/*
CP0097 Rotating Plate Control Copyright Cyrob 2019
Arduino pro Micro
==========
========== OPEN SOURCE LICENCE
==========
==========

.....
Release history
.....
Version Date Author Comment
1.0d1 02/02/19 Phildem First blood
1.0 06/02/19 Phildem Version OK
1.1 06/02/19 Phildem Version OK, relative speed improvement
1.2 07/02/19 Phildem Version OK, Code refactoring and presentation
1.2* 29/06/19 Alain A. French comment
*/

//configurer le driver en 1/4 pas

Description

Contrôler un moteur pas à pas en 2 modes :
- SpeedMode : La vitesse varie en fonction de la position du bouton d'une vitesse -maxi à +max .position zéro en milieu de course

LA LED est éteinte si le moteur est à l'arrêt
LA LED est verte si le moteur est en mouvement
LA LED est Rouge si la vitesse maxi est atteinte
- TrackMode : La position du moteur suit celle du bouton
L'incrémentation de la position change avec la vitesse

De déplacement du bouton pour permettre un meilleur positionnement
LA Led est bleue
Appuyer sur le bouton arrête le moteur et change de mode
Les enroulements du moteur ne sont plus alimentés si le moteur est Off

```
//affectation des  pins.
#define kOutCpuLed  13  // Active High led visu moteur actif
#define kOutRLed    12  // Active Low  led visu vitesse maxi atteinte
#define kOutGLed    11  // Active Low  led visu moteur actif
#define kOutBLed    10  // Active Low  led visu mode track

#define kOutDir     8   // commande direction
#define kOutStep    9   // commande pas
#define kOutEnable  5   // niveau bas active le  moteur
// codeur
#define kInKnobCk   2   // horloge
#define kInKnobDir  3   // direction
#define kInKnobPush 4   // poussoir

// initialisation des consantes
#define kMicroIdle  20 // unité de temps processus rapide en  µs
#define kStepPulseWidth_us  4 // tempo impulsion commande en µS
#define kMotorWakeUp_ms    10 // tempo en ms après un enable
#define kMinSpeedDelay_uT 10000 // délais maxi entre chaque  vitesse mini
#define kMaxSpeedDelay_uT   100 // délais mini entre chaque  vitesse maxi
//pulse in Micro idle steps vitesse max en kInKnonck
#define kMaxSpeed 100// Max Value of gSpeed valeur maxi de la vitesse
#define kSpeedInc  5 // Knob increment for Speed incrémentation vitesse
#define kPosInc 2 // TrackMode incrémentation mode track

// variables Globales
static unsigned long  gLastMicro =0;// mémorisation dernière valeur Micro
static unsigned long  gLastMilli =0;// mémorisation dernière valeur Milli

static long gSpeed=0; // 0 arret, -kMaxSpeed to +kMaxSpeed  vitese et
//direction

static unsigned long  gPulseDelay =0;// compteur entre chaque impulsion
//                                   // du moteur
static bool gMotEnable= false;           // état du moteur (M/A)
static boolgLastKnobCk;                  // dernier état du codeur  Ck
static unsigned long  gLastKnobMilli=0;  // mode track vitesse du moteur

static bool          gModeTrack = false;  // vraie si en mode  mode Track
static unsigned long     gTargetPos  =0; // track position cible
unsigned long          gActualPos    =0;  // Track position actuelle
// configuration des pins en entée ou en sortie
void setup(void) {

  //configuration des pin en entrée pullup
  pinMode(kInKnobCk, INPUT_PULLUP);
  pinMode(kInKnobDir, INPUT_PULLUP);
  pinMode(kInKnobPush, INPUT_PULLUP);

  gLastKnobCk=digitalRead(kInKnobCk); // initialiser la valeur horloge
```

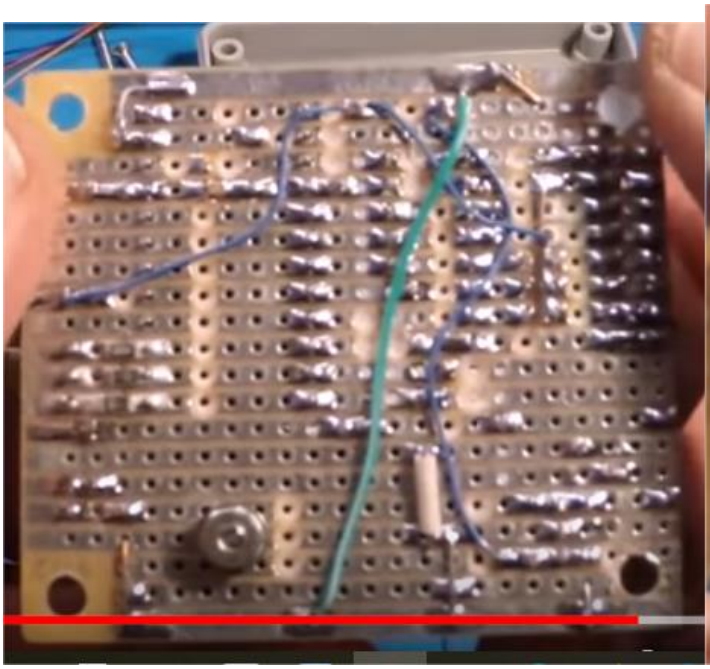
```
// configuration des pin en sortie
  pinMode(kOutCpuLed, OUTPUT);
  pinMode(kOutRLed, OUTPUT);
  pinMode(kOutGLed, OUTPUT);
  pinMode(kOutBLed, OUTPUT);

  pinMode(kOutDir, OUTPUT);
  pinMode(kOutStep, OUTPUT);
  pinMode(kOutEnable, OUTPUT);
// initialisation des valeurs au départ

  digitalWrite(kOutDir, LOW);
  digitalWrite(kOutStep, LOW);
  DisableMotor();

// initialiser le clock

  digitalWrite(kOutCpuLed, HIGH);// LED sur la platine allumée

  // Test Led
  SetLedRed();
  delay(500);
  SetLedGreen();
  delay(500);
  SetLedBlue();
  delay(500);

  for (int i=0;i<5;i++){
    SetLedRed();
    delay(20);
    SetLedOff();
    delay(150);
  }

  digitalWrite(kOutCpuLed, LOW);//led sur la platine éteinte
  SetModeSpeed(); //mode vitesse par défaut
}


// contrôle des LED

void SetLedRed(){ //sous programme allumer uniquement la LED rouge
    digitalWrite(kOutRLed, LOW);
    digitalWrite(kOutGLed, HIGH);
    digitalWrite(kOutBLed, HIGH);
}


void SetLedGreen(){  //sous programme allumer uniquement la LED verte
    digitalWrite(kOutRLed, HIGH);
    digitalWrite(kOutGLed, LOW);
    digitalWrite(kOutBLed, HIGH);
}


void SetLedBlue(){ //sous programme allumer uniquement la LED bleue
digitalWrite(kOutRLed, HIGH);
    digitalWrite(kOutGLed, HIGH);
    digitalWrite(kOutBLed, LOW);
```

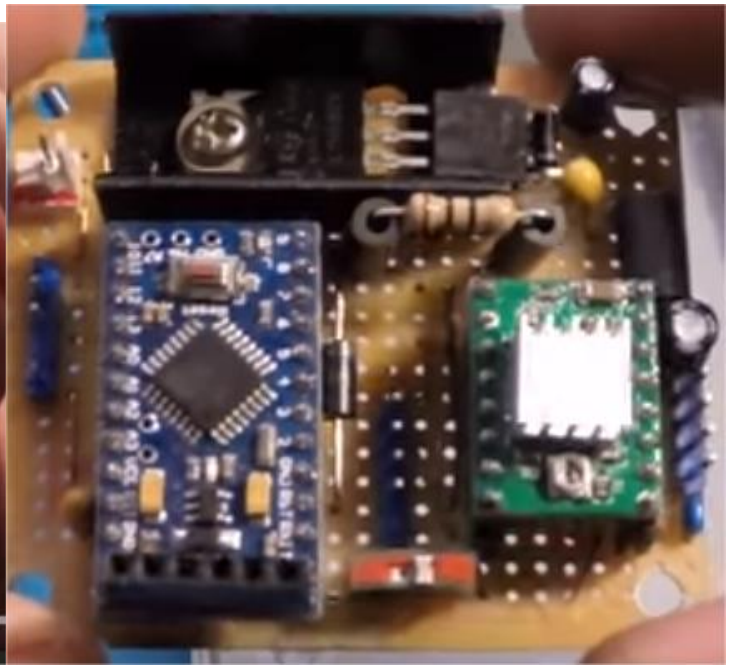
```
}


void SetLedOff(){  // sous programme allumer uniquement //LED bleue
digitalWrite(kOutRLed, HIGH);
    digitalWrite(kOutGLed, HIGH);
    digitalWrite(kOutBLed, HIGH);
}

//command des lED appelé toutes les ms
void LedIdle() {  //sous programme commande  des LED

  if (gModeTrack) //si mode Track activé
    SetLedBlue(); // allumer LED bleue
  else if (gSpeed==0)          //si mode Track non activé et si vitesse=0
    SetLedOff();                    // eteindre les LED
  else if (abs(gSpeed)>=kMaxSpeed) //si vitesse max atteinte
    SetLedRed();                   //alllumer LED rouge
  else
    SetLedGreen();                 // sinon allumer LED verte
}
// traitement du codeur
//regarder si le codeur est en mouvement retourner 0 si il ne l'est pas et
//-step ou + step suivant la direction
//-------------------------------------------------------------------------
long ReadKnob() { //fonction lecture  codeur  (retourne 0 ou -pas ou +pas)

  bool  Ck=digitalRead(kInKnobCk); //lecture horloge du codeur

  if (Ck==gLastKnobCk) // si comme l'état précédent ReadKnob()=0
    return 0;

  delayMicroseconds(400); // antirebond

  gLastKnobCk=Ck;// initialiser l'état actuel  de l'horloge

  if (Ck==LOW)// si état bas glastKnoCk=0
      return 0;

  unsigned long Step; //variable locale
  unsigned long Now=millis();
  if (gModeTrack){                        //
    gLastKnobMilli=Now-gLastKnobMilli; //si mode track gestion de la
//vitesse suivant la proximité de l'objectif
    Step=kPosInc; //pas=2
    if (gLastKnobMilli<350)
      Step*=8;
    else if (gLastKnobMilli<500)
      Step*=4;
    else if (gLastKnobMilli<600)
      Step*=2;
  }
  else
    Step=kSpeedInc;

  gLastKnobMilli=Now;

  if (digitalRead(kInKnobDir)==LOW)   // lecture codeur (pin 3)
```

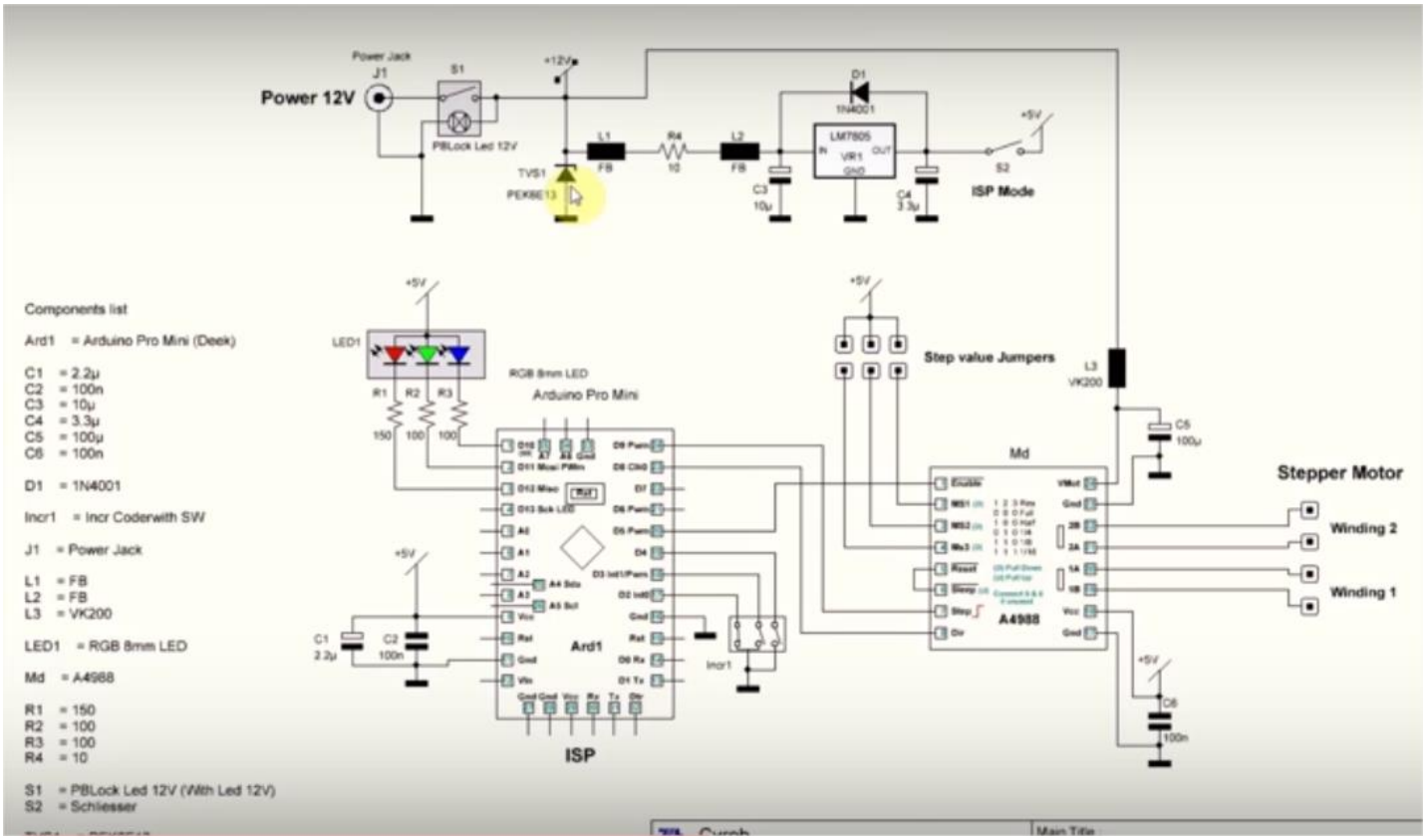
```
     return -Step;

  return Step;    //retourner la valeur du pas
}

void KnobRotIdle() {// traitement du codeur
 long k=ReadKnob();
if (k==0)
    return;

  if (gModeTrack) { // si en mode track
    SetRelativeTrackPosition(k);  // modifie le valeur cible
  } else
    SetSpeed(gSpeed+k); sinon incrementer
}
 //regarder l'etat du bouton toutes les ms
void KnobPushIdle(){ // gestion du bouton appuye

  if (digitalRead(kInKnobPush)==HIGH)
    return;

  // Wait for button released // attene du relachement du bouton
  while (digitalRead(kInKnobPush)==LOW)
    delay(1);

  // Toggle mode basculement de mode
  if (gModeTrack){      // Return to speed mode
    SetModeSpeed();
  }
  else {                // Return to Track mode
    SetModeTrack();
  }
}
// controle du moteur
// envoi d'une impulsion
void StepOne() { // envoi des impulsions moteur
  digitalWrite(kOutStep, HIGH);
  delayMicroseconds(kStepPulseWidth_us); //tempo entre impulsion
  digitalWrite(kOutStep, LOW);
}

// mise sous tension du moteur

void EnableMotor(){ //mise en route du moteur
  digitalWrite(kOutCpuLed, HIGH);//commande LED
  digitalWrite(kOutEnable, LOW); // commande mise sous tension
  if (!gMotEnable)  // si le moteur n'est pas arrete
delay(kMotorWakeUp_ms);// laisser le temps du reveil (10ms)
  gMotEnable=true; //memorisation de l'eta du moteur
}

// Disable motor power arret du moteur

void DisableMotor(){
  digitalWrite(kOutCpuLed, LOW);  // commande LED
  digitalWrite(kOutEnable, HIGH); // arret moteur
  gMotEnable=false; //memorisation de l'eta du moteur}

}
```

```
//-------------------------------------------------------------------------
// Set Global Speed, sign is direction, Speed will be clamp to max
//-------------------------------------------------------------------------
void SetSpeed(long Speed) {//reglage de la vitesse
gSpeed = constrain(Speed, -kMaxSpeed, kMaxSpeed); // encadrement de la
//vitesse (constrain est une fonction arduino)
}


void MotorIdle() {// gestion moteur
  if (gSpeed==0){       //si vitesse =0
    DisableMotor();   //arret moteur
    return;
  }

  if (gPulseDelay>0){ //si le temps entre 2 périodes n'est pas fini
    gPulseDelay--;    // decrementla vitesse
    return;
  }
 EnableMotor();// mise en route

  if (gSpeed>0){// si vitesse positive
    digitalWrite(kOutDir, LOW);  // met la sortie dans un sens
    gActualPos++;  } // incremete la position
  else {
    digitalWrite(kOutDir, HIGH); // met la sortie dans  l'autre  sens
    gActualPos--;                // incremete la position
  }

  StepOne(); // commande  le pas
  gPulseDelay=kMinSpeedDelay_uT/abs(gSpeed);  // recalculer le delais qu
'il faut attendre entre 2 impusions en fontion de la vitese
}

// gestion du mode

void SetModeSpeed(){//mise en mode vitesse
      gModeTrack=false; //intitialise le mode vitesse
      SetSpeed(0);
}
void SetModeTrack(){ // mise en mode track
      gModeTrack=true;// intitialise le mode track
      gActualPos=0; // position actuel=0
      gTargetPos=0; // position cible 0
      SetSpeed(0);  // vitesse =0
}

//  Set the target position in Track Mode (Pos or neg vs Dir)
//modification de la cible
void SetRelativeTrackPosition(long Move){
   gTargetPos+=Move; // en fonction de la position actuel incremeter la
//cible d'ue valeur move
}
//  Handle Mode must be call every ms gestion du mode
void ModeIdle(){

  if (!gModeTrack) // si pas en mode track ne rien faire
    return;
```

```
  if (gActualPos==gTargetPos){ si position cible atteinte

      gActualPos=0;
      gTargetPos=0;
      SetSpeed(0);
  }
  else
    SetSpeed(gTargetPos-gActualPos);   // calcule de la vitesse
                    // vitesse fonction de la distance de la cible
}
//  supervision multitâche
//  Main Loop programme principal

void loop(void) {

  // µs Ticker consultation  mode rapide (gestion moteur et rotation
bouton)
 // appel à intervalle de régulier (rapide) toutes les 20µS
  unsigned long CurMicro=micros();// mémorisation temps actuel en µs
// gestion de registre de temps
  if (CurMicro<gLastMicro){  //si le registre déborde (passage en négatif
      gLastMicro=CurMicro;  }// gestion du débordement registre
// test toutes les 20µS
  if ((CurMicro-gLastMicro)>kMicroIdle){//si temps actuel- temps mémorisé
                                        // > unité de temps rapide (ici
                                        //20µS)
    gLastMicro=CurMicro;                // temps mémorisé = temps présent

    // Fast Engine -------------
    MotorIdle();    // gestion du moteur
    KnobRotIdle(); // gestion codeur
 }

  // consultation  moins rapide (gestion bouton et LED toutes les ms)
  unsigned long CurMilli=millis(); //mémorisation temps pressent en ms
  if (CurMilli !=gLastMilli){ //si on a dépasser 1 ms
    gLastMilli=CurMilli;
  // Slow Engine gestion du changement de mode
    ModeIdle();
// Slow UI gestion du bouton poussoir et des LED
   KnobPushIdle(); //gestion poussoir
   LedIdle();      //gestion LED

  }
}
```






Power 12V
Power Jack
J1
S1
+12V
PBLock Led 12V
TVS1
PEK8E13
L1
FB
R4
10
L2
FB
D1
1N4001
LM7805
IN
VR1
OUT
GND
C3
10µ
C4
3.3µ
+5V
S2
ISP Mode
Components list
Ard1 = Arduino Pro Mini (Deek)
C1 = 2.2µ
C2 = 100n
C3 = 10µ
C4 = 3.3µ
C5 = 100µ
C6 = 100n
D1 = 1N4001
Incr1 = Incr Coderwith SW
J1 = Power Jack
L1 = FB
L2 = FB
L3 = VK200
LED1 = RGB 8mm LED
Md = A4988
R1 = 150
R2 = 100
R3 = 100
R4 = 10
S1 = PBLock Led 12V (With Led 12V)
S2 = Schiesser
LED1
RGB 8mm LED
R1
R2
R3
150
100
100
Arduino Pro Mini
Ard1
ISP
C1
2.2µ
C2
100n
Incr1
Step value Jumpers
L3
VK200
C5
100µ
Md
A4988
Stepper Motor
Winding 2
Winding 1
C6
100n