10_Dig_Out

1. **Projete o hardware necessário para o MSP430 controlar um motor DC de 12V e 4A. Utilize transistores bipolares de junção (TBJ) com Vbe = 0,7 V, beta = 100 e Vce(saturação) = 0,2 V. Além disso, considere que Vcc = 3 V para o MSP430, e que este não pode fornecer mais do que 10 mA por porta digital.**

Rb = (Vcc - Vbe)/Ib = (Vcc - Vbe)*B/Ic = 2,3*100/4 = 57,5 Ohms

2. **Projete o hardware necessário para o MSP430 controlar um motor DC de 10V e 1A. Utilize transistores bipolares de junção (TBJ) com Vbe = 0,7 V e beta = 120. Além disso, considere que Vcc = 3,5 V para o MSP430, e que este não pode fornecer mais do que 10 mA por porta digital.**

Rb = (Vcc - Vbe)/Ib = (Vcc - Vbe)*B/Ic = 336 Ohms.

3. **Projete o hardware utilizado para controlar 6 LEDs utilizando charlieplexing. Apresente os pinos utilizados no MSP430 e os LEDs, nomeados L1-L6.**

```
#include <msp430g2553.h>
#define LED1 BIT0
#define LED2 BIT6
#define LED3  BIT3
#define TIME 50

void atraso();

int main(void)
{
	WDTCTL = WDTPW | WDTHOLD;
	while(1)
	{
		P1REN=0;
		P1OUT |= LED1;
		P1OUT &= ~LED2;
		P1DIR |= LED1|LED2;
```

```
            P1DIR &= ~LED3;
            P1REN |= LED3;
            P1OUT |= LED3;
//            atraso();

            P1REN=0;
            P1OUT |= LED2;
            P1OUT &= ~LED1;
            P1DIR |= LED1|LED2;
            P1DIR &= ~LED3;
            P1REN |= LED3;
            P1OUT |= LED3;
//            atraso();

            P1REN=0;
            P1OUT |= LED3;
            P1OUT &= ~LED2;
            P1DIR |= LED3|LED2;
            P1DIR &= ~LED1;
            P1REN |= LED1;
            P1OUT |= LED1;
//            atraso();

            P1REN=0;
            P1OUT |= LED2;
            P1OUT &= ~LED3;
            P1DIR |= LED3|LED2;
            P1DIR &= ~LED1;
            P1REN |= LED1;
            P1OUT |= LED1;
//            atraso();

            P1REN=0;
            P1OUT |= LED1;
            P1OUT &= ~LED3;
            P1DIR |= LED1|LED3;
            P1DIR &= ~LED2;
            P1REN |= LED2;
            P1OUT |= LED2;
//            atraso();

            P1REN=0;
            P1OUT |= LED3;
            P1OUT &= ~LED1;
            P1DIR |= LED1|LED3;
```

```
            P1DIR &= ~LED2;
            P1REN |= LED2;
            P1OUT |= LED2;
//            atraso();



      }
        return 0;
}

void atraso()
{
     volatile int i=0;
     while(i<=TIME)
      i++;
}
```

4. **Defina a função `void main(void){}` para controlar 6 LEDs de uma árvore de natal usando o hardware da questão anterior. Acenda os LEDs de forma que um ser humano veja todos acesos ao mesmo tempo.**

   → Reduzir ou retirar o tempo de intervalo na função atraso

5. **Defina a função `void main(void){}` para controlar 6 LEDs de uma árvore de natal usando o hardware da questão 3. Acenda os LEDs de forma que um ser humano veja os LEDs L1 e L2 acesos juntos por um tempo, depois os LEDs L3 e L4 juntos, e depois os LEDs L5 e L6 juntos.**

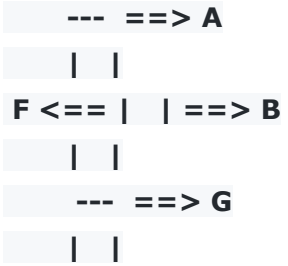
   → Reduzir ou retirar o tempo de intervalo na função atraso entre L1 e L2, entre L3 e L4 e entre L5 e L6 através de parâmetros.

```
#include <msp430g2553.h>
#define LED1 BIT0
#define LED2 BIT6
#define LED3  BIT3
#define TIME 50
```

```
void atraso();

int main(void)
{

	WDTCTL = WDTPW | WDTHOLD;
	while(1)
	{
		P1REN=0;
		P1OUT |= LED1;
		P1OUT &= ~LED2;
		P1DIR |= LED1|LED2;
		P1DIR &= ~LED3;
		P1REN |= LED3;
		P1OUT |= LED3;
//		  atraso();

		P1REN=0;
		P1OUT |= LED2;
		P1OUT &= ~LED1;
		P1DIR |= LED1|LED2;
		P1DIR &= ~LED3;
		P1REN |= LED3;
		P1OUT |= LED3;
		atraso(1000);

		P1REN=0;
		P1OUT |= LED3;
		P1OUT &= ~LED2;
		P1DIR |= LED3|LED2;
		P1DIR &= ~LED1;
		P1REN |= LED1;
		P1OUT |= LED1;
//		  atraso();

		P1REN=0;
		P1OUT |= LED2;
		P1OUT &= ~LED3;
		P1DIR |= LED3|LED2;
		P1DIR &= ~LED1;
		P1REN |= LED1;
		P1OUT |= LED1;
		atraso(1000);
```

```
            P1REN=0;
            P1OUT |= LED1;
            P1OUT &= ~LED3;
            P1DIR |= LED1|LED3;
            P1DIR &= ~LED2;
            P1REN |= LED2;
            P1OUT |= LED2;
//            atraso();

            P1REN=0;
            P1OUT |= LED3;
            P1OUT &= ~LED1;
            P1DIR |= LED1|LED3;
            P1DIR &= ~LED2;
            P1REN |= LED2;
            P1OUT |= LED2;
            atraso(1000);



      }
        return 0;
}

void atraso(volatile int i)
{
    while(i<=TIME)
      i++;
}
```

6. **Defina a função `void EscreveDigito(volatile char dig);` que escreve um dos dígitos 0x0-0xF em um único display de 7 segmentos via porta P1, baseado na figura abaixo. Considere que em outra parte do código os pinos P1.0-P1.6 já foram configurados para corresponderem aos LEDs A-G, e que estes LEDs possuem resistores externos para limitar a corrente.**



7. **Multiplexe 2 displays de 7 segmentos para apresentar a seguinte sequência em loop: 00 - 11 - 22 - 33 - 44 - 55 - 66 - 77 - 88 - 99 - AA - BB - CC - DD - EE - FF**