BOBINADORA DE MICROINDUCTORES

DR. MARTÍN SARAVIA

Grupo de Investigación en Multifísica Aplicada

Noviembre de 2017 – Version 0.0

1

HARDWARE

1.1 INTRODUCCIÓN

El desarrollo consiste en una máquina que permite bobinar automáticamente inductores de dimensiones reducidas. La máquina es automática, una vez que los parámetros de bobinado (longitud, diámetro de alambre, número de vueltas, etc) son seteados en el software, éste se encarga de controlar la ejecución del proceso.

La máquina es controlada por un par de motores paso a paso, los cuales son controlados una placa Arduino Mega acoplada a una placa RAMPS que aloja los drivers de los motores. El software de control está basado en un algoritmo que utiliza un enfoque pseudo-temporal para el conteo de ciclos de bobinado.

2

SOFTWARE

El software que controla la bobinadora fue desarrollado en la plataforma Arduino.

2.1 CÓDIGO

```
/*-------------------------------------------------------------
  License
  This file is part of Arduino Winding Control.
  windingControl is free software: you can redistribute it and/or modify it
  under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  windingControl is distributed in the hope that it will be useful, but WITHOUT
  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
  FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
  for more details.
  You should have received a copy of the GNU General Public License
  along with windingControl. If not, see <http://www.gnu.org/licenses/>.
  Martín Saravia
  Grupo de Investigación en Multifísica Aplicada
  Universidad Tecnológica Nacional - Facultad Regional Bahía Blanca
  CONICET
  2017
  -------------------------------------------------------------*/
  //-------------------------------------------------------------
  // USER CONTROL SECTION - WINDING SCHEME SETUP
  //-------------------------------------------------------------
  unsigned int strokes = 10 ; // Total strokes
  float wps = 1.0 ; // Speed (revolutions per second)
  float coilid = 11.0 ; // Internal coil diameter
  float coilth = 14.0 ; // Coil width
  float wdiam = 0.0635 ; // Wire diameter
  float micro = 4.0 ; // Micro Stepping
  float tfactor = 1.5 ; // % of increment in wire diameter to account for imperfecttions
  float Lbacklash = 0.4 ; // Left Backlash mm
  float Rbacklash = 0.4 ; // Right Backlash mm
  boolean crev = false ; //Reverse motor flag (false=starts to the right, true=starts to
the left)
  //-------------------------------------------------------------
  // Step motors control setup
  byte StartPin = 1 ;
  byte KillPin = 3 ;
```

```
byte WstepPin = 54 ;
byte WdirPin = 55 ;
byte WenaPin = 38 ;
byte CstepPin = 60 ;
byte CdirPin = 61 ;
byte CenaPin = 56 ;
byte WledPin = 13 ;
byte FanPin = 9 ;
byte RedLedPin = 44;
byte GreLedPin = 64;
byte BluLedPin = 59;
// Setup de control de tiempo
unsigned long tnow ;
unsigned long t0w ;
unsigned long dtw ; // Time between winding motor steps(micros)
unsigned long dtc ; // Time between stroke motor steps (micros)
unsigned long dtwr ; // Time between winding motor steps at startup
unsigned long dtcr ; // Time between stroke motor steps ar startup
unsigned long dtw0 ; // Tiempo entre pasos del motor bobinador (micros) en el arranque
unsigned long dtc0 ; // Tiempo entre pasos del motor de carrera (micros) en el arranque
float frac;
float spw;
float spc;
unsigned int backsteps ; // backlash steps
float cerror = 0.0 ;
unsigned int wloops ;
unsigned int wcount = 0 ; // loop count
unsigned int iw0 = 0 ; // loop count
unsigned int wstep = 0 ; // wounding step count
unsigned int ccount = 0 ; // stroke count
unsigned int cstep = 0 ; // stroking step count
boolean goloop = true ;
int temp = 0 ; // Temporary variable
int chota ;
unsigned long microdelay = 50;
unsigned long TotalTime;
unsigned long dtbk; //dt for the backlash velocity
unsigned int acbk; // Cutoff steps for backlash ramp
boolean PauseFlag = false;
//-------------------------------------------------------------
// INIT
//-------------------------------------------------------------
void setup() {
Pre(); // Pre-processing
// Wait for Start Button pressed
while(int Start = digitalRead(StartPin) == HIGH){
```

```
  BlinkLed('R');
  }
  // Let's go...
  TurnOnLed('G');
  digitalWrite(FanPin, HIGH); // Start the Fan
  digitalWrite(WenaPin, LOW); // Enable wounding motor
  digitalWrite(CenaPin, LOW); // Enable stroking motor
  digitalWrite(WdirPin,HIGH); // Set wounding dir
  if (crev == false){
  digitalWrite(CdirPin,HIGH);} // Set stroking dir
  if (crev == true){
  digitalWrite(CdirPin,LOW);} // Set stroking dir
  digitalWrite(WledPin, LOW); // Turn on some LED (which?)
  // Calculate contants
  wdiam = wdiam * tfactor; // Effective wire diamater = wire diameter + imperfection
errors
  spw = (micro * 200) * 1.5; // Steps per loop (1.5 is the reudction factor
  spc = (micro * 200) * 8 * ( wdiam / 1.5 ) ; // Steps per unit stroke (wire diameter)
(8 is the reduction factor, 1.5 is the pitch of the screw)
  dtwr = 1000000 * (1.0 / (wps * spw) ); // Time between wounder steps (microseconds)
  dtcr = 1000000 * (1.0 / (wps * spc) ); // Time between stroker steps (microseconds)
  // dtw0 = 4.0 * dtwr; // Time between wounder steps at start(microseconds)
  // dtc0 = 4.0 * dtcr; // Time between stroker steps at start (microseconds)
  wloops = int(coilth / wdiam );
  // Start looping ....
  t0w = micros() ; // Set the start time
  t0c = t0w ;
  chota = 450;
  for (int ic = 1; ic <= strokes; ic++){
  iw0 = 1; // Restart the initial loop count
  for (int iw = 1; iw <= wloops; iw++){
  while (goloop){
  tnow = micros();
  if (iw == iw0 ){
  frac = ((wstep+chota) / (spw+chota));
  dtw = dtwr / frac;
  dtc = dtcr / frac;
  }
  if (iw == wloops ){
  frac = ((spw-wstep+chota) / (spw+chota));
  dtw = dtwr / frac;
  dtc = dtcr / frac;
  }
  if (iw > iw0 && iw < wloops ){
  dtw = dtwr;
  dtc = dtcr;
  }
```

```
  // Wound step test
  if (tnow - t0w >= dtw){
  PORTF |= B00000001;
  t0w += dtw;
  wstep += 1;
  delayMicroseconds(microdelay);
  PORTF &= B11111110;
  }
  // Stroke step test
  if (tnow - t0c >= dtc){
  PORTF |= B01000000;
  t0c += dtc;
  cstep += 1;
  delayMicroseconds(microdelay);
  PORTF &= B10111111;
  }
  // Test for complete revolution
  if (wstep == spw){
  wcount += 1;
  //cerror = (cstep - wcount * spc); // Error in stroke
  //Serial.println(cerror);
  wstep = 0;
  // Stop the execution if the Start button is down
  if( digitalRead(StartPin) == HIGH ) {
  Stop();
  t0w = micros() ; // Reset the t0 instans
  t0c = t0w ;
  iw0 = iw + 1;
  }
  break;
  }
  }
  }
  // Check for reverse stroke direction
  ccount += 1;
  if (crev == true){
  crev = false;
  digitalWrite(CdirPin, HIGH);
  backsteps = int(micro * 200) * 8 * ( Lbacklash / 1.5 ) ; // Backlash steps for Left
stop
  }
  else{
  crev = true;
  digitalWrite(CdirPin, LOW);
  backsteps = int(micro * 200) * 8 * ( Rbacklash / 1.5 ) ; // Backlash steps for Left
stop
  }
```

```
  // Reverse some steps to eliminate backlash
  acbk = 50; // Number of acceleration steps
  for (int i = 1; i <= backsteps; i++){
  if(i > acbk && i < (backsteps - acbk)){
  BacklashStep(dtcr / 2.0); }
  else{ // Acceleration loop
  if(i <= acbk){
  dtbk = dtcr / ( (i+10.0) / (acbk+10.0) );} // Ojo con la coma del 10, sino da division
por cero
  if(i >= (backsteps - acbk)){
  dtbk = dtcr / ( (backsteps-i+10.0) / (acbk+10.0) );}
  BacklashStep(dtbk / 2.0);
  }
  }
  // Write the end of stroke output data
  delay(100);
  // Print stroke end info
  // Serial.print("---> Stroke ");
  // Serial.print(ccount);
  // Serial.print(" has ended. ");
  // Serial.print("Total wounds is: ");
  // Serial.print(wcount);
  // Serial.print("\n");
  // Reset t0 (very very very very important)
  t0w = micros() ;
  t0c = t0w ;
  }
  // ----------------------------------------
  // END THE EXECUTION ANF INFORM
  //----------------------------------------
  // Turn off the motors
  digitalWrite(WenaPin, HIGH);
  digitalWrite(CenaPin, HIGH);
  // Print data to serial port 4 at 115200 baud
  Serial.println("---> LOOPING FINISHED...");
  Serial.print("---> Total execution time in seconds is: ");
  TotalTime = millis() / 1000;
  Serial.print(TotalTime);
  Serial.print("\n");
  Serial.print("--> Total wounds is: ");
  Serial.print(wcount);
  Serial.print("\n");
  }
  //-------------------------------------------------------
  // LOOP
  //-------------------------------------------------------
  void loop() {
```

```
}
void Pre(){
// Start the communication variables
//Serial.begin(9600);
Serial.println("---------------------------------");
Serial.println("------- GIMAP WOUNDING 0.0 -------");
Serial.println("---------------------------------");
// Set the pins
pinMode(WdirPin, OUTPUT);
pinMode(WstepPin, OUTPUT);
pinMode(WledPin, OUTPUT);
pinMode(WenaPin, OUTPUT);
pinMode(CdirPin, OUTPUT);
pinMode(CstepPin, OUTPUT);
pinMode(CenaPin, OUTPUT);
pinMode(FanPin, OUTPUT);
pinMode(StartPin, INPUT_PULLUP);
pinMode(KillPin, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(KillPin), Kill, LOW);
pinMode(RedLedPin, OUTPUT);
pinMode(BluLedPin, OUTPUT);
pinMode(GreLedPin, OUTPUT);
digitalWrite(WenaPin, HIGH); // Disable Wound Motor
digitalWrite(CenaPin, HIGH); // Disable Stroke Motor
}
void BacklashStep(float dt){
PORTF |= B01000000;
delayMicroseconds(dt);
PORTF &= B10111111;
delayMicroseconds(dt);
}
void Stop(){
digitalWrite(WenaPin, HIGH); // Disable Wound Motor
digitalWrite(CenaPin, HIGH); // Disable Stroke Motor
while(digitalRead(StartPin) == HIGH){
BlinkLed('B');
}
digitalWrite(WenaPin, LOW); // Disable Wound Motor
digitalWrite(CenaPin, LOW); // Disable Stroke Motor
TurnOnLed('G');
}
void Kill(){
digitalWrite(WenaPin, HIGH); // Disable Wound Motor
digitalWrite(CenaPin, HIGH); // Disable Stroke Motor
TurnOnLed('R');
digitalWrite(FanPin, LOW); // Stop the fan
while(1);
```

```
}
void TurnOnLed(char LedColor){
if(LedColor == 'R'){
digitalWrite(RedLedPin, LOW);
digitalWrite(BluLedPin, HIGH);
digitalWrite(GreLedPin, HIGH);
}
if(LedColor == 'G'){
digitalWrite(RedLedPin, HIGH);
digitalWrite(BluLedPin, HIGH);
digitalWrite(GreLedPin, LOW);
}
if(LedColor == 'B'){
digitalWrite(RedLedPin, HIGH);
digitalWrite(BluLedPin, LOW);
digitalWrite(GreLedPin, HIGH);
}
if(LedColor == 'W'){
digitalWrite(RedLedPin, HIGH);
digitalWrite(BluLedPin, HIGH);
digitalWrite(GreLedPin, HIGH);
}
}
unsigned int blinktime = 500;
void BlinkLed(char LedColor){
if(LedColor == 'R'){
TurnOnLed('R');
delay(blinktime);
TurnOnLed('W');
delay(blinktime);
}
if(LedColor == 'B'){
TurnOnLed('B');
delay(blinktime);
TurnOnLed('W');
delay(blinktime);
}
if(LedColor == 'G'){
TurnOnLed('G');
delay(blinktime);
TurnOnLed('W');
delay(blinktime);
}
}
```