```
        {
            if(first_Value == true)
            {
               if (One_Time == false)
               {
                 Frequ_Mes = true;
                *Nb_ech_pos =*Nb_ech_pos + 1 ;
               }
            }
        }
        else
        {
            first_Value = true;

            if (Frequ_Mes == true)
            {
               if (One_Time == false)
               {
                   One_Time = true;
                   Frequ_Mes = false;
               }
            }
        }
    }
}

//fonction :Calcul_Frequence_Led
//EntrÃ©e: uint16_t Nb_ech,uint16_t Tension_max
//sortie :  -
//description :permet de dÃ©terminer les LEDs Ã  allumer ainsi que la
//couleur par apport Ã  la frÃ©quence

void Calcul_Frequence_Led (uint16_t Nb_ech,uint16_t Tension_max)
{

    //infomation de la fr quence mise en variable pour ainsi
    //les modifier avec la partie UART
    uint16_t _20HZ   = 5000;
    uint16_t _50HZ   = 2000;
    uint16_t _100HZ  = 1000;
    uint16_t _200HZ  = 500;
    uint16_t _500HZ  = 200;
    uint16_t _1KHZ   = 100;
    uint16_t _2KHZ   = 50;
    uint16_t _5KHZ   =  20;
    uint16_t _10KHZ  =  10;
    uint16_t _20KHZ  =  5;

    uint8_t Color  = 0;

    //choix de la couleur par apport   la tension
    if (Tension_max > MIN_DB )
```

```
    {
        Color  = GREEN;
        if (Tension_max > MOY_DB)
        {
            Color  = YELLOW;
            if (Tension_max > MAX_DB)
            {
                Color  = RED;
            }
        }
    }

    All_LED_Off();
    //choix de la fr quence
    if (Nb_ech < _20HZ)
    {
       Gest_LED(LED0,Color,true);
    }
    if (Nb_ech < _50HZ)
    {
        Gest_LED(LED1,Color,true);
    }
    if (Nb_ech < _100HZ)
    {
         Gest_LED(LED2,Color,true);
    }
    if (Nb_ech < _200HZ)
    {
        Gest_LED(LED3,Color,true);
    }
    if (Nb_ech < _500HZ)
    {
        Gest_LED(LED4,Color,true);
    }
    if (Nb_ech < _1KHZ)
    {
        Gest_LED(LED5,Color,true);
    }
    if(Nb_ech < _2KHZ)
    {
        Gest_LED(LED6,Color,true);
    }
    if(Nb_ech < _5KHZ)
    {
        Gest_LED(LED7,Color,true);
    }
    if(Nb_ech < _10KHZ)
    {
        Gest_LED(LED8,Color,true);
    }
    if (Nb_ech < _20KHZ)
    {
```

```
        Gest_LED(LED9,Color,true);
    }
}

/* ************************************************************************** */
/* ************************************************************************** */
// Section: Interface Functions                                               */
/* ************************************************************************** */
/* ************************************************************************** */

/*  A brief description of a section can be given directly below the section
    banner.
 */

// ****************************************************************************

/**
  @Function
    int ExampleInterfaceFunctionName ( int param1, int param2 )

  @Summary
    Brief one-line description of the function.

  @Remarks
    Refer to the example_file.h interface header for function usage details.
 */



/* ************************************************************************** */
 End of File
 */
```