# Ping Pong com Leds

Rafael Mendes Félix
Universidade de Brasília – Campus Gama
UnB
Gama, Brasil
rafaelmendesf@hotmail.com

***Resumo* — Este trabalho visa apresentar um projeto de um jogo do tipo ping pong que é microcontrolado por um MSP430.**

***Keywords — ping pong, msp***

## I. Introdução

Pong é o primeiro videogame lucrativo da história, dando origem a um novo setor da indústria. Não possuía gráficos espetaculares ou jogabilidade excelente, mas foi de importância fundamental na história do videogame. Foi criado por Nolan Bushnell e Ted Dabney na forma de um console ligado a um monitor, movido a moedas. *Pong* é um jogo eletrônico de esporte em duas dimensões que simula um tênis de mesa. O jogador controla uma paleta (barra vertical) no jogo movendo-a verticalmente no lado esquerdo da tela, e compete contra o computador ou outro jogador que controlam uma segunda raquete no lado oposto. Os jogadores usam suas paletas para acertar a esfera (bola) e mandá-la para o outro lado.
Deseja-se recriar este jogo, mas de uma forma mais lúdica, com leds tomando o lugar do monitor de fliperamas.

## II. Justificativa

Foi escolhido recriar um clássico dos vídeos games, mas implementando de uma forma diferente. Em vez de usar monitores ou displays, foi decido implementar em uma matriz de leds 8x8 e acelerômetros como joystick.

## III. Objetivos

Visamos implementar um sistema de mini game do tipo pong, em uma matriz de leds 8x8 e acelerômetros como controle.

## IV. Hardware

O primeiro componente essencial para o projeto é uma matriz de leds. Ela servirá como se fosse a tela de um vídeo game, então é através dos leds que será visualizado as barras que se movimentam e a bola do jogo. Então foi confeccionado uma matriz própria.

As matrizes encontradas no mercado possuem dimensões pequenas, o que pode atrapalhar na jogabilidade do jogo, o deixando menos atrativo. Por isto, optou-se por fazer uma própria. Bastou comprar 64 leds e uma placa perfurada, e fazendo as ligações corretas, com o auxílio de solda, foi possível montar a matriz do projeto.

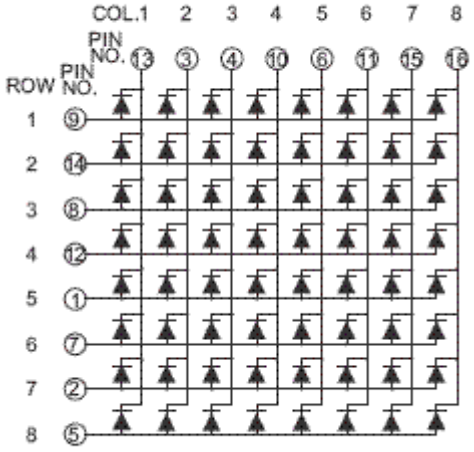


Figura1: Esquema de ligação dos leds

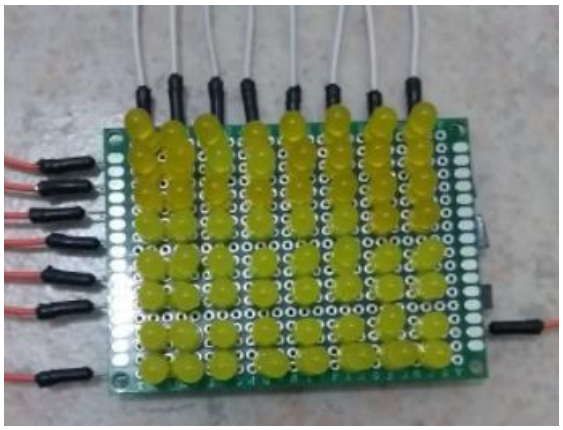

Figura2: Matriz de leds montada

Para controlar a matriz de leds, foi pensado em utilizar um shift register. Foi encontrado o CI 74595 que atende os requisitos necessários.

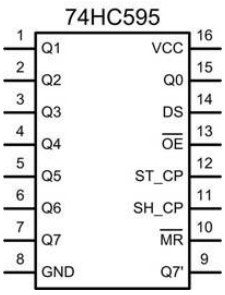

Figura3: CI 74595

Seu funcionamento é da seguinte maneira: ele recebe um sinal em série e apôs receber todo o sinal ele o envia de forma paralela. Suponha que temos um byte, equivalente a 8bits, por apenas um pino ele recebe esses 8 bits e após receber esses 8 bits ele os envia de forma paralela, os pinos responsáveis por enviar os bits de forma paralela são os pinos Q0 ao Q7, o pino que recebe o sinal em série é o pino 14, o DS.

Então, caso queiramos 8 leds, em vez de ligar cada led em um pino do MSP, o que faremos é usar esse CI. Ligaremos o DS no msp e manda os 8 sinais dos leds por ele de forma serial, e com cada led em um respectivo pino Qx (onde x varia de 0 a 7) do CI, quando acabar de mandar os 8 sinais dos LEDs pelo DS, o CI manda pela saída Q os 8 sinais diretos de forma paralela.

Podemos compreender o funcionamento do CI, bem como os seus pinos, a partir da seguinte tabela:

| | | |
|---|---|---|
| Latch | LOW | Prepara para a leitura dos bits na serial |
| Data | HIGH | Deixa data em nível alto (primeiro bit) |
| Clock | HIGH | Armazena o primeiro bit |
| Clock | LOW | Prepara para o armazenamento do segundo bit |
| Data | HIGH | Valor do segundo bit |
| Clock | HIGH | Armazena segundo bit |
| ... | .. | ... |
| Data | LOW | Valor do oitavo bit |
| Clock | HIGH | Armazena o oitavo bit |
| Clock | LOW | Prepara clock em LOW |
| Latch | HIGH | Envia os oitos bits na saída Q |

Considerando que desejamos controlar uma matriz e que o shift register funciona bem controlando uma linha, utilizaremos dois shift register, um controlando as linhas e outro as colunas da matriz.

Foi adquirido apenas um acelerômetro, então para controlar uma das barras foi adicionado dois botões, um para movimentar a barra para a esquerda e outro para a direita.

Também foi adicionado mais dois botões. Um de pause/start e outro para aumentar a velocidade do jogo, sendo que o jogo possuiria 3 velocidades.

Para controlar a última barra, a ideia era utilizarmos um acelerômetro, assim como o proposto inicialmente, porém não conseguimos implementar isto durante o prazo estabelecido do projeto. Foi levantado duas hipóteses a respeito do problema. A primeira era que a alimentação do acelerômetro influenciava na alimentação das trilhas da matriz de leds, uma vez que quando conectado ao sistema, parte da matriz piscava de forma indesejada. A segunda hipótese vem de erro de programação, que apesar do código rodar no code composer, erros de lógica poderiam estar ocorrendo e passado despercebido e não foi possível encontrar onde estava o erro. Retirando o acelerômetro, o sistema funcionava de acordo com o previsto, sendo necessário apenas posicionar a matriz de led de forma a não ficar com mal contato. Para ajustar a corrente dos leds e consequentemente a corrente que chegava, foi adicionado um potenciômetro que ajudava a conter erros e manualmente podíamos ajustar conforme o necessário.

A ideia inicial era que de acordo com a movimentação do usuário, a barra se mova para a esquerda ou direita e com maior ou menor velocidade, dependendo da movimentação.

Por ser um módulo bastante comum e utilizado em projetos com micro controladores, foram encontradas referências de conexões para arduino que auxiliaram no desenvolvimento do projeto.

| *Pino módulo* | *Função* | *Ligação ao Arduino* |
|---|---|---|
| X | voltagem de saída eixo X | A0 |
| Y | voltagem de saída eixo Y | A1 |
| Z | voltagem de saída eixo Z | A2 |
| 3V3 | Alimentação 3.3 Volts | Ligar ao pino 3.3 ou 5 volts do Arduino, dependendo da configuração desejada. Ligar ao 3.3 volts aumenta a precisão da leitura. |
| 5 V | Alimentação 5 Volts | |
| ST | Self Test | Pino 12 |
| GS | Seleção do modo de sensibilidade | Pino 10 |
| 0G | Detecção de queda livre linear | Pino 11 |
| SL | Habilita o modo sleep | Pino 13 |
| GND | Ground | GND |

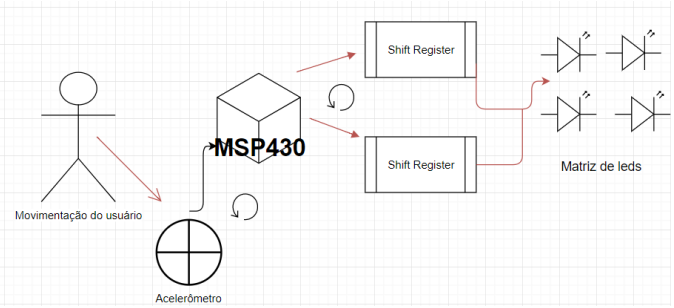

Figura5: esquemático básico do projeto

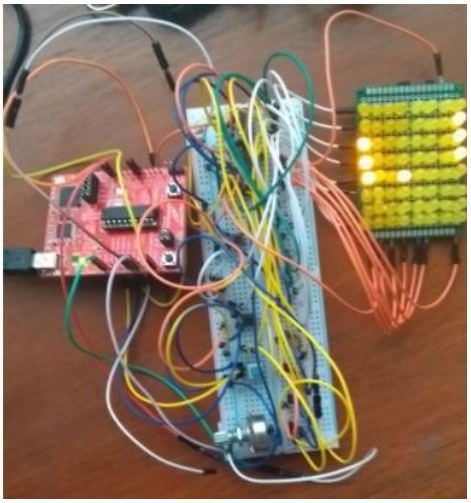
Figura6: projeto montado em funcionamento

# v. software

A lógica de programação se baseava em algumas funções, dentre elas, a função, início, início tela, set display, posição bola, condição jogo 1 e 2.
A função início servia para indicar ao micro controlador quais pinos estavam sendo utilizados e quais as condições iniciais dele, bem como desligar watch dog e botões de interrupção.

```
void inicio()
{

   WDTCTL = WDTPW | WDTHOLD;

   P1SEL2 = BIT5 + BIT6 + BIT7;

  P1OUT = 0;

   P1SEL = BIT5 + BIT6 + BIT7;
   P1DIR = BIT4;
   P1IES = 0;
   P1IFG = 0;
   P2OUT = 0;
   P2DIR = 0;
   P2IES = 0;
   P2IFG = 0;


   BCSCTL2 = SELM_0 + DIVM_0 + DIVS_0;

   if (CALBC1_16MHZ != 0xFF) {

      __delay_cycles(100000);

      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_16MHZ;
      DCOCTL = CALDCO_16MHZ;
   }

   BCSCTL1 |= XT2OFF + DIVA_0;

   BCSCTL3 = XT2S_0 + LFXT1S_2 + XCAP_1;

   UCB0CTL1 |= UCSWRST;
   UCB0CTL0 = UCCKPH + UCMST + UCMODE_0 +
UCSYNC;
   UCB0CTL1 = UCSSEL_2 + UCSWRST;
   UCB0CTL1 &= ~UCSWRST;

  __bis_SR_register(GIE);

  WDTCTL = WDTPW + WDTHOLD;



   P1REN |= BIT0 + BIT1 + BIT2 + BIT3;
   P1OUT = BIT0 + BIT1 + BIT2 + BIT3;
   P1IE |= BIT0 + BIT1 + BIT2 + BIT3;
   P1IES |= BIT0 + BIT1 + BIT2 + BIT3;
   P1IFG &= ~(BIT0 + BIT1 + BIT2 + BIT3);

   _EINT();

}
```

A função início tela imprimia na matriz de led a palavra ping pong assim que se iniciava o jogo ou após um game over.

```
void inico_tela()
{
   int arr[8][8]=
      {{0x00,0x3c,0x24,0x24,0x3c,0x20,0x20,0x00},//p
      {0x00,0x7c,0x10,0x10,0x10,0x10,0x7c,0x00},//i
      {0x00,0x44,0x64,0x54,0x4c,0x44,0x44,0x00},//n
      {0x00,0x3c,0x20,0x20,0x2c,0x24,0x3c,0x00},//g
      {0x00,0x3c,0x24,0x24,0x24,0x24,0x3c,0x00}};//o

   int seq[8]={0,1,2,3,0,4,2,3}; //PINGPONG

   int c;

   for(a=0;a<8;a++)
   {
      for(c=0;c<100;c++)
      {
         for(b=0;b<8;b++)
         {
               send(1<<b,arr[seq[a]][b]); //Envia PING PONG
na tela
               delay(1);
         }
      }
   }

   paus=1;  //Pausa
}
```

A função set display indicava a matriz de leds os bits correspondentes as barras de jogo e a bola, antes de se iniciar o jogo.

```
void set_display()
{
   do
   {
      for(b=0; b<level; b++)
      {
         sendij(i,j);          //display bola

         for(a=0;a<3;a++)
            sendij(p1[a],7);    //display player 1

         for(a=0;a<3;a++)
            sendij(p2[a],0);    //display player 2
```

```
      }
   }
   while(paus);
}
```

A função posição bola calculava para onde a bola deveria ir. Na verdade estávamos controlando os leds de linha 2 a 7, pois os de 1 e 8 eram as barras de controle. Assim, combinado com a função condição de jogo 1 e 2 que também controlavam como as barras deveriam se mover, faz com que o jogo de fato funcione, comparando a bola quando encosta na barra se ela deve continuar em jogo ou se deve computar um ponto para um jogador. Dependendo de onde a bola encosta na barra, ela deve retornar a mesma angulação em sentido oposto, ou a angulação invertida.

```
void posicao_bola()
{
   if(ver==0)  //Esquerda
      j++;
   else       //Direita
      j--;
   if(hor==0)  //Baixo
      i++;
   else       //Cima
      i--;


   if(i==0)
      hor=0;
   else if(i>=7)
      hor=1;
   if(j==1)
      ver=0;
   else if(j>=6)
      ver=1;
}

void condicao_jogo1()
{
   for(a=0;a<3;a++)
   {
      if(p1[a]==i)
         break;
   }
   if(a==1)
   {
      if(i>0 && hor==0) hor=1;
      else if(i<7 && hor==1) hor=0;
   }
   if(hor==0 && p1[0]==(i+1))
   {
      a=0;
      if(i>0) hor=1;
   }
   if(hor==1 && p1[2]==(i-1))
   {
      a=0;
      if(i<7) hor=0;
   }

   if(a==3)
   {
      game_over(1);
      scorea--;
   }
}
```

# VI. Conclusões

Apesar da lógica do jogo ter funcionado bem, ocorreram alguns problemas que fizeram com que o projeto não funcionasse como o proposto inicialmente.

A primeira dificuldade foi a confecção da matriz que por ter sido feita em uma placa muito estreita, dependendo da corrente que se passava nas trilhas, uma nova corrente indesejada era induzida, perdendo muito tempo para ser consertado.

A segunda dificuldade estava na lógica da bola que inicialmente, ao quicar nas barras, retornava de forma “reta”, não dando continuidade ao jogo, bem como também não quicava nas bordas da matriz, esse detalhe foi resolvido na programação do jogo.

O detalhe mais importante do jogo que era o controle por meio do acelerômetro, não foi implementado. A falha da implementação não foi encontrada, não se sabe se foi problemas com o hardware ou na lógica de controle. Deveria ter tido uma organização melhor de tempo para que fosse possível superar problemas do projeto, ainda mais num dos requisitos mais importantes.

# VII. Bibliografias


http://www.makeuseof.com/tag/how-to-recreate-the-classic-pong-game-using-arduino/

https://www.instructables.com/id/8-x-8-LED-Pong-with-Arduino/

https://circuits.io/circuits/2452966-fp-arduino-pong-8x8-led-matrix

http://mcuhq.com/28/msp430-3-axis-accelerometer-and-gyroscope-example-driver-using-the-lsm6ds0