Impressora para Placas de Circuitos Impressos

Ponto de controle 3

Antonio Prado da Silva Júnior
Faculdade do Gama
Universidade de Brasília
Gama, Distrito Federal
Contato.pradojr@gmail.com

Ítalo Barbosa Santos
Faculdade do Gama
Universidade de Brasília
Gama, Distrito Federal
italo.b.s.35@gmail.com

A estrutura da impressora de circuitos impressos encontra-se montada. Os eixos X e Y são dispostos em um pedaço de madeira. Atualmente o movimento do eixo X corresponde ao esperado, movimenta-se em uma velocidade adequada ao projeto, porém o eixo Y apresenta algum erro ainda não identificado, pois não se movimenta em conjunto com o eixo X. Os eixos foram testados separadamente a fim de excluir possibilidades de erro, aparentemente o não movimento do eixo Y não se dá ao fornecimento de energia, para comprovar isso o eixo Y foi ligado sozinho e mesmo assim não se movimentou. Outro possível erro descartado foi o de falha no motor ou suas ligações, para verificar tal possibilidade as entradas no código do eixo Y foram substituídas pelo eixo X e o eixo se movimentou, descartando assim a possibilidade de erro físico. O possível erro para o não movimento do eixo é erro na parte da programação, ainda não foi encontrado no código a parte que faz o eixo não se movimentar.

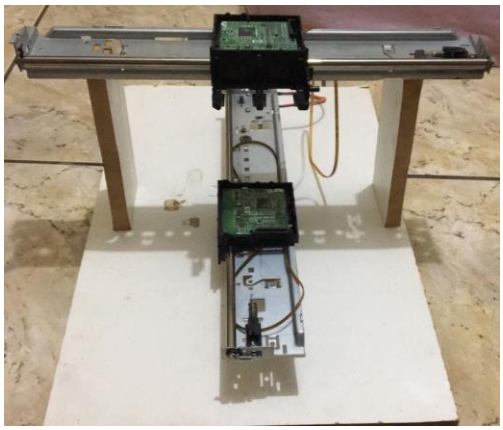

Figura 1 – Estrutura Impressora de Circuitos impressos

```
char Fim()
{

  __asm__("clr.b R12 \n"
          "cmp.w &atual_x, &PosGcode_x \n"
          "jne  fim \n"
          "cmp.w &atual_y, &PosGcode_y \n"
          "jne  fim \n"
          "cmp.w &atual_z, &PosGcode_z \n"
          "jne  fim \n"
          "mov.b #0x1, R12 \n"
          "fim : ret");

}
```

Figura2: sub-rotina em assembly.

O trecho presente na figura 2 é referente a sub-rotina em assembly do sistema que sinaliza o fim dos movimentos dos eixos ao chegar na posição indicado pelo Gcode; Para isso a rotina verifica os valores atuais dos eixos com os valores apresentados no Gcode caso todas as posições sejam iguais ele para os movimentos dos motores até uma nova posição ser lida caso contrario essa sub-rotina retorna a posição a qual ela foi chamada, nesse caso nos movimentos dos motores.

Na figura 3 é presentado o loop que faz o movimento dos motores até a posição indicada no Gcode. O loop sempre testa se o sistema chegou na posição final ao testa a sub-rotina “Fim()”. O sentido é determinado ao comparar a posição atual do eixo com o valor apresentado no Gcode e depois a posição atual era atualizada sendo incrementada ou decrementada. A função EixoY , figura 4, funciona como uma máquina de estados que enviar os valores de bits para o driver do motor de passos de maneira sequencial, a presença do atraso de 3 milissegundos é necessária pra o movimentos ocorrer antes que uma nova sequência ser iniciada.

A interrupção presente no código atual, figura 5, consiste em um reset para emergências onde ele chama uma função que move os eixos para posição 0, essa posição é determinada pelo usuário na estrutura. Outra interrupção será adicionada, ele servirá para a comunicação serial o planejado é que ocorra uma interrupção para receber uma nova posição do Gcode pela comunicação serial.

```
while(!Fim()) //Loop até chegar na posição do GCode
{

    if(PosGcode_z != atual_z)
    {
        if(PosGcode_z > atual_z)
        {
            EixoZ(1,1);
            atual_z++;
        }
        else
        {

            EixoZ(0,1);
            atual_z--;
        }

    }
    else if (PosGcode_z == atual_z)
    {
        if(PosGcode_y > atual_y)
        {
            EixoY(1,1);
            atual_y++;
        }
        else if (PosGcode_y < atual_y)
        {
            EixoY(0,1);
            atual_y--;
        }
```

Figura3: função responsável pelos movimentos dos motores de acordo com o Gcode.

```
void EixoY(volatile int Sentido1, volatile int Enable1)
{

    const int horario[4] = {BIT0, BIT1, BIT2, BIT3};
    const int anti_horario[4] = {BIT3, BIT2, BIT1, BIT0};
    int passos = 0;



       if(Enable1 == 1){
           for (passos = 0; passos < 4; passos++)
           {
               if (Sentido1 == 1){
                   P2OUT = horario[passos];
               }

               else if(Sentido1 == 0){
                   P2OUT = anti_horario[passos];
               }
               Atraso(3);
           }
       }
       else if (Enable1 ==0){
           P2OUT &= ~(BIT0+BIT1+BIT2+BIT3);
       }
    }
```

Figura4: função responsável pelos movimentos dos motores de acordo com o Gcode.

```
#pragma vector = PORT2_VECTOR
__interrupt void PORT2_ISR(void)

{
  if((P2IN&BIT4)==1)
    Inicio();
}
```

Figura5: função responsável pelos movimentos dos motores de acordo com o Gcode.