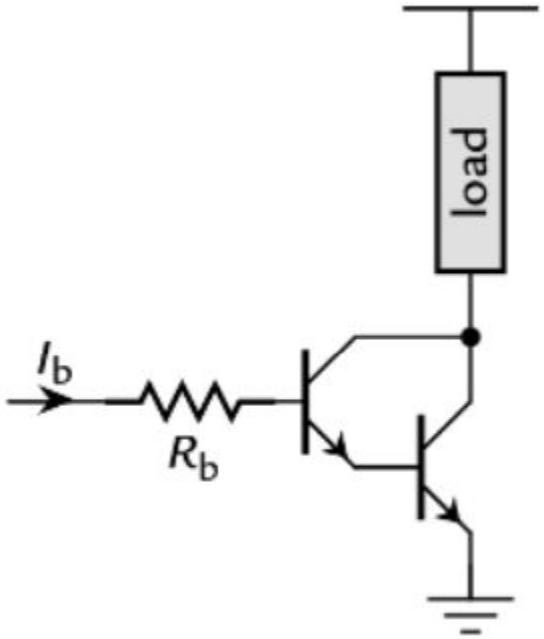
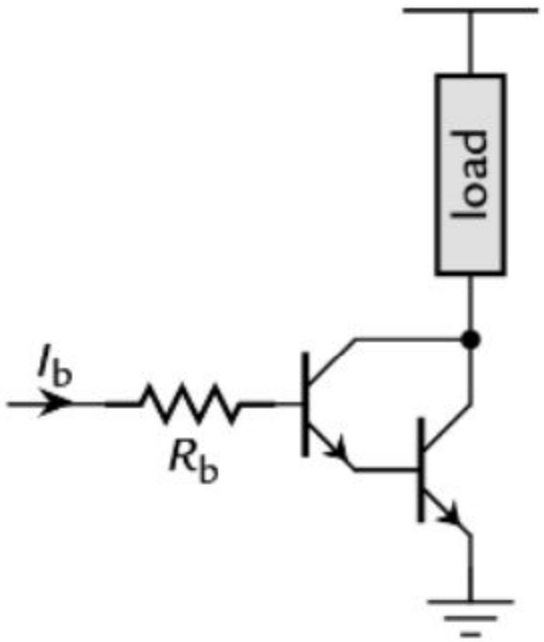
1. Projete o hardware necessário para o MSP430 controlar um motor DC de 12V e 4A. Utilize transistores bipolares de junção (TBJ) com Vbe = 0,7 V, beta = 100 e Vce(saturação) = 0,2 V. Além disso, considere que Vcc = 3 V para o MSP430, e que este não pode fornecer mais do que 10 mA por porta digital.



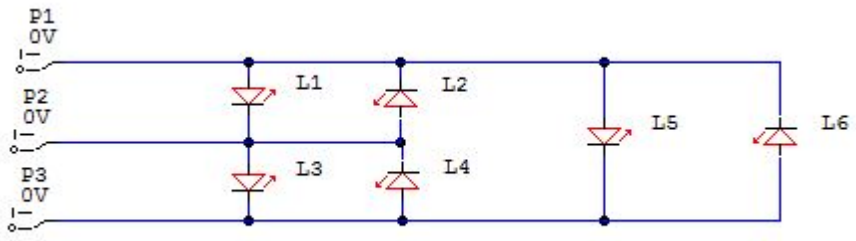
Ic = 4A
Ib = Ic/beta² = 4/100² = 0,4mA
Rb = (Vcc - (2 x Vbe))/Ib = (3 - (2 x 0,7))/0,0004 = 4000Ω

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.



Ic = 1A
Ib = Ic/beta² = 1/120² = 0,069mA
Rb = (Vcc - (2 x Vbe))/Ib = (3,5 - (2 x 0,7))/0,000069 = 30434,78Ω

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



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.

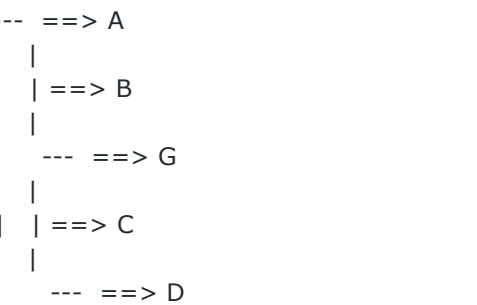
```
#include <msp430g2553.h>


void main(void)
{
  WDTCTL = WDTPW | WDTHOLD;
  while(1)
  {
    P1DIR = BIT1 + BIT2; // 1=1, 2=0, 3=IN   L1
    P1OUT = BIT1;
    P1DIR = BIT1 + BIT2; // 1=0, 2=1, 3=IN   L2
    P1OUT = BIT2;
    P1DIR = BIT1 + BIT3; // 1=1, 2=IN, 3=0   L5
    P1OUT = BIT1;
    P1DIR = BIT1 + BIT3; // 1=0, 2=IN, 3=1   L6
    P1OUT = BIT3;
    P1DIR = BIT2 + BIT3; // 1=IN, 2=1, 3=0   L3
    P1OUT = BIT2;
    P1DIR = BIT2 + BIT3; // 1=IN, 2=0, 3=1   L4
    P1OUT = BIT3;
  }
}
```

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.

```
#include <msp430g2553.h>
```

```
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW | WDTHOLD;
  while(1)
  {
    for(i=0;i<=10000;i++)
    {
      P1DIR = BIT1 + BIT2; // 1=1, 2=0, 3=IN   L1
      P1OUT = BIT1;
      P1DIR = BIT1 + BIT2; // 1=0, 2=1, 3=IN   L2
      P1OUT = BIT2;
    }
    for(i=0;i<=10000;i++)
    {
      P1DIR = BIT2 + BIT3; // 1=IN, 2=1, 3=0   L3
      P1OUT = BIT2;
      P1DIR = BIT2 + BIT3; // 1=IN, 2=0, 3=1   L4
      P1OUT = BIT3;
    }
    for(i=0;i<=10000;i++)
    {
      P1DIR = BIT1 + BIT3; // 1=1, 2=IN, 3=0   L5
      P1OUT = BIT1;
      P1DIR = BIT1 + BIT3; // 1=0, 2=IN, 3=1   L6
      P1OUT = BIT3;
    }
  }
}
```

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.




```
#include <msp430g2553.h>

#define A BIT0
#define B BIT1
#define C BIT2
#define D BIT3
```

```
#define E BIT4
#define F BIT5
#define G BIT6

void EscreveDigito(volatile char dig)
{
  WDTCTL = WDTPW | WDTHOLD;
  P1DIR = A + B + C + D + E + F + G; //
  while(1)
  {
    if(dig == '0')
    {
      P1OUT = A + B + C + D + E + F; //0
    }
    if(dig == '0')
    {
      P1OUT = B + C; //1
    }
    if(dig == '0')
    {
      P1OUT = A + B + E + G + D; //2
    }
    if(dig == '0')
    {
      P1OUT = A + B + G + C + D; //3
    }
    if(dig == '0')
    {
      P1OUT = F + G + B + C; //4
    }
    if(dig == '0')
    {
      P1OUT = A + F + G +C + D; //5
    }
    if(dig == '0')
    {
      P1OUT = A + C + D + E + F + G; //6
    }
    if(dig == '0')
    {
      P1OUT = A + B + C; //7
    }
    if(dig == '0')
    {
      P1OUT = A + B + C + D + E + F + G; //8
    }
    if(dig == '0')
    {
      P1OUT = A + B + C + D + F + G; //9
    }
    if(dig == '0')
    {
      P1OUT = A + B + C + E + F + G; //A
    }
```

```
   if(dig == '0')
   {
    P1OUT = A + B + C + D + E + F + G; //B
   }
   if(dig == '0')
   {
    P1OUT = A + D + E + F; //C
   }
   if(dig == '0')
   {
    P1OUT = A + B + C + D + E + F; //D
   }
   if(dig == '0')
   {
    P1OUT = A + D + E + F + G; //E
   }
   if(dig == '0')
   {
    P1OUT = A + E + F + G; //F
   }
  }

}
```

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

```
#include <msp430g2553.h>

#define A BIT0
#define B BIT1
#define C BIT2
#define D BIT3
#define E BIT4
#define F BIT5
#define G BIT6
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW | WDTHOLD;
  P1DIR = A + B + C + D + E + F + G;
  P2DIR = BIT0 + BIT1; // alternadores
  while(1)
  {
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + B + C + D + E + F; //0
      P2OUT = BIT1;
      P1OUT = A + B + C + D + E + F; //0
    }
    for(i=0;i<=10000;i++)
```

```
{
  P2OUT = BIT0;
  P1OUT = B + C; //1
  P2OUT = BIT1;
  P1OUT = B + C; //1
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + B + E + G + D; //2
  P2OUT = BIT1;
  P1OUT = A + B + E + G + D; //2
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + B + G + C + D; //3
  P2OUT = BIT1;
  P1OUT = A + B + G + C + D; //3
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = F + G + B + C; //4
  P2OUT = BIT1;
  P1OUT = F + G + B + C; //4
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + F + G +C + D; //5
  P2OUT = BIT1;
  P1OUT = A + F + G +C + D; //5
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + C + D + E + F + G; //6
  P2OUT = BIT1;
  P1OUT = A + C + D + E + F + G; //6
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + B + C; //7
  P2OUT = BIT1;
  P1OUT = A + B + C; //7
}
for(i=0;i<=10000;i++)
{
  P2OUT = BIT0;
  P1OUT = A + B + C + D + E + F + G; //8
  P2OUT = BIT1;
  P1OUT = A + B + C + D + E + F + G; //8
```

```
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + B + C + D + F + G; //9
      P2OUT = BIT1;
      P1OUT = A + B + C + D + F + G; //9
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + B + C + E + F + G; //A
      P2OUT = BIT1;
      P1OUT = A + B + C + E + F + G; //A
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + B + C + D + E + F + G; //B
      P2OUT = BIT1;
      P1OUT = A + B + C + D + E + F + G; //B
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + D + E + F; //C
      P2OUT = BIT1;
      P1OUT = A + D + E + F; //C
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + B + C + D + E + F; //D
      P2OUT = BIT1;
      P1OUT = A + B + C + D + E + F; //D
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + D + E + F + G; //E
      P2OUT = BIT1;
      P1OUT = A + D + E + F + G; //E
    }
    for(i=0;i<=10000;i++)
    {
      P2OUT = BIT0;
      P1OUT = A + E + F + G; //F
      P2OUT = BIT1;
      P1OUT = A + E + F + G; //F
    }
  }
}
```