Universidade da Madeira

Faculdade de Ciências Exatas e da Engenharia

**Arquitetura de computadores**

**Parque de Estacionamento**

Alunos:  
- João Sousa – 2133023  
- Lucas Araújo – 2147123

Docente:

Dionísio Barros

Data: 26 de Maio 2025Index

1. Objetivos
2. Descrição da Solucao e análise de relustados
3. Conclusão
4. Anexo A
5. Anexo B

1.Objetivos

O principal objetivo deste projeto consistiu no desenvolvimento de um sistema funcional de controlo de um parque de estacionamento, utilizando as linguagens de programação C e Assembly, para o microcontrolador AT89S51. Pretendeu-se que o sistema controlasse a entrada e saída de veículos através de botões físicos, e que apenas permitisse a entrada se existissem lugares disponíveis, ou a saída se o parque não estivesse vazio.

Adicionalmente, foi implementado um sistema de barreira automática, que impede o seu encerramento enquanto um veículo estiver por baixo, garantindo assim a segurança do utilizador. Outro dos objetivos propostos foi a utilização de um display de 7 segmentos para apresentar, de forma dinâmica, o número de lugares disponíveis no parque. Este número é determinado com base no estado de 8 LEDs que representam os lugares de estacionamento. Por fim, foram ainda implementados três LEDs sinalizadores: um LED verde para indicar disponibilidade, um LED vermelho para indicar lotação esgotada, e um LED amarelo intermitente enquanto a barreira estiver levantada.

2.Descrição da Solução e Análise de Resultados

A solução foi desenvolvida em paralelo nas linguagens C e Assembly, respeitando os requisitos estabelecidos. O controlo dos botões de entrada e saída foi feito através de interrupções externas (INT0 e INT1), permitindo uma resposta imediata à ação do utilizador. Cada botão ativa uma variável de controlo que determina se o pedido corresponde a uma entrada ou saída.

Para garantir que a barreira não fecha enquanto um carro ainda se encontra por baixo, foi implementado um mecanismo de verificação com base num sensor ótico simulado. Este sensor ativa um sinal de “buffer” ao detetar a presença de um veículo, e apenas permite o fecho da barreira após o carro ter passado completamente.

A contagem de lugares disponíveis é feita através da leitura do porto P0, onde cada bit representa o estado de um lugar. Ao detetar quais LEDs estão ativos (valor 1), o sistema calcula o número de vagas e envia este valor codificado em binário para o porto P2, que alimenta um descodificador ligado a um display de 7 segmentos.

O LED verde acende-se sempre que o número de lugares disponíveis seja superior a zero. Caso contrário, o LED vermelho é ativado para indicar que o parque se encontra cheio. O LED amarelo intermitente foi implementado através de contadores no temporizador, que alternam o seu estado (ligado/desligado) a cada segundo enquanto a barreira estiver aberta.

Em simulação, o sistema comportou-se de forma estável e de acordo com os requisitos. A lógica de controlo respondeu corretamente às entradas dos botões, a barreira funcionou conforme esperado e os LEDs e o display refletiram com precisão o estado do parque.

3.Conclusão

O projeto permitiu aplicar, de forma prática, os conhecimentos adquiridos na programação de sistemas embarcados, utilizando tanto Assembly como a linguagem C com o microcontrolador 8051. A implementação de interrupções, temporização, manipulação de pinos e lógica condicional em baixo nível revelou-se fundamental para o funcionamento correto do sistema.

Todas as funcionalidades pedidas no enunciado foram cumpridas com sucesso: o controlo de entrada e saída de veículos, o bloqueio em caso de parque cheio ou vazio, o acionamento seguro da barreira com verificação de passagem do carro, e a indicação visual do estado do sistema através de LEDs e display.

O trabalho proporcionou uma compreensão aprofundada dos desafios envolvidos no controlo direto de hardware e demonstrou a importância da programação eficiente e segura em sistemas críticos. A experiência obtida será certamente útil em contextos futuros de desenvolvimento de sistemas embarcados e automação.

4.Anexo A:

Fluxograma do progama principal:

A diagram of a computer

AI-generated content may be incorrect.

Fluxograma Interrupcoes externas:

A diagram of a flowchart

AI-generated content may be incorrect.

5.Anexo B:

Codigo em C:

#include <reg51.h>

//constantes

#define fimTempo 5000  // 5000 \*0.20ms = 1s

#define fimTempoBarreira 100// 50 \* 0.20ms = 20ms

#define zero 3 // 3 \* 0.20ms = 0.6ms

#define oitenta 6 // 6 \* 0.20ms = 1.2ms

//Pins P0

sbit L0 = P0^0;

sbit L1 = P0^1;

sbit L2 = P0^2;

sbit L3 = P0^3;

sbit L4 = P0^4;

sbit L5 = P0^5;

sbit L6 = P0^6;

sbit L7 = P0^7;

//Pins P1

sbit ledVerde = P1^0; //pino do led verde

sbit ledVermelho = P1^1; //pino do led vermelho

sbit ledAmarelo = P1^2; //pino do led amarelo

sbit barreira = P1^3; //pino da barreira

sbit sensor = P1^4; //pino do sensor

//Pins P2

sbit segmentoA = P2^0; //pino do servo A

sbit segmentoB = P2^1; //pino do servo B

sbit segmentoC = P2^2; //pino do servo C

sbit segmentoD = P2^3; //pino do servo D

//variaveis globais

bit botao = 0; //se o botao estiver a 0 entao e uma entrada se estiver a 1 e uma saida

bit pressionado = 0; //para confirmar se o botao foi pressionado

bit passouCarro = 0; //confirmar se passou o carro

bit bufferPassouCarro = 0;// buffer para confirmar se o carro passou todo

*unsigned* *int* conta = 0; //para ajudar a contar 1 segundo

*unsigned* *int* contaSegundo = 0; //contador do timer a cada 1s

*unsigned* *int* abreBarreira = 0; //para abrir a barreira

*unsigned* *char* referenciaBarreira = zero; //referencia da barreira

*unsigned* *char* contaBarreira = 0; //contador da barreira

*void* Init(*void*){

    //configuracao Registo IE

    EA = 1; //habilita interrupcao global

    ET0 = 1; //habilita interrupcao do timer 0

    EX0 = 1; //habilita interrupcao externa 0

    EX1 = 1; //habilita interrupcao externa 1

    //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

    TR0 = 1; //inicia o timer 0

    TH0 = 56; //Inicia para ser 256 -56 = 200ms

    TL0 = 56; //Inicia para ser 256 -56 = 200ms

    IT0 = 1;//interrupcao esterna ativa a falling edge

    //inicializacao do P0

    //P0 = ~P0;

    //inicializa o display

    P2 = 0x00; //inicializa o display a 0

    ledVerde = 0; //led verde ligado

    ledVermelho = 1; //led vermelho desligado

    ledAmarelo = 1; //led amarelo desligado

    barreira = 1;//a barreira comeca para baixo

    //sensor = 1;//inicializa o sensor a 1

}

//interrupcao externa 0

*void* External0\_ISR(*void*) interrupt 0{

    botao = 0; //assinala que o botao para entrar foi pressionado

    pressionado = 1;//foi pressionado

}

//interrupcao externa 1

*void* External1\_ISR(*void*) interrupt 2{

    botao = 1; //assinala que o botao para sair foi pressionado

    pressionado = 1;//foi presionado

}

//interrupcao tempo a cada segundo

*void* Timer0\_ISR(*void*) interrupt 1{

    contaBarreira++;

    conta ++;

}

*void* display(*void*){

*unsigned* *int* Segmentos = 0;

    if (~L0) {Segmentos += 1;}

    if (~L1) {Segmentos += 1;}

    if (~L2) {Segmentos += 1;}

    if (~L3) {Segmentos += 1;}

    if (~L4) {Segmentos += 1;}

    if (~L5) {Segmentos += 1;}

    if (~L6) {Segmentos += 1;}

    if (~L7) {Segmentos += 1;}

    P2 = Segmentos; //atualiza o display

}

*void* leSensor(){

    if (sensor == 0){

        bufferPassouCarro = 1; //se o sensor estiver a 0 entao o carro passou

    }

    if (bufferPassouCarro == 1 && sensor == 1){

            passouCarro = 1; //se o sensor estiver a 1 entao o carro passou todo

            bufferPassouCarro = 0; //reseta o buffer

    }

}

*void* moveBarreira(*void*){

    if (abreBarreira){

        referenciaBarreira = oitenta; //se o carro passou entao a barreira tem de ir para 80

        if (contaBarreira == referenciaBarreira){

            barreira = 0; //se ja chegou ao sitio certo entao fica igual

        }

        if (contaBarreira ==  fimTempoBarreira){

            contaBarreira = 0; //reseta o contador da barreira

            barreira = 1; //levanta a barreira

        }

    }else{

        referenciaBarreira = zero; //se o carro passou entao a barreira tem de ir para 0

        if (contaBarreira == referenciaBarreira){

            barreira = 0; //se ja chegou ao sitio certo entao fica igual

        }

        if (contaBarreira ==  fimTempoBarreira){

            contaBarreira = 0; //reseta o contador da barreira

            barreira = 1; //baixa a barreira

            //depois testar como e que se baixa a barreira

        }

    }

}

*void* main(*void*){

    //inicializacoes

    Init();

    while (1){ //loop inifinito

        display();

        leSensor();

        moveBarreira();

        if (segmentoA || segmentoB || segmentoC || segmentoD){

                ledVerde = 0; //ativa o led verde

                ledVermelho = 1; //desliga o led vermelho

        } else{

                ledVermelho = 0;//ativa o led vermelho

                ledVerde = 1; //desliga o led verde

                }

        if (conta == fimTempo){

            conta = 0; //volta a contar

            if (segmentoA || segmentoB || segmentoC || segmentoD || botao){

                if (pressionado){ //se tiver lugares ou se for para sair

                    contaSegundo ++; //passou 1 segundo

                    ledVerde = 0; //ativa o verde

                    ledVermelho = 1; //desliga o vermelho

                    ledAmarelo = ~ledAmarelo; //amarelo intermitente

                    abreBarreira = 1;//levanta a barreira

                    if (contaSegundo >= 10){//espera ate ser maior que 10 segundos

                        if (passouCarro){

                            abreBarreira = 0; //baixa a barreira

                            passouCarro = 0; //reseta o carro

                            pressionado = 0; //volta a colocar o botao de passou a 0

                            contaSegundo = 0; //reseta o timer

                            //sensor = 1; //reseta o sensor

                            ledAmarelo = 1; //desliga o led amareleo

                        }

                    }

                }

            }

        }

    }

}

Codigo em Assembly:

;Constantes

fimTempo            EQU 100; 100 \*50 \*200us = 1s

fimTempoBarreira    EQU 100; ; 100 \* 200us = 20ms

zeroBarreira        EQU 3; 3 \* 200us = 600us

oitentaBarreira     EQU 7; 7 \* 200us = 1400us

;Portas P0

L0 EQU P0.0;

L1 EQU P0.1;

L2 EQU P0.2;

L3 EQU P0.3;

L4 EQU P0.4;

L5 EQU P0.5;

L6 EQU P0.6;

L7 EQU P0.7;

;Portas P1

ledVerde   EQU P1.0;

ledVermelho EQU P1.1;

ledAmarelo EQU P1.2;

barreira   EQU P1.3;

sensor     EQU P1.4;

;Portas P2

segmentoA EQU P2.0;

segmentoB EQU P2.1;

segmentoC EQU P2.2;

segmentoD EQU P2.3;

;variaveis globais

pressionado         EQU 0x30;assinala que o botao foi pressionado

botao               EQU 0x31;assinala que o botao foi pressionado

conta               EQU 0x32;conta o tempo em 200us

contaSegundo        EQU 0x33;conta o tempo em segundos

contaBarreira       EQU 0x34;conta o tempo da barreira em 200us

abreBarreira        EQU 0x35;assinala que a barreira esta a abrir

passoucarro         EQU 0x36;assinala que o carro passou

referenciaBarreira  EQU 0x37;assinala a referencia da barreira

adjudaContador      EQU 0x39;assinala a ajuda do contador

bufferAbreBarreira  EQU 0x40;assinala que o carro passou e a barreira deve abrir

;inicio

cseg at 0

jmp main

;interrupcao esterna 0

cseg at 0x3

jmp External0\_ISR

;interrupcao esteterna 1

cseg at 0x13

jmp External1\_ISR

;Interropcao do timer0

cseg at 0Xb

jmp Timer0\_ISR

;inicializacoes

cseg at 0x50

Init:

    ;Configuracao Registo IE

    setb EA; ativa interrupcoes globais

    setb ET0;ativa interrupcao timer0

    setb EX0;ativa interrupcao externa 0

    setb EX1;ativa interrupcao externa 1

    ;Configuracao reisto TMOD

    anl TMOD, #0xF0 ;limpa os 4 bits do timer0

    orl TMOD, #0x02;modo 2 do timer 0

    ;Configuracao timer 0 - 200us

    mov TH0, #56; Timer 0 - 200

    mov TL0, #56;

    ;Configuracao registos TCON

    setb TR0; comeca o timer 0

    setb IT0; interrupcao ativa a falling edge

    ;inicializacao das variaveis

    clr pressionado;

    mov conta, #0;

    clr botao;

    clr passoucarro;

    clr adjudaContador;

    ;Configuracao dos pinos

    ;Porta 1

    clr ledVerde; led verde ligado

    setb ledVermelho; Led vermelho desligado

    setb ledAmarelo; Led amarelo desligado

    setb barreira; barreira para baixo

    mov contaBarreira, #0; contaBarreira a 0

    ;Porta 2

    mov P2, #0x00; todos os segmentos desligados

    setb segmentoA; segmento A desligado

    clr segmentoB; segmento B desligado

    clr segmentoC; segmento C desligado

    setb segmentoD; segmento D ligado

    ;

ret

;interrupcao esterna 0

External0\_ISR:

    CLR botao; assinala que o botao de sair foi pressionado

    setb pressionado;assinala que o botao foi pressionado

reti

;interrupcao esterna 1

External1\_ISR:

    setb botao; assinala que o botao fde entrar foi pressionado

    setb pressionado;assinala que o botao foi pressionado

reti

;Interrupcao timer 0

Timer0\_ISR:

    inc conta; incrementa a contagem dos 200us

    inc contaBarreira; incrementa a contagem da barreira

reti

cseg at 0xA0

main:

    mov sp,#7;inicializacao da stack

    call Init; chama a inicializacao

whileTrue:

    call display; chama a funcao display

    call leSensor; chama a funcao leSensor

    call moveBarreira; chama a funcao moveBarreira

    mov a, #fimTempo; coloca o valor de 250 no registo R0

    cjne a , conta, whileTrue; verifica se a contagem e menor que 250

    inc adjudaContador; incrementa o ajudaContador

    mov conta,#0 ; Limpa o conta

    mov r3, adjudaContador; coloca o valor do ajudaContador no registo R3

    cjne r3, #50, whileTrue; verifica se o ajudaContador e menor que 50

    mov adjudaContador, #0; ; limpa o ajudaContador

    mov conta, #0 ; limpa o conta

    ;aqui ja passou 1 segundo

    jb botao, ifPressionado;se o botao estiver on salta para o if

    setb ledVermelho; led vermelho ligado

    clr ledVerde; led verde ligado

    jb segmentoA, ifPressionado;se o a estiver on salta para o if

    jb segmentoB, ifPressionado;se o b estiver on salta para o if

    jb segmentoC, ifPressionado;se o c estiver on salta para o if

    jb segmentoD, ifPressionado;se o d estiver on salta para o if

    jmp elseNaoLugares;se nao houver nada ligado salta para o naoLugares

ifPressionado:

    jnb pressionado, whileTrue; se o botao nao estiver pressionado volta ao inicio

    inc contaSegundo; incrementa a contagem dos segundos

    clr ledVerde; led verde ligado

    setb ledVermelho; Led vermelho desligado

    cpl ledAmarelo; led amarelo intermitente

    setb abreBarreira; abre a barreira

    mov R0, contaSegundo; coloca o valor da contagem no registo R0

    cjne R0, #10, whileTrue; verifica se a contagem e menor que 10

    mov contaSegundo,#9; coloca 10 no R0 para nao dar erros de overflow

    jnb passoucarro, whileTrue; verifica se o carro passou

    clr abreBarreira; feixa a barreira

    clr passoucarro; da reset ao carro

    clr pressionado; da reset ao pressionado

    mov contaSegundo, #0; da reset a contagem dos segundos

    setb ledAmarelo; desliga o led amarelo

jmp whileTrue; volta ao inicio do ciclo

elseNaoLugares:

    clr ledVermelho; led vermelho ligado

    setb ledVerde; Led verde desligado

jmp whileTrue; volta ao inicio do ciclo

display:

    mov R0, #0x00; inicializa o registo R1

    jb L0, segmento2; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento2:

    jb L1, segmento3; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento3:

    jb L2, segmento4; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento4:

    jb L3, segmento5; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento5:

    jb L4, segmento6; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento6:

    jb L5, segmento7; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento7:

    jb L6, segmento8; verifica se esta a 0

    inc R0; soma o valor do registo R2

segmento8:

    jb L7, fimDisplay; verifica se esta a 0

    inc R0; soma o valor do registo R2

fimDisplay:

    MOV P2, R0; coloca o valor do registo R0 na porta 2

ret

leSensor:

    jb sensor, bufferFimSensor; verifica se o sensor esta a 0

    setb bufferAbreBarreira; assinala que o carro passou

bufferFimSensor:

    jnb bufferAbreBarreira, fimSensor; verifica se o carro passou

    jnb sensor, fimSensor; verifica se o sensor esta a 0

    setb passoucarro; assinala que o carro passou

    clr bufferAbreBarreira; limpa o buffer

fimSensor:

    ret

moveBarreira:

    jnb abreBarreira, elseBarreira; verifica se a barreira esta a abrir

    mov referenciaBarreira, #oitentaBarreira; coloca o valor de 80 na referencia da barreira

    mov a, contaBarreira; coloca o valor da contagem no registo A

    cjne a, referenciaBarreira, continuaMovimento; verifica se a contagem da barreira e menor que 80

    clr barreira; Barreira nao move

continuaMovimento:

    mov a, contaBarreira

    cjne a, #fimTempoBarreira, fimBarreira; verifica se a contagem da barreira e menor que 25ms

    mov contaBarreira, #0; limpa a contagem da barreira

    setb barreira; Barreira para cima

    jmp fimBarreira; salta para o fim da barreira

elseBarreira:

    mov referenciaBarreira, #zeroBarreira; coloca o valor de 0 na referencia da barreira

    mov a, contaBarreira ; coloca o valor da contagem no registo R0

    cjne a, referenciaBarreira, continuaMovimento2; verifica se a contagem da barreira e menor que 0

    clr barreira; Barreira nao move

    jmp fimBarreira; salta para o fim da barreira

continuaMovimento2:

    mov a, contaBarreira; coloca o valor da contagem no registo R0

    cjne a, #fimTempoBarreira, fimBarreira; verifica se a contagem da barreira e menor que 25ms

    mov contaBarreira, #0; limpa a contagem da barreira

    setb barreira; Barreira para baixo

    jmp fimBarreira; salta para o fim da barreira

fimBarreira:

    ret

END