```
/* -REGULATEUR DE CHARGE ALTERNATEUR- Version 6 FINAL.
 *  Mise enservice>tempo 5 Min> si superieur  à 14.2v>Tempo 5 min> passage en float.
 *  2 MosFets pour sortie Servitude et 1 MosFet pour sortie Moteur.
 *  Un relays pour charque sorties pour le mode Bulk.
 * Fonction tempo de Bulk vers Float
 * Affichage LCD pour les lectures instantanées des tensions.
 * Lectures des tensions moyennées sur 20 mesures.
 * Mode LEDs bicolores
 * Régulation de tension en mode Float asservies sur lesctures de tensions
 */


// Déclaration des constantes:
const int mosfetServ = 3; //Pin du pilotage mosfet servitude.
const int mosfetMot = 5;  //Pin du pilotage mosfet moteur.
const int ledOn = 12;     //Pin de la LED verte "mise en fonction du systeme".
const int ledServB = 11;  //Pin de la LED bleue "mode charge "Float" batteries servitude".
const int ledServV = 10;  //Pin de la LED verte "mode charge "Bulk" batteries servitude".
const int ledMotB = 9;    //Pin de la Led bleue "mode charge "Float" batterie moteur".
const int ledMotV = 8;    //Pin de la Led verte "mode charge "Bulk" batterie moteur".
const int uServRead = A1; //Pin de lecture de la tension de servitude.
const int uMotRead = A0;  //pin de lecture de la tension de moteur.
const int RelayMot = 7;   //pin du pilotage ralais moteur.
const int RelayServ = 6;  //Pin du pilotage relais srvitude.

const long timeToFloat = 300000;  //interval de temps du mode Bulk au mode Float. 5min
```

```
const int nEchantillons = 20;    //nombre d'echantillons pour moyenner.


//Declaration des variables:


//etat des modes:
boolean ServState = 0;
boolean MotState = 0;


byte mosfetServ_drive;      //Valeur de pilotage mosfet servitude.
byte mosfetMot_drive;       //Valeur de pilotage mosfet moteur.


int echantillonS[nEchantillons];    // déclaration du tableautableau pour stocker les échantillons lus
int indiceS = 0;                // l'indice de l'échantillon courant Service
float totalS = 0;               // la somme des échantillons mémorisés Service
float moyenneS = 0;             // la moyenne des échantillons mémorisés Service
int echantillonM[nEchantillons];    // un tableau pour stocker les échantillons lus Moteur
int indiceM = 0;                // l'indice de l'échantillon courant Moteur
float totalM = 0;               // la somme des échantillons mémorisés Moteur
float moyenneM = 0;             // la moyenne des échantillons mémorisés Moteur


//Prise en charge du module LCD I2C:
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>//Invite des modules LCD
LiquidCrystal_PCF8574 lcd(0x27);  // set the LCD address to 0x27



void setup() {
```

```
//initialisation du LCD display
Wire.begin();
Wire.beginTransmission(0x27);
lcd.begin(16, 2);

//incrementation du tableau des moyennes:
for (int iS = 0; iS < nEchantillons; iS++) {
   echantillonS[iS] = 0;
}
for (int iM = 0; iM < nEchantillons; iM++) {
   echantillonM[iM] = 0;
}
//Declaration des sorties:
pinMode(mosfetServ, OUTPUT);
pinMode(mosfetMot, OUTPUT);
pinMode(ledOn, OUTPUT);
pinMode(ledServB,OUTPUT);
pinMode(ledServV,OUTPUT);
pinMode(ledMotB,OUTPUT);
pinMode(ledMotV,OUTPUT);
pinMode(RelayServ,OUTPUT);
pinMode(RelayMot,OUTPUT);

//Affichage du texte au demarrage LCD.
 lcd.setBacklight(01);   //allumage du retroéclairage LCD
 lcd.home(); lcd.clear();
```

```
 lcd.setCursor(0,0);
 lcd.print("Mise en service");
 lcd.setCursor(0,1);
 lcd.print("Tempo. 5 Minutes");

//Sequence de mise en fonction:

digitalWrite(ledOn, LOW);          //mise en marche de la LED "mise en fonction".
digitalWrite(ledServB, LOW);       // LED bleue servitude.
digitalWrite(ledServV, LOW);       // LED2 verte servitude.
digitalWrite(ledMotB, LOW);        // LED bleue moteur.
digitalWrite(ledMotV, LOW);        // LED verte moteur.
analogWrite(mosfetServ,255);       //mosfet de servitude piloté en mode Float.
analogWrite(mosfetMot,255);        //mosfet de servitude piloté en mode Bulk.
digitalWrite(RelayServ,HIGH);      //Relais servitude piloté en mode float.
digitalWrite(RelayMot,HIGH);       //Relais mot piloté en mode Bulk.

analogRead(uServRead);    //lecture de la tension de service.
analogRead(uMotRead);     //lecture de la tension de service.

delay(300000);                    //Temporisation de  Secondes.

//Réinitialisation a off des LEDs de sorties:
digitalWrite(ledServB, HIGH);
digitalWrite(ledServV, HIGH);
digitalWrite(ledMotB, HIGH);
digitalWrite(ledMotV, HIGH);
```

```
}

void loop() {
  //Mise en boucle de toutes fonstions en simultané:
  task_lcd();
  task_serv();
  task_mot();
}

void task_lcd() {

  //Moyennage de lecture de tension Servitude:
  totalS = totalS - echantillonS[indiceS];      // Soustraction de l'echantillon précédent
  echantillonS[indiceS] = analogRead(uServRead);// Lecture du capteur
  totalS = totalS + echantillonS[indiceS];      // Ajout du dernier echantillon
  indiceS++;// Incrémentation de l'indice
  // si on est à la fin du tableau ...
  if (indiceS >= nEchantillons) {
    // ...retour au début
    indiceS = 0;
  }
  moyenneS = totalS / nEchantillons; // calcul de la moyenne

 //Moyennage de lecture de tension Moteur:
  totalM = totalM - echantillonM[indiceM];      // Soustraction de l'echantillon précédent
  echantillonM[indiceM] = analogRead(uMotRead);// Lecture du capteur
  totalM = totalM + echantillonM[indiceM];      // Ajout du dernier echantillon
```

```
  indiceM++;// Incrémentation de l'indice
  // si on est à la fin du tableau ...
  if (indiceM >= nEchantillons) {
    // ...retour au début
    indiceM = 0;
  }
  moyenneM = totalM / nEchantillons; // calcul de la moyenne

  lcd.home(); lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("U Serv:  U Mot:");
  lcd.setCursor(0, 1);
  lcd.print(moyenneS);//affichage de la tension batterie servitude
  lcd.setCursor(9, 1);
  lcd.print(moyenneM);//affichage de la tension batterie servitude
  delay(150);
}




void task_serv() {



static unsigned long tempoStart1 = 0;



//lecture de la tension de service:
```

```
analogRead(uServRead);


//Condition de mode charge "BULK":


 if (ServState == LOW){
  analogWrite(mosfetServ,255);   //Mosfet piloté en mode "BULK".
  //changement d'etat des LEDs et relay:
  digitalWrite(ledServB,HIGH);
  digitalWrite(ledServV,LOW);
  digitalWrite(RelayServ,HIGH);


  tempoStart1 = millis();
   }
if (moyenneS >= 790){
   ServState = HIGH;
  }
//if (moyenneS<= 620) {ServState = LOW;}



//Condition de mode de charge" FLOAT":
if (ServState == HIGH){
   unsigned long currentTempo1= millis();


   if(currentTempo1 - tempoStart1>= timeToFloat){  //Temporisation du mode Bulk avant de repasser en mode Float:


       //systeme de régulation de la tension pour le mode "Float":
       if(moyenneS < 710) {
```

```
          mosfetServ_drive ++;
          analogWrite(mosfetServ, mosfetServ_drive);
         }
        if(moyenneS > 730) {
          mosfetServ_drive --;
          analogWrite(mosfetServ, mosfetServ_drive);
         }
     digitalWrite(RelayServ,LOW);
     digitalWrite(ledServB, LOW);
     digitalWrite(ledServV, HIGH);
    }
  }
}



void task_mot() {

static unsigned long tempoStart2 = 0;

//lecture de la tension Moteur:
analogRead(uMotRead);

//Départ du cycle en mode Bulk:
  if(MotState == LOW ){
   analogWrite(mosfetMot,255);   //Mosfet piloté en mode "BULK".
   //changement d'etat des LEDs Verte "on" et relay "on"
   digitalWrite(ledMotB,HIGH);
```

```
  digitalWrite(ledMotV,LOW);
  digitalWrite(RelayMot,HIGH);

  tempoStart2 = millis();
}
if (moyenneM >= 840){
  MotState = HIGH;
}
//if (moyenneM<= 620) {MotState = LOW;}



 //Condition de mode de charge" FLOAT":
 if(MotState == HIGH){
   unsigned long currentTempo2= millis();

   if(currentTempo2 - tempoStart2>= timeToFloat){  //Temporisation du mode Bulk avant de repasser en mode Float 8min.

          digitalWrite(ledMotB, LOW);
          digitalWrite(ledMotV, HIGH);
          digitalWrite(RelayMot,LOW);

          //Systeme de régulation de tension Float:
          if(moyenneM < 720) {
            mosfetMot_drive ++;
            analogWrite(mosfetMot, mosfetMot_drive);
            }
```

```
        if(moyenneM > 740) {
          mosfetMot_drive --;
          analogWrite(mosfetMot, mosfetMot_drive);
          }
        }
    }


}
```