Equipe de Robótica da FURG
FBot/FURGBOL
Centro de Ciências Computacionais - C3
Universidade Federal do Rio Grande

Conserto dos Motores Brushless EC 45 - 30 Watt MAXON

Autor Julia
Lohanna
Mariana
Ysabella
Vivian Misaki Aoki

Rio Grande, 8 de janeiro de 2019

1 Objetivo

Recuperar máximo de motores velhos.

2 Desenvolvimento

2.1 Partes do motor

- Base do motor;
- Placa do circuito dos 3 Hall's;
- Proteção de plástico para o circuito da placa;
- Enrolamento de cobre;
- conjunto Bucha de latão;
- Coisa que fecha o conjunto de bucha de latão;
- Imã.

Figura 1: Partes do motor
Fonte: O autor

- Bucha de Latão com Rolamento no fundo;
- Mola com suporte de plástico;
- Outro Rolamento;
- Coisa que fecha a bucha.

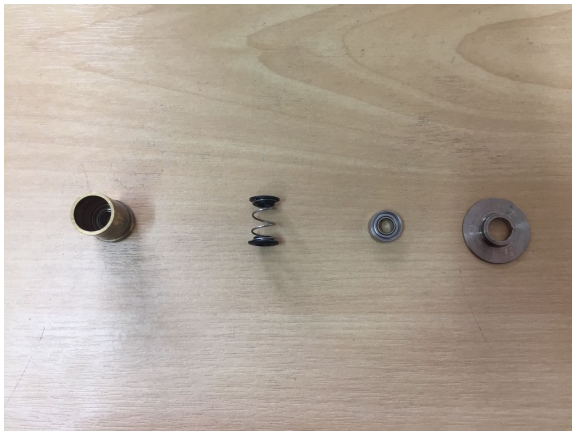

Figura 2: Partes internas do latão (peças 5 e 6 da Figura 1)
Fonte: O autor

3 Desmontar o motor

- Apoiar na fita isolante ou em algo com o mesmo diâmetro (Figura 3)

Figura 3: Uma das partes apoiada para ser desmontada
Fonte: O autor

- Usar um pino, inclinado, quando dá para sentir que está pegando no segundo rolamento, bater com martelo para abrir (Figura 4), as partes internas vão se separar (Figuras 5 e 6)

Figura 4: Primeiro passo para desmontar o motor
Fonte: O autor

Figura 5: Parte interna que será retirada
Fonte: O autor

Figura 6: Parte que se mantém junta
Fonte: O autor

- Com o ferro de solda tirar a solda dos fios das bobinas, ir batendo, com um prego, pra tirar os três pinos do centro e separar as bobinas da placa/flat (Figura3)
- Separar todas as partes

Figura 7: Placa interna
Fonte: O autor

4 Identificação e solução de problemas

- Testar os Hall
  - Pinos utilizados: o 5V, GND e as leituras dos Hall
  - Colocar a fonte em 5V, ligar o GND antes de qualquer coisa e depois os 5V
  - Ler os Hall com o osciloscópio, precisa ter movimento no motor, gera uma onda quadrada

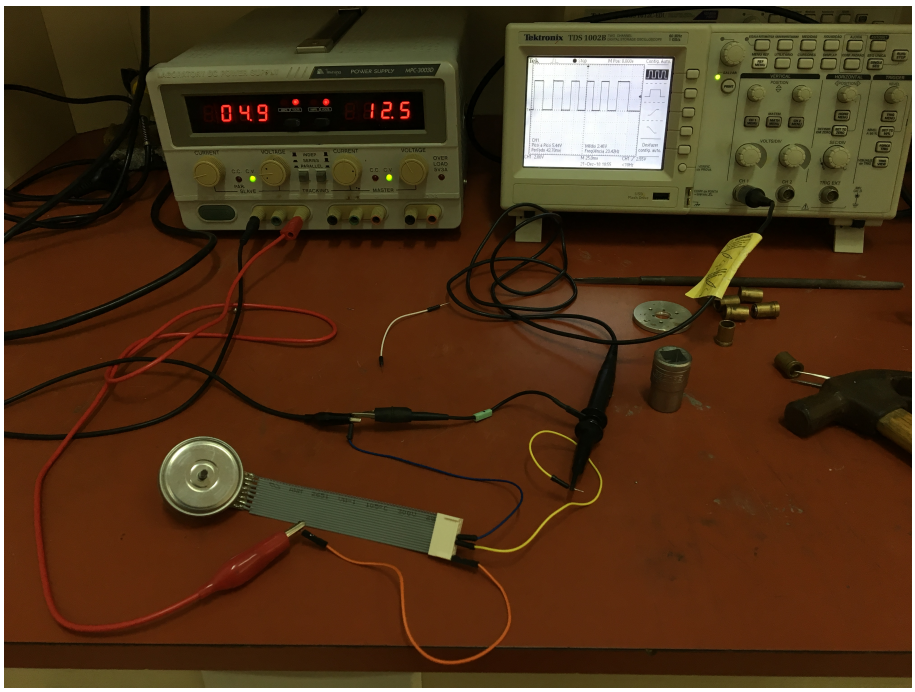

Figura 8: Teste do hall
Fonte: O autor

- Testar se há curto entre os pinos e a borda dos mesmo. Se houver:
  - Borda dos furos centrais pode estar colocando o motor inteiro em curto
  - Observar se os pinos estão encostando na borda, se sim, usar uma broca maior que o furo e raspar a borda (três furos centrais e o latão)
- Verificar se os pinos do plug estão corretos e bem encaixados
- Verificar a resistência entre as bobinas

5 Placa interna do motor

- Aumentar os três furos centrais com uma broca de 2mm
- Prender a placa e o disco externo em uma madeira, com pregos, com a parte metálica para cima

- Utilizando o furo central da parte externa como base, acertar o furo da placa com uma broca de 11mm
- Desmontar para fazer as soldas

6 Montar o motor

- Soldar as bobinas de volta na placa, tomar cuidado para não fechar curto
- inserir a bucha com rolamento $\longrightarrow$ suporte de plástico $\longrightarrow$ mola $\longrightarrow$ suporte de plástico $\longrightarrow$ rolamento $\longrightarrow$ o item que fecha a bucha ao conjunto de bobina.

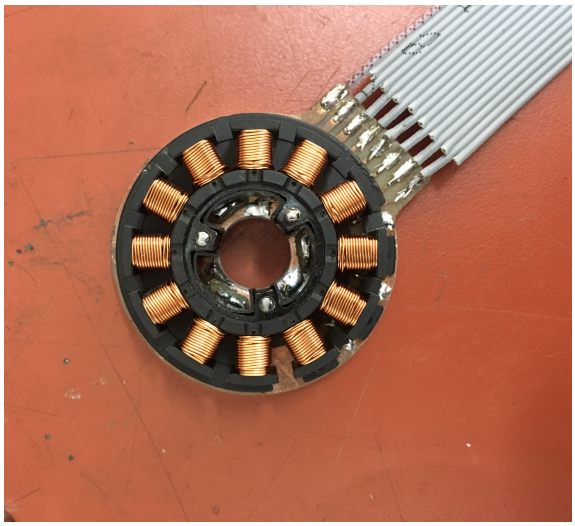

Figura 9: Conjunto da bobina
Fonte: O autor

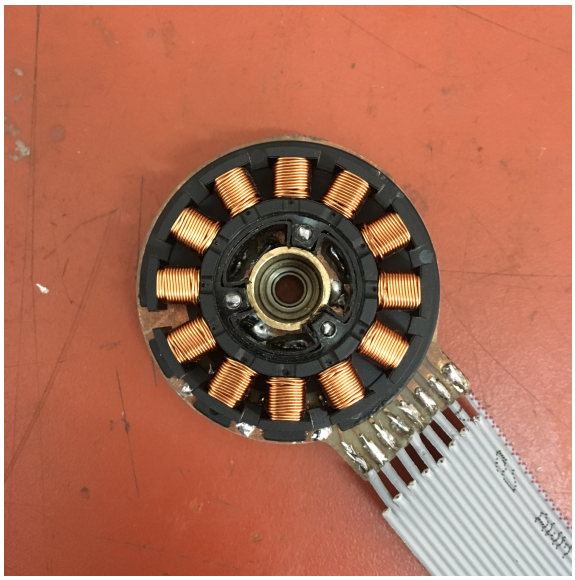

Figura 10: Inserindo bucha no conjunto de bobina
Fonte: O autor

Figura 11: Conjunto de mola
Fonte: O autor

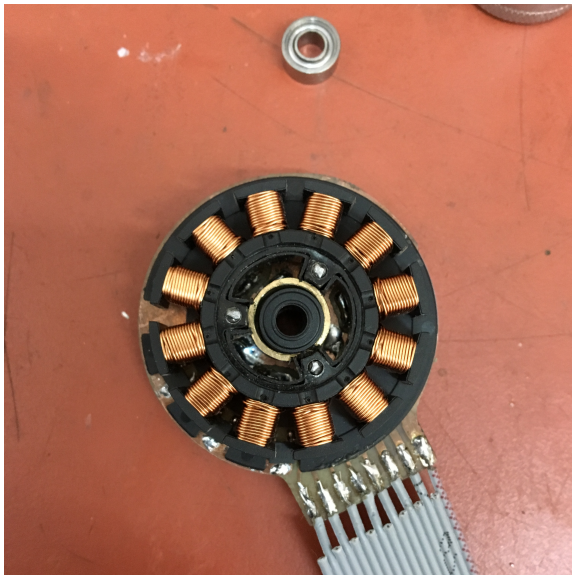

Figura 12: Procedimento1
Fonte: O autor

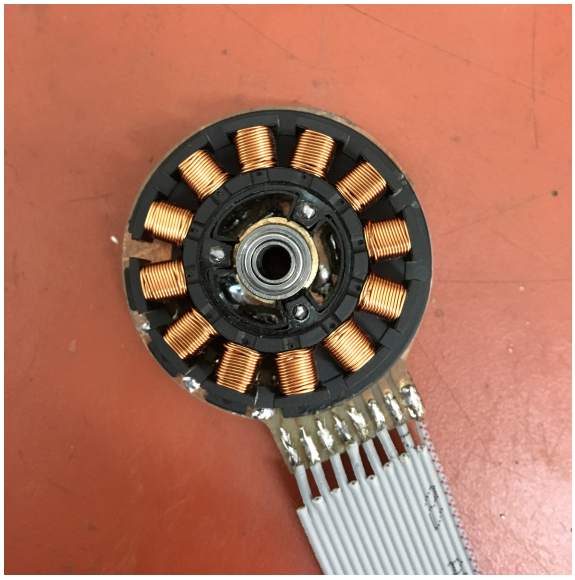

Figura 13: Procedimento2
Fonte: O autor

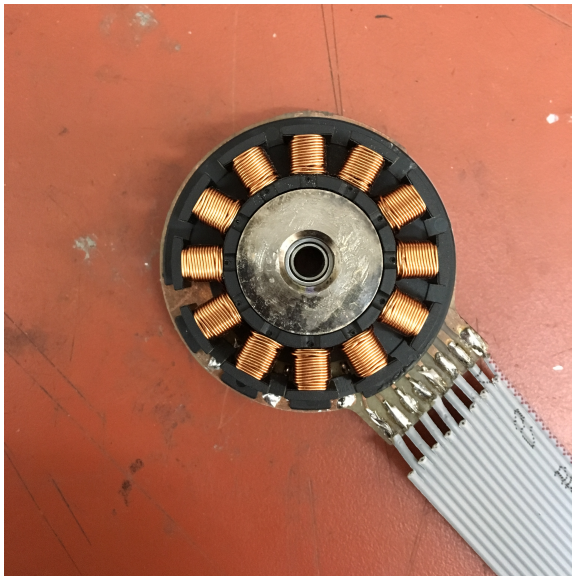

Figura 14: Conjunto da bobina
Fonte: O autor

- Usar uma ferramenta redonda, soquete de chave estrela, (que não encoste nos fios das bobinas) e um martelo para fechar as partes do meio

Figura 15: Fechando o conjunto
Fonte: O autor

- Fechar o motor com cuidado

7 Driver's do motor

- Foi feito teste de acionamento de todos os motores disponíveis nos driver's com o código de teste dos motores encontrado no google drive (A) usando a biblioteca para modificar a frequência do PWM (B).
- Nos driver's sem números não funcionou, e no número 11 necessita de impulso externo a cada parada do motor.
- Testes de continuidade e curto são feitos e faz a troca dos CI's que continuam não funcionando.

Apêndice A Código Teste dos Motores

```
//#include ".\Pwm01\pwm01.h"
#include "pwm01.h"

#define dir_m1 9
#define en_m1 8
//#define pwm_m1 3
#define hall_1 10
#define hall_2 12
#define hall_3 11

uint32_t  pwm_duty = 32767;
uint32_t  pwm_freq1 = 20000;
uint32_t  valPWM = 0;


void setup()
{
  // Set PWM Resolution
  //  pwm_set_resolution(16);

  // Setup PWM Once (Up to two unique frequencies allowed
  //-------------------------------------------------------
  pwm_setup( dir_m1, pwm_freq1, 1);


  // Write PWM Duty Cycle Anytime After PWM Setup
  //-------------------------------------------------------

  pinMode(en_m1, OUTPUT);
  pinMode(hall_1, INPUT);
  pinMode(hall_2, INPUT);
  pinMode(hall_3, INPUT);

  Serial.begin(115200);

    digitalWrite(en_m1, HIGH);
}

void loop()
{

  while (valPWM <= 255) {
    pwm_write_duty( dir_m1, valPWM);
    delay(50);
    valPWM++;

    Serial.println(valPWM);
  }
  delay(2000);

  while (valPWM > 0) {
    valPWM--;
    pwm_write_duty (dir_m1, valPWM);
    delay(50);
```

```
55        Serial.println(valPWM);
56    }
57    delay(2000);
58 }
```

Apêndice B Biblioteca Teste dos Motores

```
/*

Library:      pwm01.h (version 01)
Date:         2/11/2013
Written By:   randomvibe

Purpose:
   Setup one or two unique PWM frequenices directly in sketch program,
   set PWM duty cycle, and stop PWM function.

User Functions:
   pwm_set_resolution(int res) ~ setup PWM resolution; up to 16 bit
   pwm_setup( uint32_t pwm_pin, uint32_t pwm_freq, int iclock) ~ Setup PWM on clock-A (iclock=1) or clock-B (iclock-2)
   pwm_write_duty( uint32_t pwm_pin, uint32_t pwm_duty) ~ Write PWM duty cycle
   pwm_stop( uint32_t pwm_pin) ~ Force PWM stop

Notes:
   - Applies to Arduino-Due board, PWM pins 6, 7, 8 & 9.
   - Libary Does not operate on the TIO pins.
   - Unique frequencies set via PWM Clock-A ("CLKA") and Clock-B ("CLKB")
     Therefore, up to two unique frequencies allowed.
   - Set max duty cycle counts (pwm_max_duty_Ncount) equal to 255
     per Arduino approach.  This value is best SUITED for low frequency
     applications (2hz to 40,000hz) such as PWM motor drivers,
     38khz infrared transmitters, etc.
   - Future library versions will address high frequency applications.
   - Arduino's "wiring_analog.c" function was very helpful in this effort.

*/

#include "Arduino.h"


static int       pwm_resolution_nbit = 8;
static uint32_t  pwm_clockA_freq = 0;
static uint32_t  pwm_clockB_freq = 0;
static uint32_t  pwm_max_duty_Ncount = 255;


void pwm_set_resolution(int res)
{
  pwm_resolution_nbit = res;
}


void pwm_set_clockA_freq(uint32_t  val)
```

```
47 {
48   pwm_clockA_freq = val;
49 }
50
51
52 void pwm_set_clockB_freq(uint32_t  val)
53 {
54   pwm_clockB_freq = val;
55 }
56
57
58 static inline uint32_t mapResolution(uint32_t value, uint32_t from
       , uint32_t to)
59 {
60     if (from == to)
61         return value;
62     if (from > to)
63         return value >> (from-to);
64     else
65         return value << (to-from);
66 }
67
68
69
70 // MAIN PWM INITIALIZATION
71 //-------------------------------------
72 void  pwm_setup( uint32_t  pwm_pin,  uint32_t  pwm_freq,  int
      iclock  )
73 {
74     uint32_t  pwm_duty = 0;
75     uint32_t  chan = g_APinDescription[pwm_pin].ulPWMChannel;
76
77     // SET CLOCK FREQUENCY
78     //-----------------------------
79     if (iclock==1)  pwm_set_clockA_freq( pwm_max_duty_Ncount*
      pwm_freq );
80     if (iclock==2)  pwm_set_clockB_freq( pwm_max_duty_Ncount*
      pwm_freq );
81
82
83     if (pwm_pin>=6 && pwm_pin<=9)
84     {
85         // SET PWM RESOLUTION
86         //-----------------------------
87         pwm_duty = mapResolution( pwm_duty, pwm_resolution_nbit,
      PWM_RESOLUTION);
88
89         // PWM STARTUP AND SETUP CLOCK
90         //-------------------------------------
91         pmc_enable_periph_clk( PWM_INTERFACE_ID );
92         PWMC_ConfigureClocks( pwm_clockA_freq, pwm_clockB_freq,
      VARIANT_MCK );
93
94         // SETUP PWM FOR pwm_pin
95         //-----------------------------
```

```
96          PIO_Configure( g_APinDescription[pwm_pin].pPort,
        g_APinDescription[pwm_pin].ulPinType,  g_APinDescription[
        pwm_pin].ulPin,  g_APinDescription[pwm_pin].ulPinConfiguration
        );
97          if (iclock==1)  PWMC_ConfigureChannel(PWM_INTERFACE, chan,
         PWM_CMR_CPRE_CLKA, 0, 0);
98          if (iclock==2)  PWMC_ConfigureChannel(PWM_INTERFACE, chan,
         PWM_CMR_CPRE_CLKB, 0, 0);
99          PWMC_SetPeriod(PWM_INTERFACE, chan, pwm_max_duty_Ncount);
100         PWMC_SetDutyCycle(PWM_INTERFACE, chan, pwm_duty);
101         PWMC_EnableChannel(PWM_INTERFACE, chan);
102     }
103
104 }
105
106
107
108 // WRITE DUTY CYCLE
109 //-----------------------------------
110 void  pwm_write_duty( uint32_t  pwm_pin,  uint32_t  pwm_duty )
111 {
112     if (pwm_pin>=6 && pwm_pin<=9)
113     {
114         pwm_duty = mapResolution( pwm_duty, pwm_resolution_nbit,
        PWM_RESOLUTION);
115
116         uint32_t  chan = g_APinDescription[pwm_pin].ulPWMChannel;
117
118         PWMC_SetDutyCycle(PWM_INTERFACE, chan, pwm_duty);
119     }
120 };
121
122
123
124 // FORCE PWM STOP
125 //-----------------------------------
126 void  pwm_stop( uint32_t  pwm_pin )
127 {
128     pinMode(pwm_pin, OUTPUT);        // sets the digital pin as
        output
129     digitalWrite(pwm_pin, LOW);      // sets the LED off
130 };
```