a) Configuração inicial do uCon. Considere os sinais de clock: MCLK = DCOCLK = 1 MHz; SMCLK = DCOCLK/4 = 250 kHz.

```
void ini_uCon(void){

  WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

  DCOCTL = CALDCO_1MHZ; //Freq. calibrada de 1MHZ
  BCSCTL1 = CALBC1_1MHZ;
  BCSCTL2 = DIVS1; //Fator de divisão de 4 para SMCLK
  BCSCTL3 = XCAP0 + XCAP1; //Capacitor do cristal ~12.5 pF

  while(BCSCTL3 & LFXT1OF); //aguardar o oscilador LFXT1 atingir a freq. do cristal
  __enable_interrupt(); //permite as interrupções
}
```

b) Configuração inicial das Portas 1 e 2. **Os pinos não conectados**/utilizados devem ser configurados como saída em nível baixo.

```
void ini_P1_P2(void){
  P1DIR ~= BIT4; //P1.4 é entrada
  P1REN = 0; //P1.4 sem resistor
  P1OUT = 0; //Sem resistor
  P1IES = BIT4; //borda de interrupção de descida
  P1IFG = 0; //Limpa a flag de interrupção
  P1IE = BIT4; //habilita requisição de interrupção

  P2SEL = BIT6 + BIT7; //Função XIN e XOUT em P2.6 e P2.7
  P2DIR = BIT6 + BIT7; //BIT7 e BIT6 (não utilizado) como saída
  P2REN = 0; //entradas sem resistores
  P2OUT = 0; //P2.7 e P2.6 em nível baixo

  P2IES = BIT3+BIT4+BIT5;
  P2IFG = 0;
  P2IE = BIT3+BIT4+BIT5;
}
```

c) Inicialização do Timer 0 para **tempo de debounce (25 ms)**. O contador deve ficar no modo parado (stop) e passar para o modo Up na RTI da porta.

```
void ini_Timer0_debouncer(void){
  // Considere os sinais de clock: MCLK = DCOCLK = 1 MHz; SMCLK = DCOCLK/4 = 250 kHz.
  TA0CTL = TASSEL1; //TASSEL1 = SMCLK
  TA0CCTL0 = CCIE;
  TA0CCR0 = 6249; //(25ms)*(250kHz)-1
}
```

d) Inicialização do Timer 1 para a **temporização t(S1)**. O contador deve ficar no modo parado (stop) até ser ativado quando **Sin** for para nível alto. Considerar um tempo base de **1 segundo**. A temporização **t(S1)** será atingida na RTI desse timer, incrementando-se uma variável **tempo**.

```
void ini_Timer1_tS1(void){

   TA0CTL |= MC0+ID1; //função Up e FDIV de 4
  TA0CCTL0 = CCIE;
  TA0CCR0 = 62499; //((1s)*(250kHz))/4-1
}
```

e) RTI da Porta 1. A CPU vai entrar nessa RTI quando o sinal do sensor (Sin) for para nível alto. Então, deve-se verificar se **Sin** realmente está em nível alto (não precisa fazer o debouncer, pois Sin não provém de uma chave). Se tiver, a **temporização t(S1)** deve ser estabelecida pela leitura da chave S1 via função void read_tS1(void), a lâmpada dever ser ligada e iniciado o temporizador de **t(S1)**.

```
#pragma vector=PORT1_VECTOR
__interrupt void RTI_Porta_1(void){
  P1IFG = 0;

  if((~P1IN) & BIT4)//verificar se esta em nivel alto{
     read_tS1();  // Leitura de t(S1)
     tempo = 0; // Zera variável global de temporizacao
     P2OUT |= BIT7; // Liga Lamp.
     ini_Timer1_tS1(); // Inicia temporizador de t(S1)
  }
}
```

f) Função void read_tS1(void). Nessa função a **chave S2** deve ser lida e um valor de 5 a 120 segundos deve ser atribuído à variável global **tS1**, referente ao tempo **tS1** que será temporizado.

```
void read_tS1(void){
  encoder_in = (~P2IN) & (BIT0 + BIT1 + BIT2);

  if (encoder_in == 0) tS1 = 5;

  if (encoder_in == 1) tS1 = 10;

  if (encoder_in == 2) tS1 = 20;

  if (encoder_in == 3) tS1 = 30;

  if (encoder_in == 4) tS1 = 40;

  if (encoder_in == 5) tS1 = 50;

  if (encoder_in == 6) tS1 = 60;

  if (encoder_in == 7) tS1 = 120;
}
```

g) RTI do módulo 0 do Timer 1. A CPU vai entrar nessa RTI a cada 1 segundo. A variável global **tempo** deve ser incrementada a até atingir o valor de **tS1**. Ao atingir essa temporização, a lâmpada deve ser desligada e esse temporizador de ser parado.

```
#pragma vector=TIMER1_A0_VECTOR
__interrupt void RTI_Modulo_0_Timer_1(void){

  if(tempo >= tS1){
     tempo = 0; //reseta
     P2OUT &= ~BIT7; //desliga o LED
     TA0CTL &= ~MC0; //para o temporizador e volta modo Stop
  }else{
    tempo++;
  }
}
```

h) RTI da Porta 2. A CPU vai entrar nessa RTI quando houver uma mudança na posição da chave **S2**. O debouncer para essas chaves deve ser realizado. As interrupções associadas às entradas das chaves de S2 devem ser desabilitadas e o temporizador do debouncer iniciado.

```
#pragma vector=PORT2_VECTOR
__interrupt void RTI_Porta_2(void){
   P2IE &= ~(BIT3 + BIT4 + BIT5); //Desabilita interrupções das chaves S2
  TA0CTL |= MC0; //Iniciar o contador
}
```

i) RTI do módulo 0 do Timer 0 para concluir o debouncer de S2. Nessa RTI deve-se verificar qual das chaves de S2 foi selecionada pelo usuário e associar uma valor para a variável global **modo** (0 - modo desligado; 1 - modo sempre ligado; 2 - modo automático). No final, a função **set_modo(modo)** é chamada para realizar as configurações necessárias para o modo selecionado.

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void RTI_Modulo_0_Timer_0(void){

  ?????;

  switch( P2IFG & (BIT3 + BIT4 + BIT5) ){
     case BIT3:
        if(P2IN & BIT3){
           P2IFG &= ~BIT3;
           modo = 0; //desligado
        }
        break;
   case BIT4:
        if(P2IN & BIT4){
           P2IFG &= ~BIT4;
           modo = 1; //sempre ligado
        }
        break;

     case BIT5:
        if(P2IN & BIT5){
           P2IFG &= ~BIT5;
           modo = 2; //automático
        }
        break;

     default:   // Em caso de falha na leitura de S2
        ???????;
        modo = 0;  // modo DESLIGADO
        break;
  }
  P2IE |= BIT3 + BIT4 + BIT5; //habilita interrupção novamente

  set_modo(modo);
}
```

j) Função void set_modo(unsigned char mode) para estabelecer as configurações para o modo selecionado.

```
void set_modo(unsigned char mode){

  switch(mode){
  case 0: // modo DESLIGADO
     // DESLIGA Lamp., desabilita int. de P1.4, Timer1 parado
     P2OUT &= ~BIT7;
     P1IE &= ~BIT4;
     TA1CTL &= ~MC0;
     break;
  case 1: // modo S_LIGADO
     P2OUT |= BIT7; // Liga o LED
     P1IE &= ~BIT4; // Desabilita interrupção de P1.4
     TA1CTL &= ~MC0; // Timer1 parado
     break;
  case 2:
     P2OUT &= ~BIT7; // Desliga o LED
     P1IE |= BIT4; // Habilita interrupção de P1.4
     TA1CTL |= MC0; // Timer1 habilitado
     tempo = 0;
     break;
  }
}
```