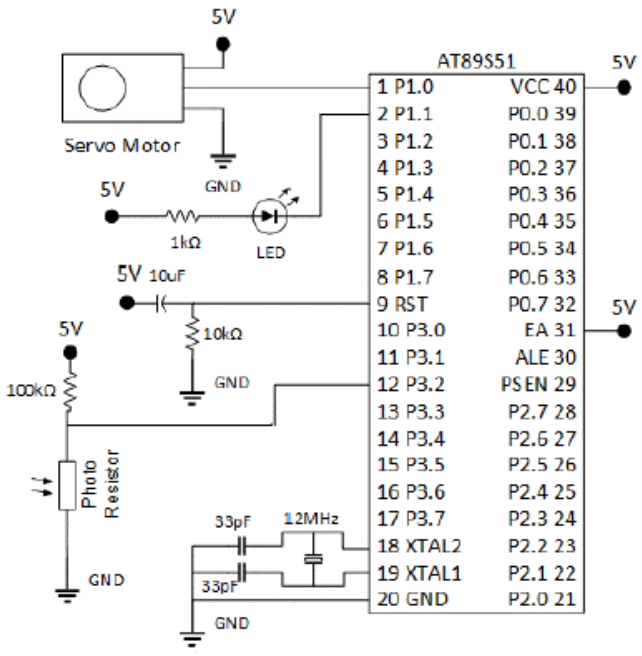
UNIVERSIDADE da MADEIRA

Arquitetura De Computadores

Licenciatura Engenharia Informática

Ano Letivo: 2017 – 2018



Projeto 3 - Light Tracker

Relatório

Docentes:

Dionísio Barros
Nuno Ferreira
Pedro Camacho
Sofia Inácio

Discente:

Lisandro Marote – 2030315

Funchal, 3 de Junho 2018

UNIVERSIDADE da MADEIRA

Índice

Pág.

Introdução 3
Desenvolvimento 4
Conclusão 9
Bibliografia 10
Anexos 11
Fluxogramas 11
Código em Linguagem C 14
Código em Linguagem Assembly 17

Introdução

No âmbito da cadeira de Arquitetura de Computadores foi-nos proposta a realização de um projeto, tendo em vista a implementação de um programa que visa o funcionamento de um "*Ligth Tracker*" em linguagem de programação C e linguagem de baixo nível *assembly*.

Para o desenvolvimento do programa utilizou-se o *software* disponibilizado pelos docentes, mais concretamente o programa *Keil*, onde efetuou-se as devidas simulações do programa quer em C quer em linguagem *assembly*, onde por base utilizou-se o microcontrolador AT89S51.

Para realização deste projeto foi efetuado uma análise detalhada, recorrendo a diversos fluxogramas de maneira a poder atingir os objetivos inicialmente propostos.

O circuito elétrico é composto por 4 componentes principais:

. Servo
. Sensor Luz
. LED
. Microcontrolador

O servo / motor tem como função efetuar um varrimento entre 0º e 180º (*e vice-versa*) onde durante esse mesmo varrimento, caso o sensor de Luz detete algum sinal luminoso, interrompe o varrimento do servo e aciona o LED informando que este detetou um sinal luminoso. Este estado mantém-se até que sensor de luz deixe de detetar um sinal luminoso voltando a efetuar constantemente varrimentos entre os valores referidos anteriormente. Estes componentes estão ligados a pinos específicos do microcontrolador, onde servo está conectado ao pino 1.0, LED ao pino 1.1 e por fim sensor luz ao pino 3.2.

Objetivos:

Principal objetivo deste projeto é dominar e adquirir conhecimentos sobre a linguagem *Assembly* e C, consolidar os conhecimentos adquiridos quer nas aulas teóricas, teórico-práticas e práticas laboratoriais, compreender como utilizar as interrupções e aprender a projetar um projeto em 2 linguagens de programação distintas bem como saber efetuar um mapeamento das instruções e interrupções para o microcontrolador AT89S51.

Desenvolvimento

O microcontrolador AT89S51 possui cinco fontes de interrupções, duas externas (*INT0 e INT1*), duas de temporização (*Timer0 e Timer1*) e uma de série. Para projeção deste trabalho utilizou-se a interrupção externa bem como a interrupção de tempo, tendo como base de desenvolvimento os exercícios realizados nas aulas laboratoriais da cadeira.

A interrupção externa 0 é utilizada para o sensor de luz, onde caso este detete algum sinal luminoso faz acionar o LED (*este funciona em lógica negada*).

A interrupção de temporização 0 é utilizada para efetuar o incremento dos ciclos máquina.

Para o desenvolvimento do código em linguagem alto nível C, utilizou-se por base inicialmente o exercício da aula laboratorial 13, onde efetuou-se as alterações necessárias para cumprir com o pretendido. A constante "fimTempo" foi estabelecida com base no seguinte cálculo (*Figura 1*), garantido assim que cada pulso no servo esteja sempre entre 20 ms.

```
#define fimTempo 100                    // timer = 0.2ms -> fimTempo = 100*0.2ms = 20ms
```

Figura 1- Cálculo para fimTempo.

Foi também estabelecido o valor das constantes "atraso", "centoOitenta" e "zero" e feito os cálculos como demostra a *Figura 2*.

```
#define fimTempo 100                    // timer = 0.2ms -> fimTempo = 100*0.2ms = 20ms
#define zero 3                          // 3*0.2ms = 0.6ms -> 0°
#define centoOitenta 12                 // 12*0.2 = 2.4ms -> 180°
#define atraso 10                       // 10*20ms = 0.2s
```

Figura 2 – Cálculo efetuado para as constantes utilizadas.

Os cálculos foram efetuados de maneira a cumprir o Anexo I disponibilizado no enunciado do projeto, respeitando assim os tipos de sinais de controlo do servo motor (*Figura 3*).

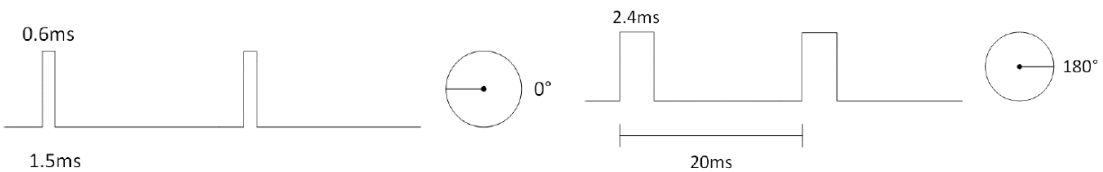


Figura 3 – Sinais de controlo do servo motor.

Seguidamente declarou-se como pinos de controlo o servo, Led e sensorLuz nos pinos designados no enunciado do projeto *(Figura 4).*

```
sbit servo = P1^0;                    // Pino De Controlo Para Servo Motor.
sbit LED = P1^1;                      // Pino De Controlo Para o LED.
sbit sensorLuz = P3^2;                // Pino De Controlo Para O Sensor De Luz.
```

Figura 4 – Atribuição dos pinos para o microcontrolador.

Foi necessário a utilização de uma variável auxiliar responsável pela direção que o servo motor teria de tomar (*dir*), onde caso esta seja 0 o servo motor estará a tomar a direção entre 0º e 180º, já quando esta toma o valor de 1 servo motor terá a direção entre 180º e 0º (*Figura 5*).

```
unsigned char conta = 0;              // Contador Que Conta a Cada 200us.
unsigned char conta2 = 0;             // Tempo de Espera Entre Mudança De Ângulos.
unsigned char referencia = zero;      // O Servo Começa Nos 0°.
unsigned int dir = 0;                 // Variável Auxiliar Para Direção Do Servo. (0-> 180° e 1-> 0°).
```

Figura 5 – Atribuição de valores às variáveis.

Já na configuração das interrupções externas, TMOD, timer0, TCON e inicialização do LED foi implementado como mostra a *Figura 6.*

```
// Declaração De Funções.
void Init(void) {
    // Configuracao Registo IE
    EA = 1;                           // Activa Interrupções Globais.
    ET0 = 1;                          // Activa Interrupção timer 0.
    EX0 = 1;                          // Ativa Interrupção Externa 0.
    // Configuracao Registo TMOD
    TMOD &= 0xF0;                     // Limpa Os 4 bits Do timer 0 (8 bits - auto reload).
    TMOD |= 0x02;                     // Modo 2 Do Timer 0.
    // Configuracao Timer 0
    TH0 = 0x37;                       // Timer 0 - 200us.
    TL0 = 0x37;
    // Configuracao Registo TCON
    TR0 = 1;                          // Começa O timer 0.
    IT0 = 1;                          // Interrupção Externa Activa A Falling Edge.
    // Configuração Do LED.
    LED = 1;                          // Inicialização Do LED A 1 (Funciona Pela Lógica Negada).
}
```

Figura 6 – Inicializações e configuração interrupções e timer.

Optou-se por fazer a inicialização do LED dentro da função inicializações, tendo o cuidado que este funciona por lógica negada, ao invés de declarar no main principal, tendo isto sido uma decisão de maneira a estruturar e organizar o código em linguagem C melhor possível, de maneira que a sua compreensão e coerência seja mais clara possível.

Já para no tratamento da interrupção externa, é feito o acionamento do LED para indicar que este recebeu um sinal luminoso diretamente do sensor de luz. Já na interrupção do timer 0 este efetua as contagens de 200 ciclos máquina continuamente (*Figura 7*).

```
// Interrupcao Externa.
void External0_ISR(void) interrupt 0 {
    LED = 0;                                    // Ativação Do LED (Lógica Negada Uma Vez Mais).
}

// Interrupcao Tempo.
void Timer0_ISR(void) interrupt 1 {
    conta++;                                    // Incrementa a Cada Contagem de 200us.
}
```

Figura 7 – Tratamento das interrupções timer0 e externa0.

Já no main principal, utilizou-se o loop infinito para verificar todas as condições constantemente, sendo que a solução encontrada para ter mais do 1 ângulo seria incrementar o valor da referência de 1 a 1, tendo noção que a referencia poderá ter como valor mínimo 3 (*0 graus*) e valor máximo 12 (*180 graus*). A *Figura 8* representa as condições para que o servo atinga o novo valor de referência (novo ângulo), tendo na primeira condição feito uma pequena alteração de maneira que quando seja detetado um sinal luminoso o servo fique bloqueado mandando constantemente o mesmo pulso para o mesmo.

```
void main (void) {
// Inicializações.
    Init();
    while(1)                                    // Loop Infinito.
    {
        // Atingiu O Valor De Referência (0.6ms, 1.5ms ou 2.6ms).
            if(conta == referencia)
            {
                servo = 0;                      // Coloca a Saída a 0 Até Atingir Os 20ms.
                if(sensorLuz == 0) conta2 = 0;  // Mantem o Servo Bloqueado Enquanto Sensor De Luz Estiver A Receber Um Sinal Luminoso.
            }

            // Atingiu Os 20ms
            if(conta == fimTempo)
            {
                conta = 0;                      // Reinicia A Contagem.
                servo = 1;                      // Impulso Positivo Até Atingir O Valor De Referencia.
                conta2++;                       // Incrementa A Variável Do Tempo De Espera Entre Os Ângulos.
            }
```

Figura 8 – Condições para servo atingir novas referências.

```
            //Se Atingiu o Tempo Definido De Espera Para Mudar De Ângulo.
            if(sensorLuz == 1)
            {
                if(conta2 == atraso)
                {
                    LED = 1;
                    if(dir == 0)
                    {
                        if(referencia == centoOitenta)
                        {
                            dir = 1;                // Muda Direção A Começar Dos 180° Para 0°.
                        }
                            else
                            {
                                referencia+=1;      // Incrementação Da Referência Até Atingir os 180°.
                                conta2 = 0;         // Volta a Reiniciar a Contagem.
                            }
                    }
                        else
                        {
                        if(referencia == zero)
                        {
                            dir = 0;            // Muda Direção A Começar Dos 0° Para 18+p0°.
                        }
                            else
                            {
                                referencia-=1;
                                conta2 = 0;     // Volta a Reiniciar A Contagem.
                            }
                        }
                    }
                }
    }                                       // Fim Do While.
}                                           // Fim Do Void Main.
```

Figura 9 – Condições responsáveis pela movimentação do servo motor.

Para efetuar o varrimento de 180 graus para 0 graus e vice-versa utilizou-se tal como referido anteriormente uma variável auxiliar (*dir*), onde sua principal funcionalidade é indicar que o valor da referência deve de incrementar no caso de estar a ir para 180 graus ou decrementar no caso de estar a vir para os 0 graus. É indicado para o LED estar com valor lógico a 1 de maneira a garantir que este não esteja ligado no caso de o servo motor estiver a fazer as condições de movimentação, salientando que este funciona em lógica negada.

Para a linguagem de baixo nível assembly, foi adotado o mapeamento direto da linguagem alto nível C linha a linha, tido isto levado a poupar imenso tempo no desenvolvimento do mesmo e na própria extensão do programa em si. Houve a particularidade de efetuar saltos específicos onde foi uma necessária pesquisa e compreensão do mesmo de maneira a efetuar o pretendido. Utilizou-se 5 registos (R0, R1, R2, R3 e R4) para as variáveis conta, conta2, referência, sensorluz e por fim para a variável auxiliar dir *Figura 10.*

```
; ------------------------------------------------------------
; Registos
; R0 - conta.
; R1 - conta2.
; R2 - referencia.
; R3 - sensor luz.
; R4 - (Dir) Variável Auxiliar Responsável Pela Direção Do Servo.
; ------------------------------------------------------------
```

Figura 10 – Registos utilizados para linguagem assembly.

Os saltos utilizados foram jumb if bit = 1, jump if bit = 0 e jump if bit != #data.

```
BEGIN:                                  ; Inicialização Do Loop Infinito.
    MOV A, R0
    MOV B, R2
    CJNE A, B, AtingiuOs20              ; Caso Conta Seja Diferente A Referência Salta Para AtingiuOs20.
    CLR servo                           ; Coloca Servo A 0.
    JB sensorLuz, AtingiuOs20           ; Caso Sensor De Luz Esteja A Receber Sinal Luminoso.
    MOV R1, #0                          ; Conta2 Igual A 0 (Mantém Servo Bloqueado).

AtingiuOs20:
    MOV A, R0                           ; Coloca Conta No Acumulador.
    CJNE R0, #FIMTEMPO, sinalLuminoso   ; Caso Conta Atinga Fim De Tempo Salta Para Rotina Sinal Luminoso.
    MOV R0, #0                          ; Reinicia A Contagem Do Conta.
    SETB servo                          ; Coloca O Servo A 1 Para Atingir Valor Da Referencia.
    INC R1                              ; Incrementa O Conta2.
```

Figura 11 – Exemplo tipos de saltos utilizados em linguagem assembly.

Conclusão

Após a realização deste projeto, podemos concluir que todos os objetivos inicialmente propostos foram alcançados, tendo assim sido possível projetar o programa quer em linguagem de programação C quer em linguagem assembly com todas as funcionalidades pretendidas.

No decorrer deste projeto foram encontradas algumas dificuldades, tais como os pequenos pormenores que seria esperado na programação do projeto, mais especificamente ao ser detetado um sinal luminoso o servo teria de receber pulsos constantemente para manter a sua posição ao invés de cortar por completo a energia para o mesmo. Uma outra dificuldade sentida foi no desenvolvimento da linguagem de programação assembly, dificuldades tais como saltos específicos, onde recorrendo à sebenta disponibilizada pelos docentes esta mesma dificuldade foi ultrapassada, compreendendo melhor as várias diferenças que existe nos saltos em linguagem assembly.

Concluindo, com este projeto foi adquirido mais experiência e uma melhor compreensão da linguagem de programação assembly e C para o microcontrolador utilizado. Tivemos oportunidade de por em prática os conhecimentos adquiridos nas aulas teóricas, teórico-práticas e práticas laboratoriais da cadeira.

Bibliografia

. DELGADO, José; RIBEIRO, Carlos; *Arquitetura de Computadores,* FCA, 2007.

. Aula Prática Laboratorial 11, *Sistemas Embebidos – Microcontrolador 80C51*

http://moodle.cee.uma.pt/pluginfile.php/43183/mod_resource/content/4/11%C2%AA%20Aula%20Pr%C3%A1tica%20laboratorial.pdf

. Aula Prática Laboratorial 12, *Sistemas Embebidos – Microcontrolador 80C51*

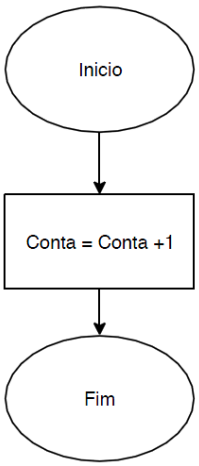
http://moodle.cee.uma.pt/pluginfile.php/49035/mod_resource/content/1/Guia%20Aula%20Pr%C3%A1tica%2012.pdf

. Aula Prática Laboratorial 13, *Sistemas Embebidos – Microcontrolador 80C51*

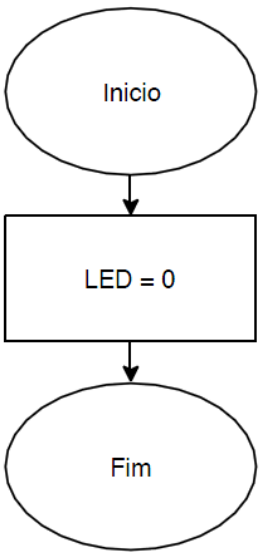
http://moodle.cee.uma.pt/pluginfile.php/43308/mod_resource/content/3/13%C2%AA%20Aula%20Pr%C3%A1tica%20laboratorial.pdf

Anexos

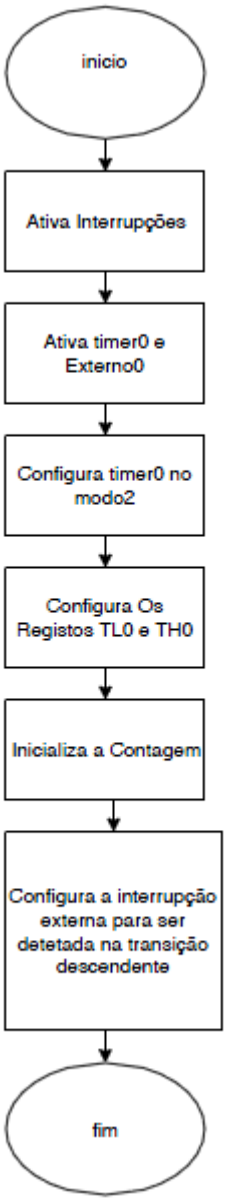
Fluxogramas



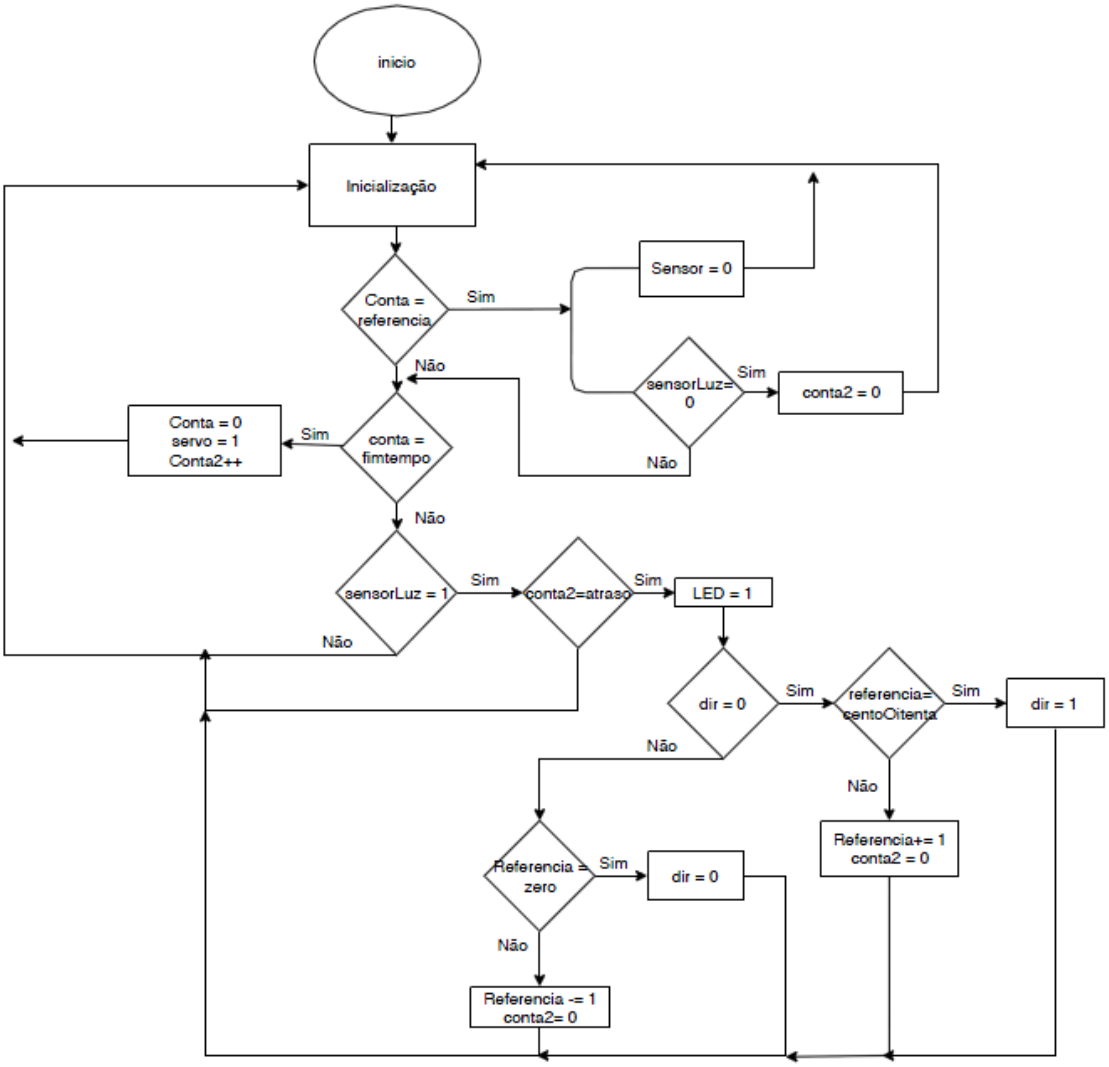
timer0



Externa0



Inicializações


inicio
Inicialização
Conta = referencia
Sim
Sensor = 0
Não
sensorLuz= 0
Sim
conta2 = 0
Não
Conta = 0
servo = 1
Conta2++
Sim
conta = fimtempo
Não
sensorLuz = 1
Sim
conta2=atraso
Sim
LED = 1
Não
dir = 0
Sim
referencia= centoOitenta
Sim
dir = 1
Não
Não
Referencia = zero
Sim
dir = 0
Não
Referencia+= 1
conta2 = 0
Referencia -= 1
conta2= 0

Principal

Código em Linguagem C

```
#include <reg51.h>
#define fimTempo 100                                    // timer = 0.2ms -
> fimTempo = 100*0.2ms = 20ms
#define zero 3                                          // 3*0.2ms = 0.6ms
-> 0
#define centoOitenta 12                                 // 12*0.2 = 2.4ms
-> 180
#define atraso 10                                       // 10*20ms = 0.2s


sbit servo = P1^0;                                      // Pino De
Controlo Para Servo Motor.
sbit LED = P1^1;                                        // Pino De
Controlo Para o LED.
sbit sensorLuz = P3^2;                                  // Pino De
Controlo Para O Sensor De Luz.

unsigned char conta = 0;                                // Contador Que
Conta a Cada 200us.
unsigned char conta2 = 0;                               // Tempo de Espera
Entre Mudan硃De ®gulos.
unsigned char referencia = zero;                        // O Servo Come硃
Nos 0.
unsigned int dir = 0;                                   // Vari'l Auxiliar
Para Dire磯 Do Servo. (0-> 180 e 1-> 0).

// Declara磯 De Fun紱s.
void Init(void) {
    // Configuracao Registo IE
    EA = 1;                                             // Activa Interrup
紱s Globais.
    ET0 = 1;                                            // Activa Interrup
磯 timer 0.
    EX0 = 1;                                            // Ativa Interrup磯
Externa 0.
    // Configuracao Registo TMOD
    TMOD &= 0xF0;                                       // Limpa Os 4 bits
Do timer 0 (8 bits  auto reload).
    TMOD |= 0x02;                                       // Modo 2 Do Timer
0.
    // Configuracao Timer 0
    TH0 = 0x37;                                         // Timer 0 -
200us.
    TL0 = 0x37;
    // Configuracao Registo TCON
    TR0 = 1;                                            // Come硃O timer 0.
    IT0 = 1;                                            // Interrup磯
Externa Activa A Falling Edge.
    // Configura磯 Do LED.
    LED = 1;                                            // Inicializa磯 Do
LED A 1 (Funciona Pela L□a Negada).
}

// Interrupcao Externa.
```

```
void External0_ISR(void) interrupt 0 {
    LED = 0;                                        // Ativa磯 Do LED
(L□a Negada Uma Vez Mais).
}

// Interrupcao Tempo.
void Timer0_ISR(void) interrupt 1 {
    conta++;                                        // Incrementa a
Cada Contagem de 200us.
}

void main (void) {
// Inicializa紱s.
    Init();
    while(1)                                        // Loop Infinito.
    {
        // Atingiu O Valor De Refer□ia (0.6ms, 1.5ms ou 2.6ms).
            if(conta == referencia)
            {
                servo = 0;                          // Coloca a Sa a 0
At頁tingir Os 20ms.
                if(sensorLuz == 0) conta2 = 0;      // Mantem o Servo
Bloqueado Enquanto Sensor De Luz Estiver A Receber Um Sinal Luminoso.
            }

            // Atingiu Os 20ms
            if(conta == fimTempo)
            {
                conta = 0;                          // Reinicia A
Contagem.
                servo = 1;                          // Impulso
Positivo At頁tingir O Valor De Referencia.
                conta2++;                           // Incrementa A
Vari'l Do Tempo De Espera Entre Os ®gulos.
            }
            //Se Atingiu o Tempo Definido De Espera Para Mudar De
®gulo.
            if(sensorLuz == 1)
            {
                if(conta2 == atraso)
                {
                    LED = 1;
                    if(dir == 0)
                    {
                        if(referencia == centoOitenta)
                        {
                            dir = 1;                // Muda Dire磯 A
Come硲 Dos 180 Para 0.
                        }
                            else
                            {
                                referencia+=1;      // Incrementa磯 Da
Refer□ia At頁tingir os 180.
                                conta2 = 0;         // Volta a
Reiniciar a Contagem.
                            }
                    }
                        else
```

```
                    {
                        if(referencia == zero)
                        {
                            dir = 0;            // Muda Dire磯 A
Come硲 Dos 0 Para 18+p0.
                        }
                            else
                            {
                                referencia-=1;
                                conta2 = 0;     // Volta a
Reiniciar A Contagem.
                            }
                    }
                }
            }
    }                                           // Fim Do While.
}                                               // Fim Do Void
Main.
```

Código em Linguagem Assembly

```
; ------------------- Definição De Constantes -------------------
FIMTEMPO        EQU         100                 ; timer = 0.2ms ->
fimTempo = 100*0.2ms = 20ms
ZERO            EQU         3                   ; 3*0.2ms = 0.6ms ->
0°
CENTOOITENTA    EQU         12                  ; 12*0.2 = 2.4ms ->
180°
ATRASO          EQU         10                  ; 10*20ms = 0.2s

; ------------------- Definição De Portas -------------------
servo           EQU         P1.0                ; Pino Para Controlo
Do Servo.
LED             EQU         P1.1                ; Pino De Acionamento
Do LED Quando Deteta Um Sinal Luminoso.
sensorLuz       EQU         P3.2                ; Pino De Acionamento
Do Sinal Luminoso.

; ------------------------------------------------------------
; Primeira Instrução, Após o Reset Do MicroControlador.
CSEG AT 000H
    JMP MAIN

; Se Ocorrer A Interrupção Externa 0.
CSEG AT 0003h
    JMP  InterrupcaoExt0

; Tratamento Da Interrupção De Temporização 0, Para Contar 20ms.
CSEG AT 000Bh
    JMP  InterrupcaoTemp0

CSEG AT 0050H
MAIN:
    LCALL INICIO

BEGIN:                                          ; Inicialização Do
Loop Infinito.
    MOV A, R0
    MOV B, R2
    CJNE A, B, AtingiuOs20                      ; Caso Conta Seja
Diferente A Referência Salta Para AtingiuOs20.
    CLR servo                                   ; Coloca Servo A 0.
    JB sensorLuz, AtingiuOs20                   ; Caso Sensor De Luz
Esteja A Receber Sinal Luminoso.
    MOV R1, #0                                  ; Conta2 Igual A 0
(Mantém Servo Bloqueado).

AtingiuOs20:
    MOV A, R0                                   ; Coloca Conta No
Acumulador.
    CJNE R0, #FIMTEMPO, sinalLuminoso           ; Caso Conta Atinga
Fim De Tempo Salta Para Rotina Sinal Luminoso.
    MOV R0, #0                                  ; Reinicia A Contagem
Do Conta.
    SETB servo                                  ; Coloca O Servo A 1
Para Atingir Valor Da Referencia.
    INC R1                                      ; Incrementa O Conta2.
```

```
sinalLuminoso:
    JNB sensorLuz, BEGIN                        ; Avança Caso Sensor
De Luz Não Esteja Ativo.
    MOV A, R1
    CJNE A, #ATRASO, BEGIN                      ; Caso Conta2 Seja
Diferente De Atraso Salta Novamente Para Begin.
    SETB LED                                    ; Continua A Enviar
Sinal Para LED Continuar Desativado.
    MOV A, R4
    CJNE R4, #0 , direcaoZero                   ; Caso A Variável DIR
Seja Diferente Zero, Salta Para direcaoZero.
    MOV A, R2
    CJNE A, #CENTOOITENTA, reiniciaContagem ; Caso A DIR Não Esteja Na
Referência 180° Salta Para  reiniciaContagem.
    MOV R4, #1                                  ; Coloca a DIR A 1
(180° Para 0°).
    JMP BEGIN

reiniciaContagem:
    INC R2                                      ; Incrementa A
Referência.
    MOV R1, #0                                  ; Volta A Reiniciar A
Contagem.
    JMP BEGIN

direcaoZero:
    MOV A, R2
    CJNE A, #ZERO, direcaoCentoOitenta          ; Caso Referência Seja
Diferente De Zero, Salta Para direcaoCentoOitenta.
    MOV R4, #0                                  ; Coloca a DIR A 0 (0°
Para 180°).
    JMP BEGIN

direcaoCentoOitenta:
    DEC R2                                      ; Decrementa O Valor
Da Referência.
    MOV R1, #0                                  ; Volta A Reiniciar A
Contagem.

JMP BEGIN                                       ; Loop Infinito Fim.

; ---------------------------------------------------------
; Registos
; R0 - conta.
; R1 - conta2.
; R2 - referencia.
; R3 - sensor luz.
; R4 - (Dir) Variável Auxiliar Responsável Pela Direção Do Servo.
; ---------------------------------------------------------
;Tratamento Da Interrupção Externa.
InterrupcaoExt0:
            CLR LED                             ; Coloca O LED A 0, Ou
Seja Liga (Funciona Pela Lógica Negada).
            RETI

; Tratamento Da Interrupção De Temporização.
InterrupcaoTemp0:
            INC R0                              ; Incrementa O Conta.
            RETI
```

```
; ---------------------------------------------------------
; Ciclo Do Programa Inicial.
INICIO:

    MOV IE ,#10000011B
                                                ; Activa Interrupções
Globais.
                                                ; Activa Interrupção
timer 0.
                                                ; Ativa Interrupção
Externa 0.
    ; Configuração Do Registo TMOD
    MOV TMOD, #00000010B
                                                ; Limpa Os 4 bits Do
timer 0 (8 bits - auto reload).
                                                ; Modo 2 Do Timer 0.
    ; Configuração Timer 0
    MOV TH0, #037H                              ; Timer 0 - 200us.
    MOV TL0, #037H
    ; Configuração Registo TCON
    SETB TR0                                    ; Começa O timer 0.
    SETB IT0                                    ; Interrupção Externa
Activa A Falling Edge.
    ; Configuração Do LED.
    SETB LED                                    ; Inicialização Do LED
A 1 (Funciona Pela Lógica Negada).

    ; Configuração Dos Registos.
    MOV R2, #ZERO                               ; Atribuição À
Referência Valor Do Zero (3).
    MOV R4, #0                                  ; Atribuição À
Variável DIR Valor De 0.
    MOV R0, #0                                  ; Inicialização Da
Variável Conta Com Valor De 0.
    MOV R1, #0                                  ; Inicialização Da
Variável Conta2 Com Valor De 0.
    MOV R3, #0                                  ; Inicialização Da
Variável Sensor Luz Com Valor De 0.
    RET

    END
```