ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO DE ÁGUEDA
UNIVERSIDADE DE AVEIRO

Relatório de Projecto Temático em Electrónica e Sistemas Digitais

Parque de estacionamento

André Costa 50207
Diogo Pombo 49538
Marcelo Coelho 50478
Nuno Baioneta 50250
Richard Ramos 46545

Professora orientadora:

Margarida Urbano

Águeda 2010/2011

Relatório de Projecto Temático em Electrónica e Sistemas Digitais

Parque de estacionamento

André Costa	Nº 50207
Diogo Pombo	Nº 49538
Marcelo Coelho	Nº 50478
Nuno Baioneta	Nº 50250
Richard Ramos	Nº 46545

Professora orientadora:

Margarida Urbano

Águeda 2010/2011

Índice

Índice de ilustrações

Introdução

O presente projecto insere-se no módulo temático de Electrónica e Sistemas Digitais, módulo que reúne as disciplinas de Microprocessadores e Microcontroladores e Sistemas Digitais. Foi pedido ao grupo que desenvolvesse um sistema electrónico que tivesse como função a gestão de um parque de estacionamento. O sistema deveria gerir as entradas e saídas dos utentes do parque, contabilizar o tempo que cada utente usufruiu do parque, calcular e efectuar o respectivo pagamento, levando assim à automatização do processo de gestão de parques de estacionamento.

Pretende-se explicar o funcionamento do projecto desenvolvido e dar a conhecer o percurso da sua concepção.

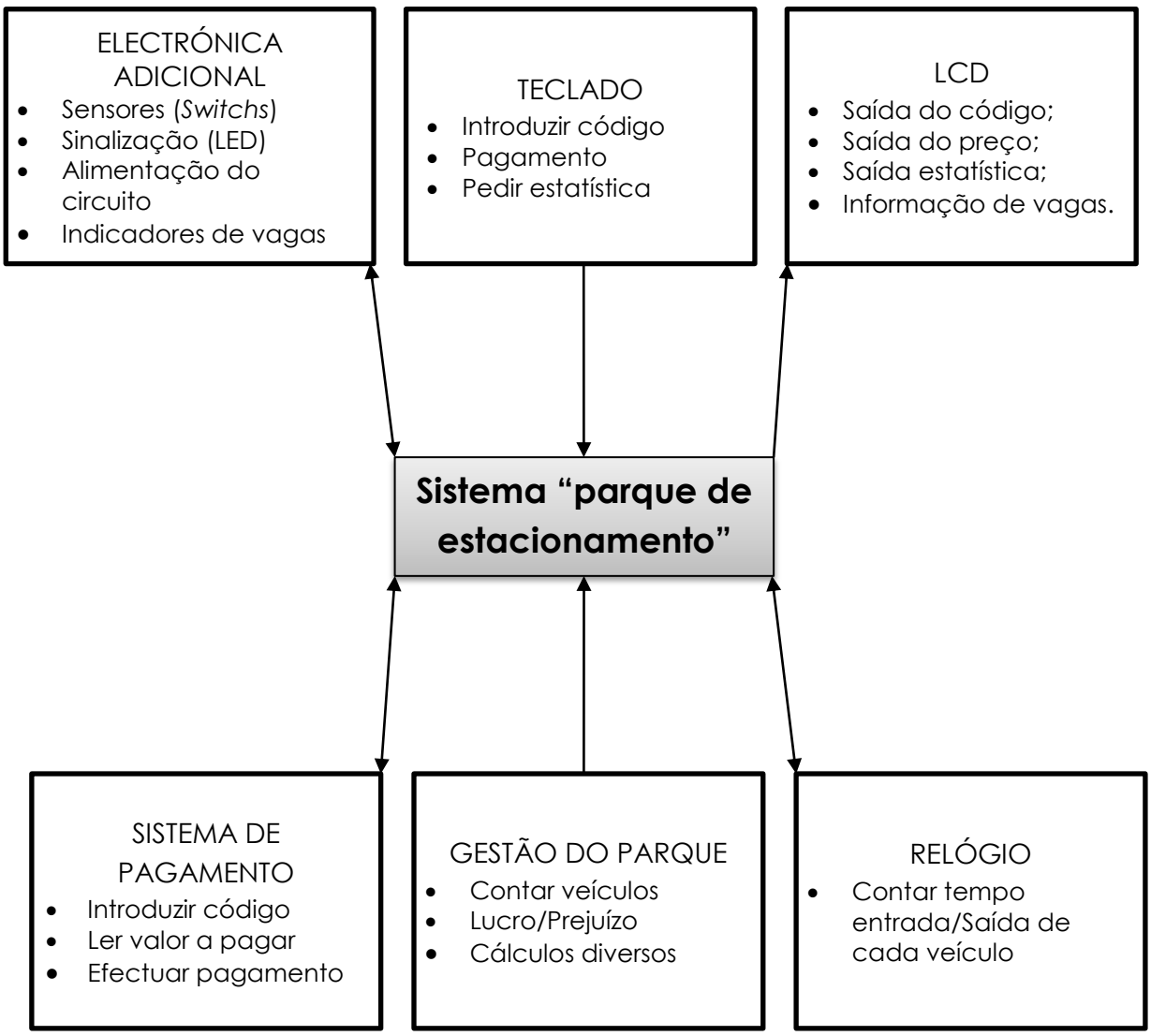
Objectivos

Inicialmente, definiram-se os seguintes objectivos para a realização do projecto, de acordo com o enunciado que se encontra em anexo:

- Construção de um sistema que permita contar o número de vagas de cada piso no parque;
- Implementação de um semáforo para a entrada do parque de modo a indicar se o parque está cheio ou se ainda tem vagas;
- Realização de um sistema que permita contar o tempo que cada condutor esteve dentro do parque;
- Construção de um sistema que permita o condutor à saída pagar uma taxa mediante o tempo que usufruiu do parque de estacionamento;
- Elaboração de uma função no microcontrolador que permita fazer a contagem do número de carros que entraram num dia.

Diagrama de blocos do sistema de gestão

Abaixo apresenta-se o diagrama do sistema de gestão efectuado, por forma a compreender melhor a função de cada elemento.

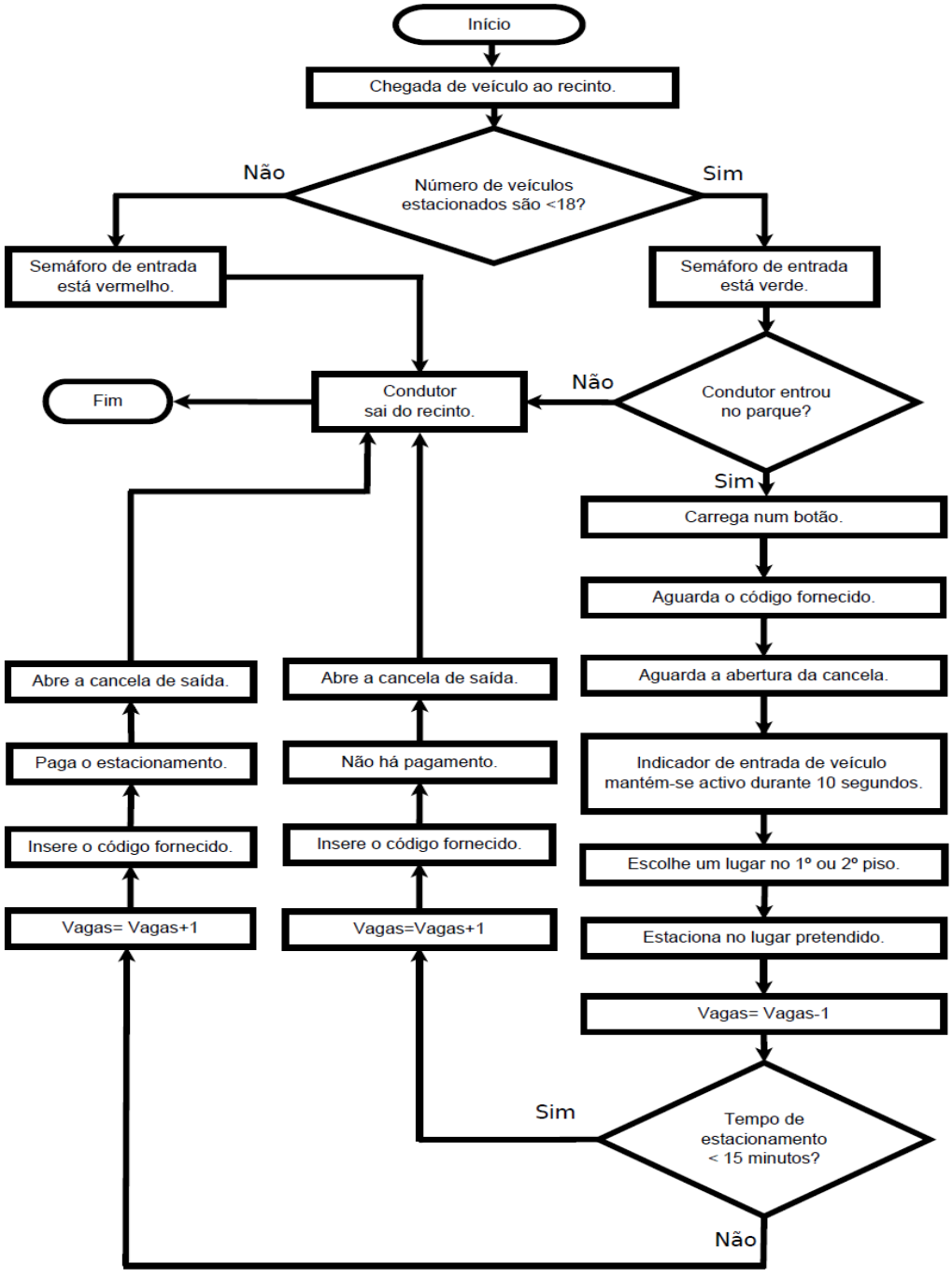


Descrição do diagrama de blocos

Através do diagrama de funcionamento pode-se observar que o sistema do parque de estacionamento é constituído por seis blocos que podem ser divididos pela sua função. Os conjuntos *Teclado* e *Gestão do parque* fornecem informação ao sistema pois a função do teclado é permitir a comunicação do utilizador com o sistema, inserir o código e efectuar o pagamento. A função do segundo grupo mencionado tem como objectivo fornecer dados como a contagem de carros dentro do parque. Os blocos "Electrónica adicional", "Sistema de pagamento" e "Relógio" tanto fornecem informação como a recebem, devendo-se ao facto de serem grupos mais amplos que contêm diferentes funções e diferentes partes ou estão dependentes da informação de outros conjuntos. Estes grupos serão abordados com mais detalhe nos capítulos seguintes. O conjunto LCD recebe informação proveniente do sistema para assim ser possível saber o código de entrada, a quantia a pagar e também ser possível ver o código e a quantia que o condutor inseriu. Na página a seguir apresenta-se o fluxograma do parque por forma a compreender melhor a interacção dos elementos de cada bloco.

Fluxograma do funcionamento do parque

O fluxograma apresentado a seguir descreve as etapas que o sistema de gestão do parque de estacionamento vai desempenhar, de acordo com os objectivos especificados anteriormente e ainda permitir obter uma melhor perspectiva do funcionamento do sistema desenvolvido neste projecto.



Controlador do sistema do parque

No sistema de gestão do parque de estacionamento, para ser possível efectuar todo processamento de dados provenientes do parque, utilizaram-se microcontroladores PIC *18F2520*. A razão do uso deste modelo deve-se à utilização do mesmo na disciplina associada de Microprocessadores e Microcontroladores.

Este modelo possui 28 pinos, dos quais apenas 22 pinos poderão ser utilizados como entradas/saídas (consultar anexo 3). Inicialmente apenas um microcontrolador poderia ser utilizado, porém o número de entradas e saídas necessárias para o funcionamento de todo projecto é superior a 22. O que obrigou à utilização de um segundo microcontrolador, que para além de fornecer mais entradas/saídas torna independentes certos sistemas. Em relação ao processo de programação do microcontrolador, após a conclusão de um programa, é necessário converter a linguagem de programação (linguagem C) em hexadecimal, linguagem máquina a utilizar no microcontrolador. Após a conversão em hexadecimal, é necessário um programa no computador para comunicar com o microcontrolador para enviar o programa já compilado.

Seguidamente, abordar-se-á o modo de comunicação de vários periféricos utilizados assim como o teclado e o LCD.

Teclado

O teclado é um dispositivo que permite estabelecer a interacção com o utilizador. O teclado utilizado é do tipo numérico pois torna mais simples algumas operações, como aceder às estatísticas, introduzir o código do veículo e simular o pagamento. A ilustração 1 mostra o teclado utilizado.

Ilustração 1: Teclado utilizado para a realização do projecto.

Este teclado dispõe de 7 pinos para conexão. O princípio de funcionamento do teclado baseia-se numa matriz constituída por quatro linhas e três colunas.

Após a abertura do teclado para verificar as suas ligações internas, desenhou-se o circuito que se segue abaixo:

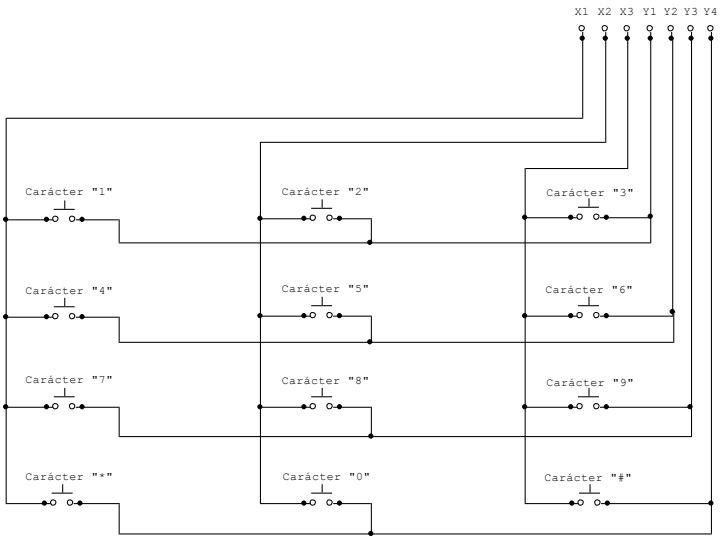


Ilustração 2: Circuito interno do teclado utilizado.

Assumindo que o teclado funciona como uma matriz de 3 colunas por 4 linhas, definiu-se que a letra X se refere ao número da coluna e a

letra Y ao número da linha do carácter. Por exemplo, o carácter '7' corresponde às coordenadas X=1 e Y=3.

Para que todas as teclas desempenhem as funções pretendidas devem-se conectar todos os pinos ao microcontrolador. Para o microcontrolador definiram-se como variáveis de entrada X_1, X_2 e X_3 e como variáveis de saída Y_1, Y_2, Y_3 e Y_4, estabelecendo estas inicialmente com o nível lógico *zero*. A cada variável de entrada foi conectada uma resistência de 10k Ω por forma a limitar a intensidade de corrente para o microcontrolador.

Na figura seguinte encontra-se um prolongamento dos pinos do teclado para verificar o método de conexão das resistências.

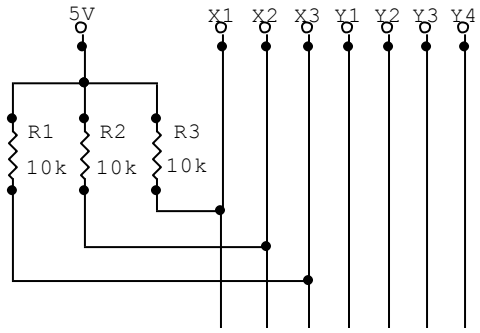


Ilustração 3: Método utilizado para proteger o circuito interno do teclado

Foi necessário implementar um código que pudesse informar ao microcontrolador o número premido. O teclado foi ligado ao porto *C* e definiu-se como entradas para o microcontrolador os pinos *RC0*, *RC1* e *RC2* enquanto que os pinos *RC3*, *RC4*, *RC5* e *RC6* foram definidos como saídas. As quatro saídas mencionadas anteriormente mudam de estado individualmente, isto é, quando a saída *RC3* está ao nível lógico zero, as restantes saídas têm de estar ao nível lógico um e, pouco tempo depois, as saídas *RC3, RC4, RC5* e *RC6* já se encontram no nível lógico um e a saída *RC4* no nível lógico zero e assim sucessivamente. A seguir apresenta-se o código implementado.

```
int ler_tecla()
{
	int num;
	do{
		num=20;
				do
				{
					RC0=0;RC1=1;RC2=1;RC3=1;
					if(RC4==RC0)num=1;
					if(RC5==RC0)num=2;
					if(RC6==RC0)num=3;
				}
				while(RC4!=1 || RC5!=1 || RC6!=1);

				do
				{
					RC0=1;RC1=0;RC2=1;RC3=1;
					if(RC4==RC1)num=4;
					if(RC5==RC1)num=5;
					if(RC6==RC1)num=6;
				}
				while(RC4!=1 || RC5!=1 || RC6!=1);

				do
				{
					RC0=1;RC1=1;RC2=0;RC3=1;
					if(RC4==RC2)num=7;
					if(RC5==RC2)num=8;
					if(RC6==RC2)num=9;
				}
				while(RC4!=1 || RC5!=1 || RC6!=1);

do
				{
					RC0=1;RC1=1;RC2=1;RC3=0;
					if(RC4==RC3)num=10;
					if(RC5==RC3)num=0;
					if(RC6==RC3)num=11;
				}
				while(RC4!=1 || RC5!=1 || RC6!=1);
	}
	while(num==20);

	return num;
}
/* Fim da função*/
```

No final, a função retornará o valor *num*, que é o número da tecla. Na página seguinte apresenta-se a conexão do teclado ao microcontrolador.

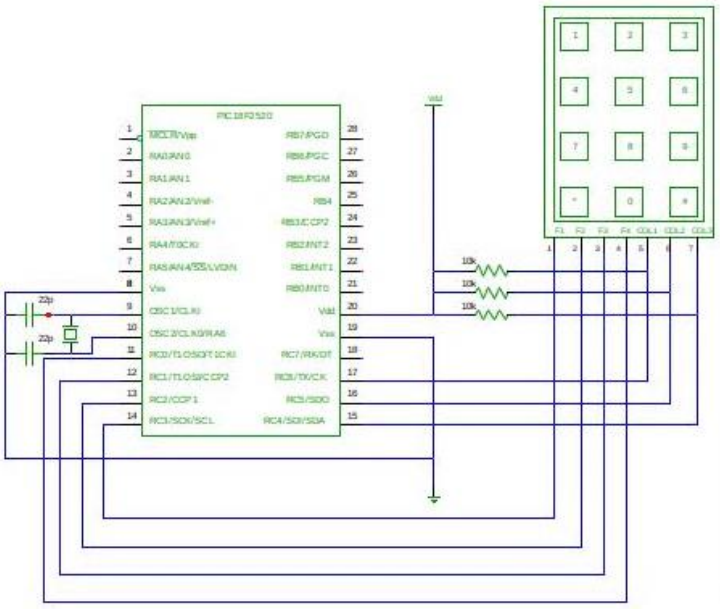


Ilustração 4: Conexão do teclado ao microcontrolador.

Nas páginas seguintes apresenta-se a comunicação com o LCD utilizado.

LCD

O LCD, do inglês *Liquid Cristal Display*, é um painel onde é possível visualizar mensagens enviadas pelo microcontrolador. Inicialmente definiu-se que este dispositivo teria de mostrar mensagens relacionadas com o fornecimento do código à entrada do parque, a indicação do preço à saída e a visualização das estatísticas. A ilustração abaixo mostra o tipo de *display* utilizado.

Ilustração 5: LCD utilizado para a realização do projecto.

O LCD utilizado é composto por duas linhas e vinte colunas, dispondo de dezasseis pinos. O *datasheet* deste dispositivo pode ser consultado no anexo 1.

Com o objectivo de estabelecer a comunicação deste dispositivo com o microcontrolador foi necessária uma rotina onde se configura o texto a aparecer em cada linha, utilizando a função *sprintf()*; a posição do LCD onde inicia o texto, utilizando a função lcd_*goto()* e a limpeza de cada linha utilizando a função *lcd_clear()*. Encontra-se em anexo 9 a rotina de configuração do LCD.

A rotina fornecida para a configuração deste dispositivo configura unicamente saídas que pertencessem ao porto *B*. Caso estas saídas fossem alteradas para outros portos o microcontrolador não conseguia estabelecer a comunicação com o LCD. Este *display* possui 16 pinos, dos quais oito servem exclusivamente para transferência de dados entre o microcontrolador e o mesmo. Para este projecto serão apenas utilizados quatro pinos do LCD que serão *DB4*, *DB5*, *DB6* e *DB7*, ficando assim a

operar no modo de quatro *bits*. Existem dois pinos que servem para alimentar a luz de fundo que neste modelo não se encontra disponível. Existe um terminal que serve para regular o contraste que ficará ligado à *massa*. Outro terminal serve para configurar o modo de funcionamento do *display*, se estiver ligado à *massa*, o *display* irá funcionar no modo de leitura e a receber dados, caso contrário funcionará no modo de escrita, enviando os dados. Para esta situação, o LCD funcionará sempre no modo de leitura. Outro terminal serve para seleccionar os registos. Este pino quando se encontra no nível lógico zero, uma instrução é escrita no LCD caso contrário é enviado um carácter para o LCD. O valor lógico desta variável é configurado durante a escrita do código. Na página seguinte apresenta-se a conexão do LCD ao microcontrolador.

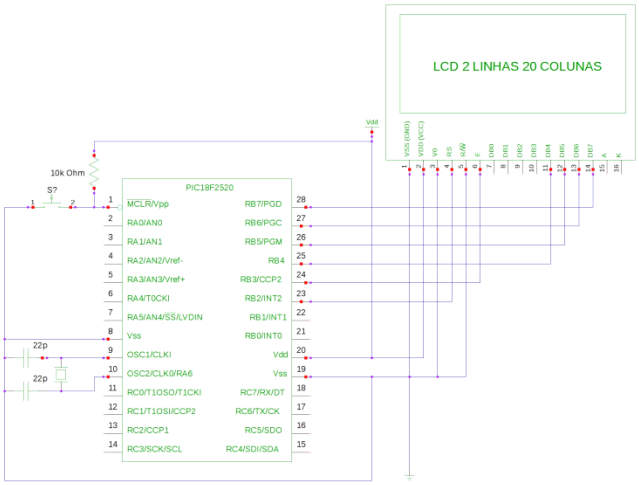


Ilustração 6: Conexão do LCD ao microcontrolador

As mensagens enviadas para o LCD não devem ter palavras com acentuação devido ao aparecimento de caracteres desconhecidos em vez do caracter pretendido.

Gestão do número de vagas

Devido a um número insuficiente de portas disponíveis inicialmente no microcontrolador, houve a necessidade de reduzir o número de pisos, dos três inicialmente definidos pelo enunciado para dois, e o número de lugares existentes por piso, de vinte para nove.

Uma vez que já estão em uso treze portas do microcontrolador pelo LCD e teclado, restam apenas oito para entradas ou saídas de dados. O sistema em análise foi implementado noutro microcontrolador do mesmo modelo, uma vez que para este sistema foram necessárias nove portas de transferência de dados.

Para o microcontrolador que se encarrega de estabelecer a comunicação com o LCD e teclado, das oito mencionadas anteriormente, serão utilizadas três neste sistema, sendo uma para controlo do LED amarelo que integra o semáforo da entrada do parque, outra porta para a simulação da barreira de entrada e outra porta para a simulação da barreira de saída. O sistema para o controlo destes indicadores será discutido em subcapítulos adiante.

Para cada piso utilizou-se um *display* de sete segmentos para indicar o número de vagas. O mesmo configurou-se ligando sete resistências de 270 Ω a cada segmento para limitar a corrente. Como os *displays* são de cátodo comum indica que todos os segmentos que os constituem têm um ponto comum que, neste caso, ligar-se-á à *massa*. Para que o microcontrolador estabelecesse o controlo de vagas de cada piso utilizaram-se dois descodificadores *HEF4511* para que convertesse as palavras de binário para utilização posterior em cada *display* de sete segmentos. Cada descodificador possui uma saída específica para cada segmento, tornando-se possível mostrar os dígitos de 0 a 9 utilizando apenas 4 saídas do microcontrolador. Encontra-se no anexo 4 o *datasheet* do descodificador utilizado. O motivo de ter utilizado este modelo de descodificador deve-se à existência do mesmo no armazém da escola. A seguir apresenta-se o circuito implementado para os dois pisos.

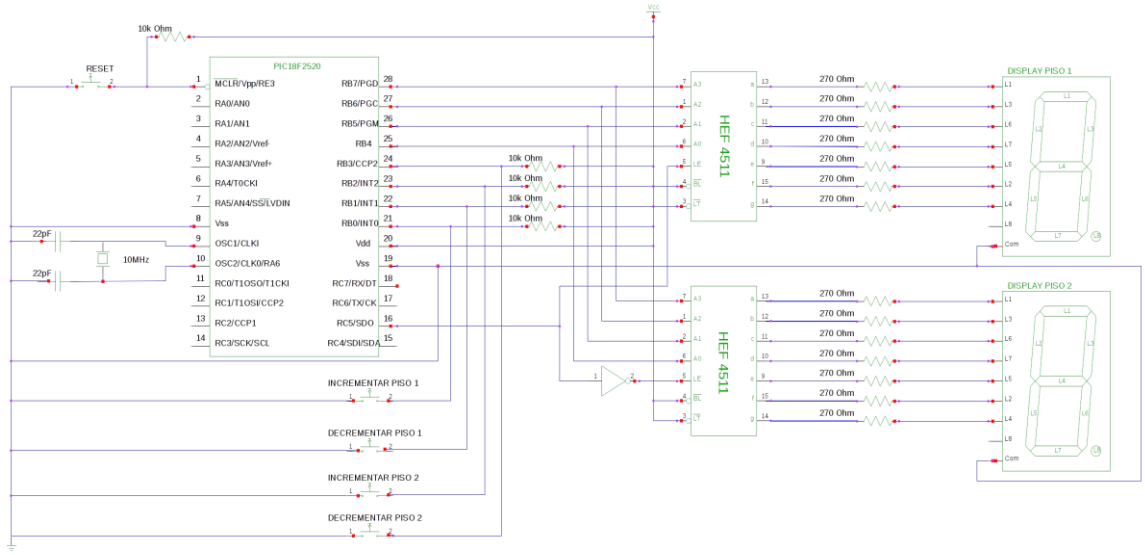


Ilustração 7: Circuito implementado para os dois pisos.

Os terminais assinalados *RB4*, *RB5*, *RB6* e *RB7* dizem respeito às portas do microcontrolador utilizadas para controlar as variáveis. A porta *RB4* irá controlar o *bit* menos significativo enquanto a porta *RB7* irá controlar o *bit* mais significativo. A porta *RC5* irá controlar os *enables* dos descodificadores variando o seu estado lógico. As portas *RA0*, *RA1*, *RB0* e *RB1* irão incrementar ou decrementar em cada piso conforme o caso. A seguir apresenta-se o código fonte desenvolvido.

```
#include <pic18.h> /*continua na próxima caixa de texto*/
```

Para cada número em decimal foi necessário convertê-lo para binário. Assim, criou-se uma função para cada número em decimal, cujo conteúdo é a configuração das saídas *RB4*, *RB5*, *RB6* e *RB7*.

```
void nove()  {RB4=1;RB5=0;RB6=0;RB7=1;}
void oito()  {RB4=0;RB5=0;RB6=0;RB7=1;}
void sete()  {RB4=1;RB5=1;RB6=1;RB7=0;}
void seis()  {RB4=0;RB5=1;RB6=1;RB7=0;}
void cinco() {RB4=1;RB5=0;RB6=1;RB7=0;}
void quatro(){RB4=0;RB5=0;RB6=1;RB7=0;}
void tres()  {RB4=1;RB5=1;RB6=0;RB7=0;}
void dois()  {RB4=0;RB5=1;RB6=0;RB7=0;}
void um()    {RB4=1;RB5=0;RB6=0;RB7=0;}
void zero()  {RB4=0;RB5=0;RB6=0;RB7=0;}

/*continua na próxima caixa de texto*/
```

Para cada piso construiu-se uma função para controlar o número de vagas. O descodificador só altera o número se e só se o seu *enable* estiver no nível lógico *zero*. De acordo com o *datasheet* deste dispositivo, quando o *enable* passa para o nível lógico *um* mantém o mesmo estado das saídas quando se encontrava no nível lógico *zero*. Assim, a porta *NOT* irá inverter o estado da porta *RC5*, sempre que esta última se controlar para o piso 1 enquanto a porta *RA5* controlará o piso 2. A seguir apresenta-se a função implementada para controlar o piso 1.

```
void display_1(int x) /*Função para o piso 1*/
{
      RC5=1;
      switch(x)
      {
            case 0 :zero();   break;
            case 1 :um();     break;
            case 2 :dois();   break;
            case 3 :tres();   break;
            case 4 :quatro(); break;
            case 5 :cinco();  break;
            case 6 :seis();   break;
            case 7 :sete();   break;
            case 8 :oito();   break;
            case 9 :nove();   break;
      }
}

/*continua na próxima caixa de texto*/
```

Para o piso 2 implementou-se uma função designada *display_2*. O conteúdo da função será idêntico, à excepção do estado da porta *RC5* que passará a ter o *zero* lógico.

Utilizaram-se algumas variáveis que desempenhavam funções importantes: *atraso*, que servia para aumentar ou diminuir o tempo de resposta do programa quando se carrega em algum botão; *en1* que servia para controlar o *enable* no piso 1; *en2* para controlar o *enable* no piso 2; *v1* para armazenar o número de vagas no piso 1; *v2* para armazenar o número de vagas no piso 2; *r1*, *r2*, *r3* e *r4* para assegurar que não irá haver alteração de números se o botão estiver premido por muito tempo.

A seguir apresenta-se o conteúdo da função *main()*, onde as funções dos números representados em binário serão aqui invocadas.

Quando o programa inicia, o número que aparecerá em cada *display* será o número 9.

```
void main()
{
      unsigned long int atraso,en1,en2;
      int v1,v2,r1,r2,r3,r4;
      TRISA=0b00000011;
      TRISB=0b11000011;
      TRISC=0b00000000;
      ADCON1=0b00001111; //Todas as saídas são digitais
      RB0=1; //Botão para incrementar no piso 1
      RB1=1; //Botão para decrementar no piso 1
      RB2=1; //Botão para incrementar no piso 2
      RB3=1; //Botão para decrementar no piso 2
      v1=9;
      v2=9;
      en1=0; en2=0;
      r1=0; r2=0; r3=0; r4=0;

/*continua na próxima caixa de texto*/
```

Atendendo ao excerto de código que se encontra na página seguinte verifica-se que, quando é premido o botão para decrementar e se o número for igual ou menor que zero a variável *num* terá de manter o valor zero, caso contrário, se o valor for superior a zero, terá de fazer *num=num-1*. Se o botão não for premido, o programa terá de manter o valor presente da variável *num*. Quando o botão para incrementar o número de vagas é premido pode ocorrer uma das seguintes situações: antes do botão ser premido, o número de vagas pode ser nove; antes do botão ser premido, o número de vagas pode ser menor do que nove. Caso ocorra a primeira situação, o programa terá de manter o valor nove. Se a segunda situação ocorrer, o programa terá fazer *num=num+1*. Todo o código desenvolvido para este microcontrolador encontra-se no anexo 11.

```
while(1)          //Ciclo infinito
{
            for(atraso=0;atraso<60000;atraso++){}     //Cria um atraso
            if(r1==0)
            {
                  if(RB0==0)
                  {
                        if(v1==9 || v1>9) v1=9;
                        else v1++;
                  }
                  else v1=v1;
            }
            else v1=v1;
            if(r2==0)
            {
                  if(RB1==0)
                  {
                        if(v1==0 || v1<0) v1=0;
                        else v1--;
                  }
                  else v1=v1;
            }
            else v1=v1;
            if(r3==0)
            {
                  if(RB2==0)
                  {
                        if(v2==9 || v2>9) v2=9;
                        else v2++;
                  }
                  else v2=v2;
            }
            else v2=v2;
            if(r4==0)
            {
                  if(RB3==0)
                  {
                        if(v2==0 || v2<0) v2=0;
                        else v2--;
                  }
                  else v2=v2;
            }
            else v2=v2;

/*continua na próxima caixa de texto*/
```

Os ciclos *for* que se seguem irão controlar o *enable* de cada descodificador. Assim, as funções implementadas para cada piso serão aqui invocadas.

O programa realizado impede que se o botão estiver premido após algum tempo, o contador incremente ou decremente mais do que um número.

```
            for(en1=0;en1<1000;en1++){} display_1(v1);

            for(en2=0;en2<1000;en2++){} display_2(v2);

            if(RB0==1) r1=0;
            else r1=1;

            if(RB1==1) r2=0;
            else r2=1;

            if(RB2==1) r3=0;
            else r3=1;

            if(RB3==1) r4=0;
            else r4=1;
      }     /*Fim do ciclo infinito*/
}/*Fim do programa*/
```

A seguir abordar-se-á o modo como foi implementado o semáforo para a entrada do parque.

Semáforos

De acordo com o enunciado do projecto, à entrada do parque deverá existir um semáforo que será composto por três indicadores: vermelho, para informar aos condutores de que não existem vagas em nenhum piso; amarelo, para informar ao condutor de que irá entrar um veículo no parque e verde, para indicar que existem vagas no parque.

Os indicadores vermelho e verde foram ambos implementados num circuito lógico. As variáveis de entrada são as mesmas que controlam os segmentos de cada *display*. O indicador amarelo foi implementado junto das barreiras que será discutido no capítulo referente às barreiras.

Quando ambos os displays mostram o número *zero* existe um único segmento em cada um deles que se encontra apagado que é designado pelo segmento *g*. Para tal, utilizou-se uma porta *NOT* à saída de cada descodificador onde pudesse controlar esta entrada. Assim, o nível lógico posterior será *um*. Por forma a estabelecer uma condição com os restantes segmentos activos, utilizaram-se portas *AND* para o restante circuito lógico. A seguir, apresenta-se o circuito implementado para activar a luz vermelha e verde, dependendo do estado do parque.

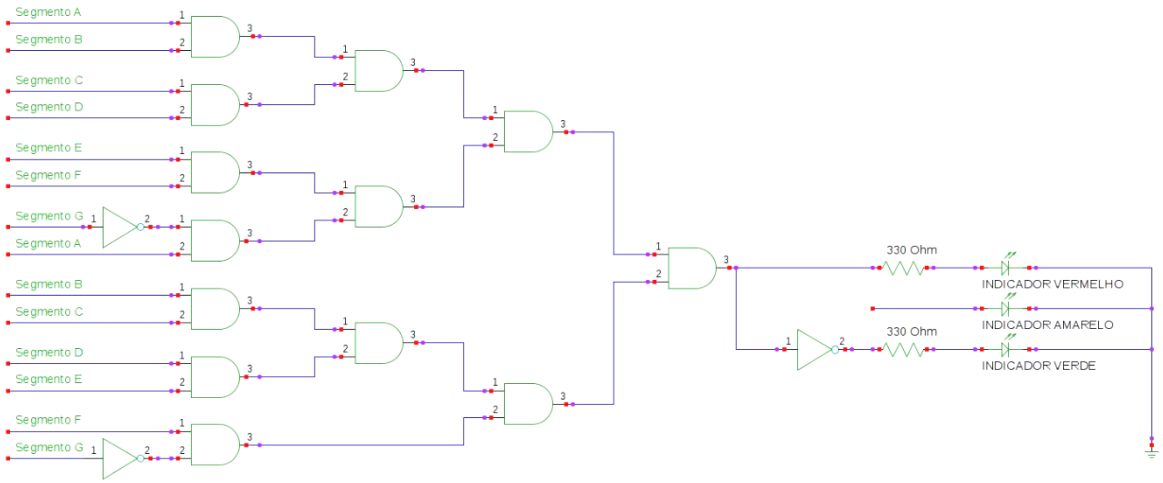


Ilustração 8: Circuito final para os indicadores verde e vermelho.

O motivo de ter utilizado só portas *AND* de duas entradas deve-se ao facto de que este era o único modelo de portas lógicas para este efeito no armazém da escola.

Como o indicador vermelho só está activo nas condições descritas anteriormente, o indicador verde nunca ficará activo nestas condições. Assim, sempre que o indicador vermelho esteja no nível lógico *zero*, o indicador verde ficará activo, colocando antes uma porta *NOT* no mesmo poto onde se encontra o indicador vermelho. A seguir apresentam-se duas ilustrações onde fazem referência a dois casos possíveis, onde mostra os diferentes estados de cada indicador.

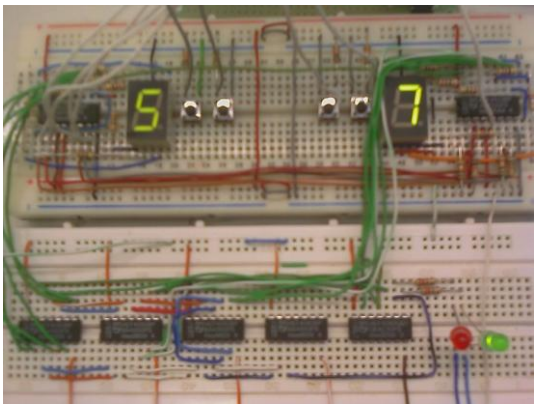


Ilustração 9: Teste do indicador verde.

Por observação da ilustração anterior, num piso existem cinco vagas e noutro existem sete. Assim, o indicador verde ficará activo. A ilustração seguinte mostra o caso onde não existe alguma vaga.

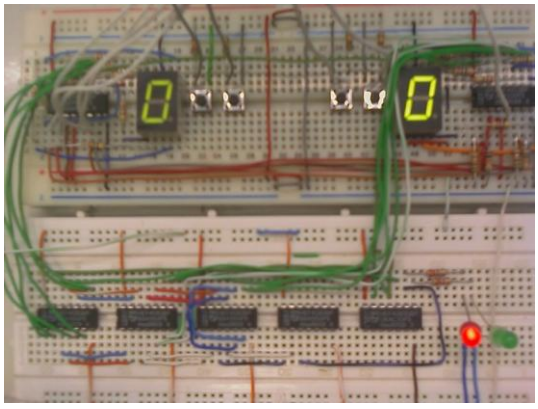


Ilustração 10: Teste do indicador vermelho.

Verifica-se assim que o indicador vermelho ficará activo.

Nas páginas seguintes explicar-se-á o modo de atribuição do código.

Atribuição de código a cada veículo

Quando o programa inicia, a mensagem que aparece no *LCD* é a indicada na ilustração abaixo.

A função criada anteriormente para ler o teclado será invocada, aguardando a opção do utilizador. A seguir apresenta-se parte do conteúdo da função *main()* para mostrar a mensagem no LCD e aguardar a escolha do utilizador.

```
void main()
{

      int a,d;
      LCD_clear();
      DelayMs(50);
      sprintf(str,"Para entrar --> [#]");
      LCD_goto(0);
      LCD_puts(str);
      sprintf(str,"Para sair --> [*]");
      LCD_goto(40);
      LCD_puts(str);
      d=0;
      a=ler_tecla();
/*continua na caixa de texto da página seguinte*/
```

Definiu-se que o código a atribuir era um número inteiro composto por cinco números. Para que seja fornecido um código ao utilizador, terá de ser premido o botão de cardinal. Caso seja premido o botão de asterisco, pede ao utilizador que insira o seu código fornecido à entrada. A seguir apresenta-se uma ilustração onde mostra no LCD um código a atribuir.

Ilustração 11: Formato do código fornecido.

Na ilustração anterior encontra-se um número inteiro após o código indicado que inicialmente servia para indicar quantas posições de uma determinada matriz que se encarregava de guardar os instantes de tempo estavam ocupadas. Para a fase final, este número não aparecerá.

O modo de introdução e validação do código será discutido noutro capítulo. Caso seja premida outra tecla, a mensagem terá de se manter. Para agrupar as instruções em relação às opções mencionadas anteriormente, foi necessário elaborar um *switch case* conforme mostra o próximo código, posterior ao da função do teclado.

```
switch(a){
case 11:
lcd_clear();DelayMs(50);
if(x>19){
      lcd_clear();
      DelayMs(50);
      sprintf(str,"Nao ha vagas");
      lcd_goto(0);
      lcd_puts(str);
      for(f=0;f<700000;f++){} break;
}
if(car[x]==0)//procura os zeros no vector
            {
                  lcd_clear();
                  DelayMs(50);
                  car[x]=segundos;
                  sprintf(str,"O codigo:  %ld",car[x]);
                  lcd_goto(0);
                  lcd_puts(str);
                  sprintf(str,"OK -> #");
                  lcd_goto(40);
                  lcd_puts(str);
                  sprintf(str,"%d",x);
                  lcd_goto(58);
                  lcd_puts(str);
            }
            e=ler_tecla();
            if(e!=10 && e!=11)
            {
                  while(e!=10 && e!=11)
                  {
                        e=ler_tecla();}
                  }
                  if(e==11)
                  {
                        x++;
                        break;
                  }
                  if(e==10)
                  {
                        a=11;
                        car[x]=0;
                  }

                  break;
```

As instruções foram implementadas por forma a construir uma matriz linha que só tivesse dezoito células. Quando o programa inicia, a matriz referida antes do excerto de código da página anterior é só constituída por zeros. Quando entram sucessivos carros, esta matriz encarrega-se de guardar o instante de tempo em segundos em que o veículo entrou. Se, por algum motivo, a matriz já se encontrar com todas as suas posições diferentes de zero, no LCD terá de aparecer uma mensagem a indicar que não há vagas. Assim se justifica o motivo de ter colocado uma instrução que verificasse esta condição em primeiro lugar. Suponha-se agora que após o programa ter iniciado, só entrou um veículo. Após armazenados os valores de tempo e posição, o programa terá de estar preparado para receber um próximo pedido, adicionando uma unidade à variável que controlará a posição do vector. Se, por qualquer motivo, o utilizador se enganar, pode voltar ao menu principal, recorrendo ao botão de asterisco e o programa assegura que não guardará o código fornecido, mantendo a posição a zero, como mostra a última instrução. Na página seguinte discutir-se-á o modo de validação de código à saída do parque.

Validação de código à saída do parque

Conforme foi visto no subcapítulo anterior, a mensagem que aparece no LCD após iniciar o programa informa o utilizador para premir o botão de cardinal para entrar ou asterisco para sair. Abordar-se-á neste subcapítulo o método de validação do código introduzido.

Após o utilizador premir o botão de asterisco, o programa irá pedir o código fornecido à entrada. Conforme indicado no subcapítulo anterior, o código é composto por cinco números. O programa irá ler um número de cada vez até chegar à quinta posição de um vector através de um ciclo. Definiu-se ainda que, enquanto o utilizador estivesse a digitar o código este apareceria primeiro em asteriscos e depois que fosse digitado o último número, nas mesmas posições, apareceria o número digitado para que o utilizador se certificasse de teria introduzido correctamente o código. Caso o utilizador se enganasse na introdução, o programa permitia limpar o espaço e pedir para digitar o código novamente, quantas vezes fossem necessárias. Abaixo apresenta-se o código fonte implementado, tendo em conta que se integra dentro do *switch case* abordado no capítulo anterior.

```
case 10:
      do
      {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Inserir codigo:");
            lcd_goto(0);
            lcd_puts(str);
            for(q=0;q<5;q++)
            {
                  w=ler_tecla();
                  if(w==10 || w==11)
                  {
                        while(w==10 || w==11)
                        {
                              w=ler_tecla();
                        }
                  }
                  z[q]=w; DelayMs(15);
                  sprintf(str,"*"); lcd_cursor(15+q);
                  lcd_puts(str);
            }
            /* continua na caixa de texto da página seguinte*/
```

```
            lcd_clear(); DelayMs(50);
            sprintf(str,"Inserir codigo:");
            lcd_goto(0); lcd_puts(str); DelayMs(50);
            sprintf(codigo,"%d%d%d%d%d",z[0],z[1],z[2],z[3],z[4]);
            lcd_cursor(15);lcd_puts(codigo);
            e=ler_tecla();
            while(e!=11 && e!=10)
            {
                  e=ler_tecla();
            }
      } while(d!=1); break;
```

Uma vez que um dia é composto por 86400 segundos, o contador de segundos terá de iniciar em 10000 e terminar em 96400 devido ao número de caracteres para os códigos dos veículos. Se fosse introduzido um código inferior a 10000 ou superior a 96400, o programa teria de estar preparado para indicar ao utilizador que o código seria inválido, uma vez que estes códigos também nunca são gerados. O excerto de código que se apresenta a seguir, tem como função comparar com a matriz de códigos atribuídos o valor introduzido.

```
for(t=0;t<18;t++)
{
      if(car[t]!=g){h++;}     //g é o código introduzido
}
if(h==18)
{
      lcd_clear();
      DelayMs(50);
      sprintf(str,"Codigo invalido...");
      lcd_goto(0);
      lcd_puts(str);
      for(f=0;f<999999;f++){}
      break;
}
```

O ciclo de repetição implementado incrementará caso o código de cada posição seja diferente. Assim que a variável auxiliar atingir o valor dezoito, no LCD terá de aparecer uma menagem a alertar o utilizador que o código é inválido.

Implementação do sistema de pagamento

O sistema de pagamento foi implementado em euros, uma vez que surgiram dificuldades de conversões para euros e cêntimos em simultâneo.

Por forma a tornar um preço apelativo aos utilizadores, definiu-se que o preço de estacionamento seria de 0,0006 cêntimos por segundo, o que representa aproximadamente 2 € por hora. O pagamento só será feito com números inteiros. A expressão para o cálculo do pagamento é o produto da constante indicada acima pela diferença dos instantes de tempo.

Definiu-se ainda que, se o utilizador mantivesse o seu veículo por menos de vinte e oito minutos ficaria isento de pagamento, caso contrário a taxa entraria em vigor. O motivo de ter escolhido o intervalo de tempo indicado deve-se ao facto de, enquanto o resultado determinado for menor do que um, o sistema consideraria que o utilizador não teria de pagar. Após a validação do código, no LCD irá aparecer uma mensagem a indicar o preço, como indica a ilustração abaixo.

Ilustração 12: Indicação do valor a pagar.

Partindo do exemplo acima, o valor a pagar é de dezasseis euros. Caso o utilizador introduza a menos, a mensagem que aparecerá terá o formato conforme mostra a ilustração da página seguinte. Para este caso, o indicador vermelho terá de se manter activo.

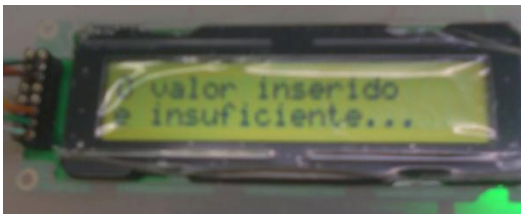


Ilustração 13: Alerta de valor insuficiente.

Após algum tempo de mostrar a mensagem indicada na ilustração anterior, esta muda novamente para outra que confirme o preço a pagar, quantas vezes necessárias.

A partir do momento que o utilizador insira um valor inferior ao pedido pelo sistema, o valor a pagar posteriormente não irá aumentar.

Implementação do sistema de contagem de tempo

O sistema de contagem de tempo baseia-se num contador do microcontrolador que está configurado para contar o tempo real. Quando um condutor entra no parque e prime o botão de entrada, o instante em que é premido o botão é guardado juntamente com o código do veículo. Quando o condutor sai e vai pagar a taxa de estacionamento, ao inserir o código do veículo o sistema guarda o tempo actual e subtrai com o tempo de entrada guardado inicialmente de modo a obter o intervalo de tempo em que o condutor esteve dentro do parque e assim poder obter o valor que o condutor tem de pagar.

Inicialmente, o relógio foi implementado tendo os seguintes campos: *número de dias*, *horas*, *minutos* e *segundos*. Surgiram dificuldades em efectuar operações neste formato por forma a converter posteriormente para um valor a pagar. Assim, o formato de contagem do tempo foi constituído só por segundos, tornando assim mais fáceis as operações a efectuar. O temporizador utilizado para efectuar a contagem do tempo foi o *timer0*, que interrompe o programa num curto intervalo de tempo para incrementar a variável *segundos*. Abaixo segue-se o código implementado para o relógio.

```
long int segundos=10000;
void interrupt cont(void)
{
      segundos++;
      if((TMR0IE==1)&&(TMR0IF==1))
      {
            TMR0IF=0;
            TMR0H=0xB3;
            TMR0L=0xCF;
      }
}
```

A variável *segundos* foi declarada como uma variável global porque posteriormente será invocada na função de interrupção *cont*. O motivo do tipo de variável ser inteiro longo deve-se ao facto de que este tipo pode representar um número superior a 65535, uma vez que um dia é composto por 86400 segundos. Na página seguinte apresenta-se o modo de simulação das barreiras.

Simulação das barreiras de entrada e saída

Para tornar a entrada e saída dos veículos um processo mais controlado, decidiu-se implementar barreiras que impeçam os veículos entrar sem premir o botão de entrada e não pagar à saída. Como se trata de um projecto de sistemas digitais, não existe a necessidade de construir barreiras mecânicas mas sim simular o levantar e o descer das cancelas, através de dois LED, um vermelho que indica que está fechado e um verde que indica que está aberto. O facto de usar dois microcontroladores no projecto permitiu fazer o controlo das cancelas através dos mesmos, por intermédio de programação, o que evitou o uso de mais electrónica adicional para o comando das cancelas. Quando o condutor prime o botão na entrada a cancela abre, ou seja, o LED vermelho (RA5) desliga e o indicador verde (RA2) liga. No momento em que isto ocorre, o indicador amarelo também está activo. No LCD aparece uma mensagem a indicar que está um veículo a entrar, conforme mostra a figura abaixo.

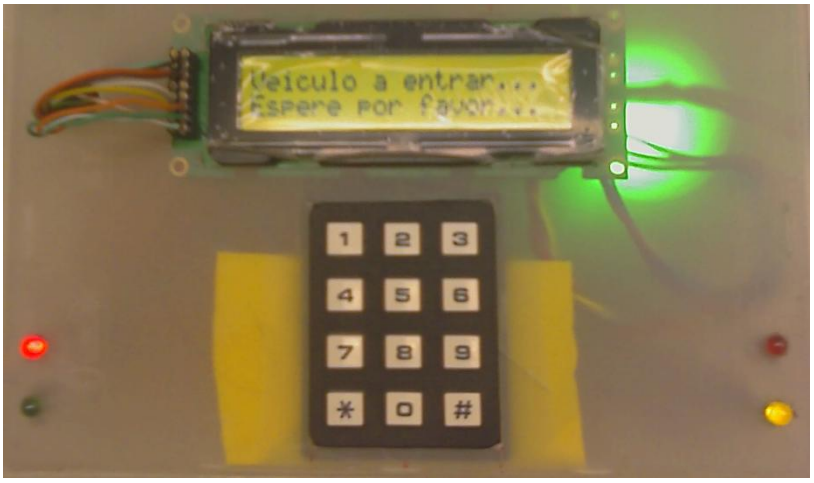


Ilustração 14: Simulação da cancela à entrada.

Passado aproximadamente um segundo, os indicadores mudam de estado. No programa foi colocado um ciclo *for* para fornecer o tempo que o veículo necessita para passar pela cancela. Ambos os LED trocam de estado para indicar que a cancela está a descer. O mesmo acontece na saída, inicialmente o LED vermelho (RA3) está ligado e o LED verde

(RA1) está desligado. Quando o condutor efectua o pagamento, ambos os LED que se encontram no canto inferior esquerdo da figura acima mudam de estado e, ao mesmo tempo, aparece uma mensagem no LCD a informar que o veículo está a sair. Passado aproximadamente um segundo os indicadores voltam ao estado inicial.

Realização de estatísticas

De acordo com o enunciado do projecto, pretende-se que ao final de cada dia se faça uma estatística do modo de ocupação do parque. Definiu-se que as estatísticas seriam o número de veículos que ocuparam o parque e o número de veículos que estão a ocupar o parque num determinado momento.

Definiu-se ainda que o modo de acesso seria através do modo de inserção do código para sair do parque. No campo para inserção da palavra-passe dever-se-ia introduzir *00000* e confirmar posteriormente.

A todo o código desenvolvido para o sistema de gestão adicionaram-se mais duas variáveis do tipo numérico: *cont*, para mostrar o número de saídas num dia e *carros* para determinar o número de veículos estacionados num determinado momento. Estas variáveis, quando o programa inicia, encontram-se a *zero*. A variável *cont* teria duas hipóteses de incrementar isto é, aumentava uma unidade caso o utilizador pagasse a quantia certa ou introduzisse um valor superior. Assim que o utilizador receba o seu código a variável *carros* incrementa uma unidade. À saída a variável *carros* decrementa se e só se o pagamento for suficiente. Este valor terá de ser a soma dos dois números que aparecem nos displays de sete segmentos. A seguir apresenta-se uma ilustração que exemplifica o formato da estatística após confirmação.

Ilustração 15: Formato das estatísticas.

As mensagens mudam após algum tempo e regressam ao menu principal sem que o administrador o pretenda fazer.

Se, em algum caso, o administrador aceder às estatísticas e apenas um veículo ocupou o parque, no LCD terá de aparecer "*Esteve 1 carro no parque*" e não "*Estiveram 1 carros no parque*".

Conclusões

Em suma e para concluir este projecto, apresentam-se as conclusões mais relevantes que se retiraram do mesmo. Este trabalho não correu de todo como o esperado, isto é, houve muitos problemas na compreensão do modo de funcionamento de vários dispositivos, nomeadamente, o teclado e o LCD o que originou atrasos significativos em todo o projecto. Felizmente na recta final do mesmo conseguiu-se recuperar algum tempo perdido.

O sistema que controla o parque de estacionamento apresenta alguns problemas, um deles é o facto de o parque não funcionar por mais de um dia. Tentou-se solucionar este problema com o reinício do sistema a uma determinada hora mas em vão, pois surgiu um erro ao compilar o código ao qual não se conseguiu dar resposta. Outro problema que se encontra neste trabalho é no sistema de pagamento que poderia ser implementado simultaneamente em euros e cêntimos, mas tornou-se complicado contornar este assunto devido ao separador das unidades monetárias. Já na parte de estatística tentou-se implementar um sistema que indicasse o dinheiro obtido durante um dia. Estes problemas mais concretos foram, de certa forma, objectivos que ficaram pelo caminho.

O sistema contador de vagas foi implementado de modo a simular um sensor de passagem. O que acontece para este caso é o facto de se mais uma roda accionar o botão, irá incrementar ou decrementar conforme o caso.

Uma vez que foi utilizado outro microcontrolador, não haveria limitação do número de pisos e vagas inicialmente propostos pelo enunciado pelo que esta decisão foi tomada num período de tempo que não o permitiria realizar.

O sistema reinicia se houver uma falha de energia, o que se tornaria um melhoramento possível no futuro juntamente com os outros problemas mencionados.

Anexos

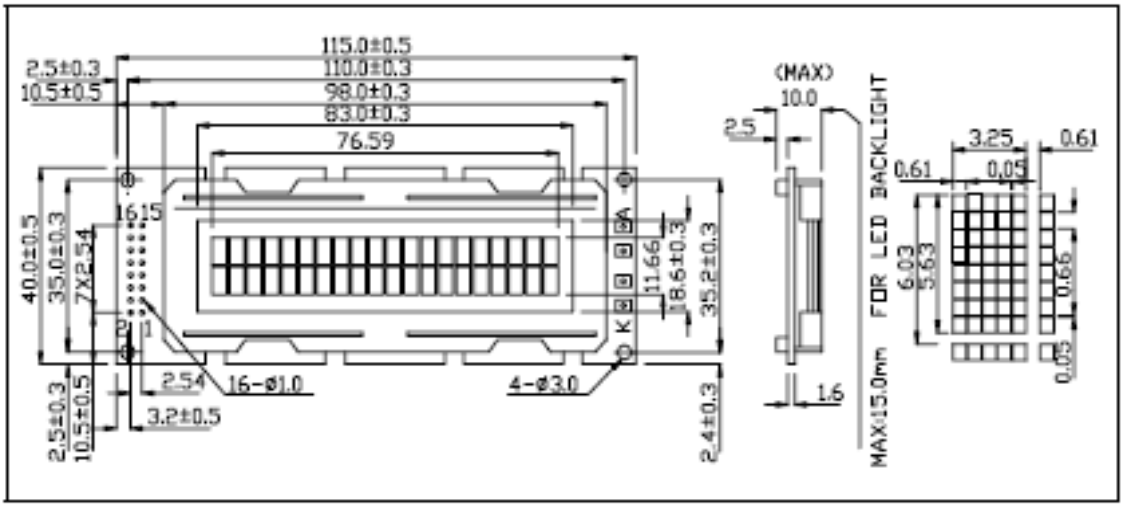
- Anexo 1: *Datasheet* do LCD utilizado;
- Anexo 2: *Datasheet* do teclado utilizado;
- Anexo 3: *Pinout* do microcontrolador utilizado;
- Anexo 4: *Datasheet* do descodificador utilizado;
- Anexo 5: *Datasheet* do circuito integrado composto por portas *AND*;
- Anexo 6: *Datasheet* do circuito integrado composto por portas *NOT*;
- Anexo 7: Circuito final;
- Anexo 8: Enunciado do projecto;
- Anexo 9: Rotina utilizada para o LCD;
- Anexo 10: Código fonte desenvolvido para controlo de LCD e teclado;
- Anexo 11: Código fonte desenvolvido para controlo dos *displays* de sete segmentos;
- Anexo 12: Orçamento do projecto;
- Anexo 13: Mapa de Gantt.

Anexo 1: *Datasheet* do *LCD* utilizado

JM202A

20 CHAR× 2LINE
1/ 16 DUTY,1/5 BIAS

1 EXTERNAL DIMENSION



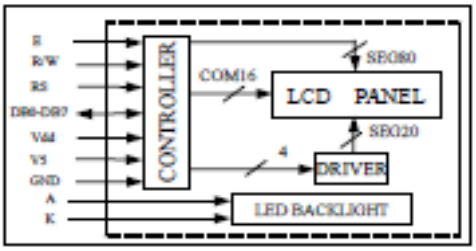
2 MECHANICAL DATA

ITEM	SPECIFICATION	UNIT
Module Size (W×H×T)	115.0×40.0× 10.0(LED:15.0)	mm
Viewing Area (W×H)	83.0×18.6	mm
Character Font(W×H)	5×7+cursor	dots
Character Size (W×H)	3.25×5.63	mm
Character Pitch(W×H)	3.86×6.03	mm
Dot Size(W×H)	0.61×0.66	mm

3 PIN CONNECTIONS

PIN	SYMBOL	SIGNAL DESCRIPTION
1	GND	Power Supply: 0V
2	Vdd	Power Supply: +5V
3	V5	Power Supply for LCD
4	RS	Register Select(H=DATA,L=Instruction)
5	R/W	Read/Write L=MPU to LCM,H=LCM to MPU
6	E	Enable
7 to 14	DB0 to DB7	Data Bus for 4 bit or 8 bit Mode
15	A	Anode of LED Unit
16	K	Cathode of LED Unit

4 BLOCK DIAGRAM



5 ABSOLUTE MAXIMUM RATINGS

PARAMETER	SYMBOL	MIN.	MAX.	UNIT
Supply Voltage	Vdd	-0.3	7	V
LCD Supply Voltage	V5	Vdd-15	Vdd+0.3	V
Input Voltage	Vi	-0.3	Vdd+0.3	V
Operating Temperature	Top	0/-20/-30	50/70/80	℃
Storage Temperature	Tstg	-20/-30/-40	70/80/80	℃

6 ELECTRICAL CHARACTERISTICS(Ta=25℃)

ITEM	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply Voltage(logic)	Vdd-GND	-	4.5	5.0	5.5	V
Supply Current(logic)	Idd	Vdd=5.0V	-	2.0	-	mA
Driving Current(LCD)	Icc		-	0.7	-	mA
Driving Voltage(LCD)	Vdd-V5		3.8	4.5	4.9	V
Input Voltage "H"	Vih		2.2	-	Vdd	V
Input Voltage "L"	Vil		-0.3	-	0.6	V
Output Voltage "H"	Voh	Ioh=-0.205mA	2.4	-	-	V
Output Voltage "L"	Vol	Iol=1.2mA	-	-	0.4	V

7 BOTTOM BACKLIGHT CHARACTERS(Ta=25℃)

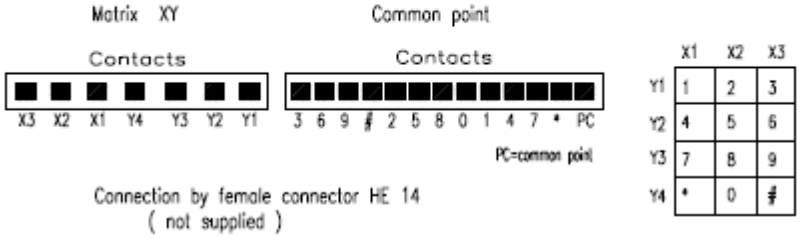
PARAMETER	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	V_{LED}	-	-	4.1	-	V
LED Forward Consumption Current	If	Ta=25℃ Vf=4.1V	-	115	-	mA
LED Allowable Dissipation	Pd	-	-	480	-	mW

Anexo 2: *Datasheet* do teclado numérico utilizado

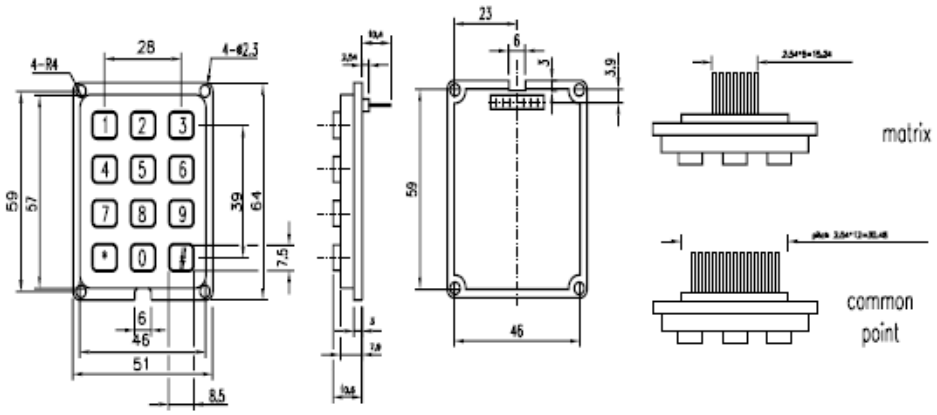
1 keypad
page ECO.1

12 KEYS – CONNECTOR

(rear view)



12 – BUTTON KEYPAD



electrical characteristics

contact power rating
nominal: 0,5 W

contact resistance
< 200 Ω

current
nominal: 20 mA,
minimum: 10 mA

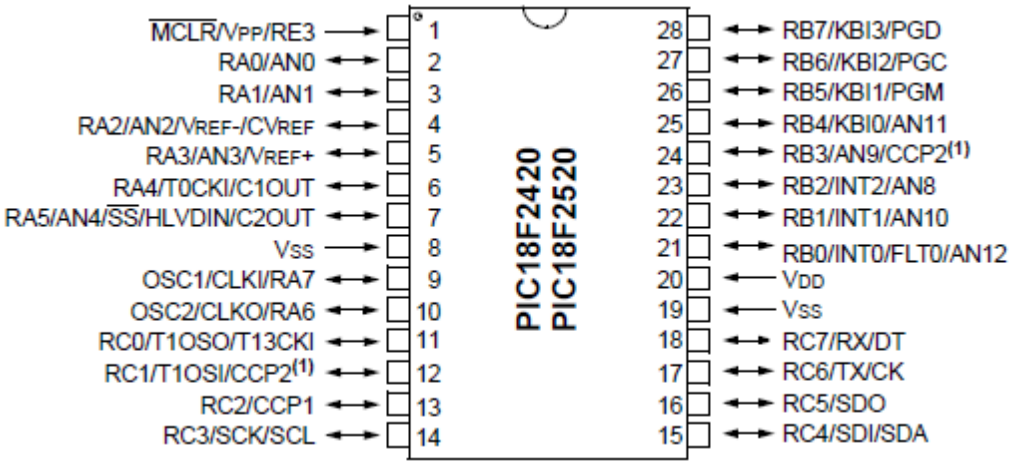
ESD protection
5000 V

insulation resistance
at 100 Vcc: > 1000 MΩ

life time
at nominal rating: > 10^6 operations

voltage
nominal: 24 V,
maximum: 24 V,
minimum: 500 mV

Anexo 3: *Pinout* do microcontrolador utilizado



Anexo 4: *Datasheet* do descodificador utilizado

Philips Semiconductors — **Product specification**

BCD to 7-segment latch/decoder/driver — HEF4511B MSI

DESCRIPTION

The HEF4511B is a BCD to 7-segment latch/decoder/driver with four address inputs (D_A to D_D), an active LOW latch enable input ($\overline{EL}$), an active LOW ripple blanking input ($\overline{BI}$), an active LOW lamp test input ($\overline{LT}$), and seven active HIGH n-p-n bipolar transistor segment outputs (O_a to O_g).

When $\overline{EL}$ is LOW, the state of the segment outputs (O_a to O_g) is determined by the data on D_A to D_D.
When $\overline{EL}$ goes HIGH, the last data present on D_A to D_D are stored in the latches and the segment outputs remain stable. When $\overline{LT}$ is LOW, all the segment outputs are HIGH independent of all other input conditions. With $\overline{LT}$ HIGH, a LOW on $\overline{BI}$ forces all segment outputs LOW. The inputs $\overline{LT}$ and $\overline{BI}$ do not affect the latch circuit.

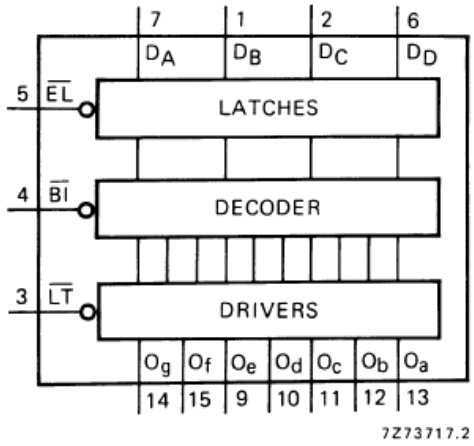


Fig.1 Functional diagram.

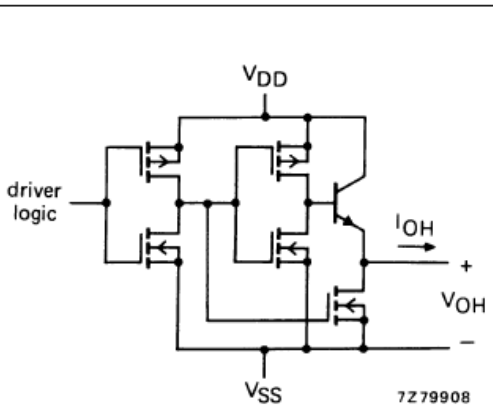


Fig.3 Schematic diagram of output stage.

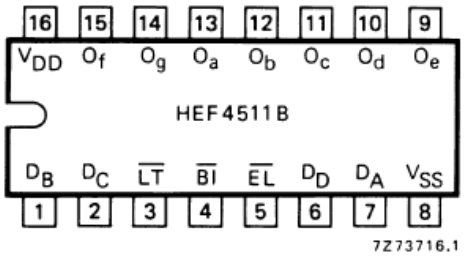


Fig.2 Pinning diagram.

HEF4511BP(N): 16-lead DIL; plastic (SOT38-1)
HEF4511BD(F): 16-lead DIL; ceramic (cerdip) (SOT74)
HEF4511BT(D): 16-lead SO; plastic (SOT109-1)
( ): Package Designator North America

PINNING

D_A to D_D	address (data) inputs
$\overline{EL}$	latch enable input (active LOW)
$\overline{BI}$	ripple blanking input (active LOW)
$\overline{LT}$	lamp test input (active LOW)
O_a to O_g	segment outputs

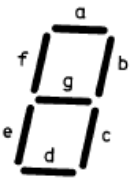


Fig.4 Segment designation.

FAMILY DATA, I_{DD} LIMITS category MSI

See Family Specifications

Anexo 5: *Datasheet* do circuito integrado composto por portas *AND*

Philips Semiconductors **Product specification**

Quadruple 2-input AND gate — HEF4081B gates

DESCRIPTION

The HEF4081B provides the positive quadruple 2-input AND function. The outputs are fully buffered for highest noise immunity and pattern insensitivity of output impedance.

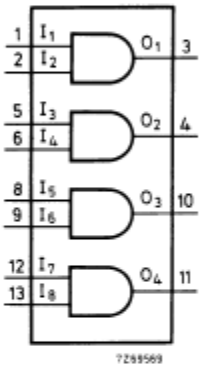


Fig.1 Functional diagram.

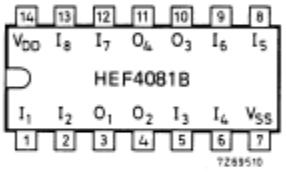


Fig.2 Pinning diagram.

HEF4081BP(N): 14-lead DIL; plastic (SOT27-1)
HEF4081BD(F): 14-lead DIL; ceramic (cerdip) (SOT73)
HEF4081BT(D): 14-lead SO; plastic (SOT108-1)
( ): Package Designator North America

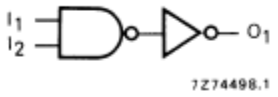


Fig.3 Logic diagram (one gate).

FAMILY DATA, I_{DD} LIMITS category GATES

See Family Specifications

Anexo 6: *Datasheet* do circuito integrado composto por portas *NOT*

5. Functional diagram

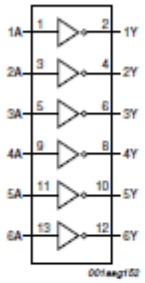


Fig 1. Functional diagram

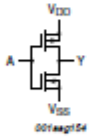


Fig 2. Schematic diagram (one inverter)

6. Pinning information

6.1 Pinning

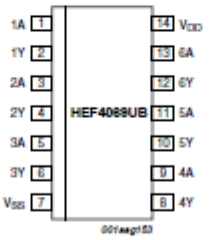


Fig 3. Pin configuration

6.2 Pin description

Table 2. Pin description

Symbol	Pin	Description
1A to 6A	1, 3, 5, 9, 11, 13	input
1Y to 6Y	2, 4, 6, 8, 10, 12	output
V_{SS}	7	ground (0 V)
V_{DD}	14	supply voltage

Anexo 7: Circuito final

Anexo 8: Enunciado do projecto

universidade de aveiro
escola superior de tecnologia e gestão de águeda

tema: electrónica e sistemas digitais 2010/11

projecto III:- Parque de estacionamento

Objectivo: Construir um sistema electrónico que permita fazer a gestão de um parque de estacionamento.

Descrição Sumária:

Pretende-se com este projecto implementar um sistema electrónico que simule o funcionamento de um parque de estacionamento composto por 3 pisos. Cada piso tem uma capacidade de 20 veículos. Existe à entrada do parque um semáforo de três cores (vermelho, verde e amarelo) e uma cancela que só se levantará quando for premido o botão de entrada. Quando entrar um carro terá de aparecer um nº identificativo do carro que está a entrar. O semáforo amarelo indica que existe um carro a entrar no parque. Em cada um dos pisos existe a indicação do nº de lugares livres. À saída do parque, terá de ser introduzido o nº de carro. A indicação de quanto se tem de pagar terá de aparecer num LCD. Pretende-se que no final de cada dia se possa fazer uma estatística da ocupação deste parque de estacionamento.

O grupo responsável por este projecto deverá:

- projectar o sistema;
- especificar os componentes a utilizar;
- construir e testar um protótipo do sistema.

Orientadores:

Margarida Urbano

Anexo 9: Rotina utilizada para o LCD

```
#include <pic18.h>
#include <stdio.h>

#define LCD_STROBE()          RB3=1;  asm("nop");  asm("nop");
RB3=0
#define LCD_cursor(x)         write_LCD(((x)&0x7F)|0x80,0)
#define LCD_clear()           write_LCD(0x1,0)
#define LCD_putch(x)          write_LCD(x,1)
#define LCD_goto(x)           write_LCD(0x80+(x),0);
#define LCD_cursor_right()    write_LCD(0x14,0)
#define LCD_cursor_left()     write_LCD(0x10,0)
#define LCD_display_shift()   write_LCD(0x1C,0)
#define LCD_home()            write_LCD(0x2,0)
#define LCD_cursor_on()       write_LCD(14,0)
#define LCD_blink_on()        write_LCD(13,0)
#define LCD_cursor_off()      write_LCD(12,0)
#define LCD_backspace()       write_LCD(5,0)

char str[20];
/*Esta é a string responsável por colocar o texto nas duas
linhas.*/

void DelayUs(unsigned char x)
{
     while(--x != 0) continue;
}
void DelayMs(unsigned char cnt)
{
     unsigned char i;
     do{
          i = 6;
          do
{
               DelayUs(250);
          } while(--i);
       } while(--cnt);
}
void write_LCD (unsigned char c, unsigned char CMD_DATA)
/* Função responsável por enviar dados para o LCD*/
{
     DelayUs(100);
     if(CMD_DATA==1) RB2=1;
     RB4=0; RB5=0; RB6=0; RB7=0;
     if(c & 0b00010000) RB4=1;
     if(c & 0b00100000) RB5=1;
     if(c & 0b01000000) RB6=1;
     if(c & 0b10000000) RB7=1;
     LCD_STROBE();
     RB4=0; RB5=0; RB6=0; RB7=0;
     if(c & 0b00000001) RB4=1;
     if(c & 0b00000010) RB5=1;
     if(c & 0b00000100) RB6=1;
     if(c & 0b00001000) RB7=1;
     LCD_STROBE();
     RB2=0; /* Selector de registos do LCD*/
```

```
}
void LCD_puts(const char *s)
/* Função responsável por enviar uma string de carácteres
para o LCD*/
{
     while(*s)
     write_LCD(*s++,1);
}
void LCD_init()     //Inicia o LCD
{
     TRISB2=0;      //Selector de registos
     TRISB3=0;      //Enable
     TRISB4=0;      //Data4
     TRISB5=0;      //Data5
     TRISB6=0;      //Data6
     TRISB7=0;      //Data7
     RB2=0;    //Selector de registos
     RB3=0;    //Enable
     RB4=1;    //Escrever 0x3
     RB5=1;
     RB6=0;
     RB7=0;
     LCD_STROBE();
     DelayMs(5);
     RB4=0; RB5=0; RB6=0; RB7=0;
     LCD_STROBE();
     DelayMs(5);
     write_LCD(0x28,0);
     write_LCD(0xF,0);
//Activa   o   display,   activa   o   cursor,   cursor   fica
intermitente
     write_LCD(0x1,0);  //Limpeza do display
     write_LCD(0x6,0);  //Modo de entrada
     write_LCD(0x80,0); //Inicia a zero o endereço DDRAM.
}
void main(void)
{
     LCD_init();
     DelayMs(5);
     LCD_cursor_off();
     while(1)
     {
/*Escreve na primeira linha do LCD*/
          sprintf(str,"Linha 1");
          LCD_goto(0);
          LCD_puts(str);
/*Escreve na segunda linha do LCD*/
          sprintf(str,"Linha 2");
          LCD_goto(40);
          LCD_puts(str);

     }
} /*Fim*/
```

Anexo 10 : Código fonte desenvolvido para controlo de LCD e teclado

```
# include <pic18.h>
/*Rotina do LCD também tem de estar incluída*/
int ler_tecla() /*Função para ler teclas*/
{
      int num;
      do{
            num=20;
                        do
                        {
                           RC0=0;RC1=1;RC2=1;RC3=1;
                           if(RC4==RC0)num=1;
                           if(RC5==RC0)num=2;
                           if(RC6==RC0)num=3;
                        }
                        while(RC4!=1 || RC5!=1 || RC6!=1);
                        do
                        {
                              RC0=1;RC1=0;RC2=1;RC3=1;
                              if(RC4==RC1)num=4;
                              if(RC5==RC1)num=5;
                              if(RC6==RC1)num=6;
                        }
                        while(RC4!=1 || RC5!=1 || RC6!=1);
                        do
                        {
                           RC0=1;RC1=1;RC2=0;RC3=1;
                           if(RC4==RC2)num=7;
                           if(RC5==RC2)num=8;
                           if(RC6==RC2)num=9;
                        }
                        while(RC4!=1 || RC5!=1 || RC6!=1);
                        do
                        {
                           RC0=1;RC1=1;RC2=1;RC3=0;
                           if(RC4==RC3)num=10;
                           if(RC5==RC3)num=0;
                           if(RC6==RC3)num=11;
                        }
                        while(RC4!=1 || RC5!=1 || RC6!=1);
      }
      while(num==20);
      return num;
}
int segundos, minutos, horas, dias;
long int segundos=10000;
void interrupt cont(void)  //Função da interrupção
{
      if((TMR0IE==1)&&(TMR0IF==1))
      {     //Quando a interrupção é causada pelo timer0
            segundos++;
            //Incrementa a cada meio segundo
      TMR0IF=0;               //Repor a flag a 0
            TMR0H=0xB3;           //Define o início do contador
            TMR0L=0xCF;           //   "        "          "
      }
}
//*********************************************************//
//         -->   PROGRAMA CORRE A PARTIR DAQUI    <--    //
```

```
//*********************************************************//
void main(){
/*
a -> ler tecla para o switch
b -> ler tecla para validação
c ->  cálculos de troco
*/
int a,x,q,w,e,d,b,t,n,v,i,h,cont;
// x -> número da matriz para atribuição do código
long int u,f,z,y,s,r,g,j,p,c,car[18],k[20],carro;
// q ->    apenas para for() contagens
// w -> ler tecla validações
// e -> ler tecla validações
// d -> para sair do case
// k[20] -> para cálculos
ADCON1=0xff;
// t -> for para descobrir carro correspondente ao codigo inserido
TRISB0=0;
// u ->    Cálculo para o tempo que a viatura está no parque
TRISB1=1;
// f -> for e variável para igualar aos segundos sempre que o
Cálculo de 'u' é efectuado
TRISB=0b00000000;
// z ->    variável cálculo para ler tecla
TRISA=0b00010000;
// y -> variável cálculo para ler tecla
TRISC=0b01110000;
// s -> variável cálculo para ler tecla
// r -> variável cálculo para ler tecla
lcd_init();     //1. inicia lcd
// g -> variável cálculo para ler tecla
DelayMs(5);
//2. atrasa
// j -> variavel que imprime o valor lido no pagamento
lcd_cursor_off();
//3. desliga o cursor
// p -> variavel de cálculo para o valor a pagar
// car[18] -> Matriz dos codigos de entrada
GIE=1;
// global interrupt enable
IPEN=1;
//Configurar as interrupções
TMR0IE=1;//  "                          "
TMR0IP=1;
// Alta prioridade
TMR0IF=0;
//Flag inicialmente a 0
TMR0H=0x67;
//Define o início do contador
TMR0L=0x96;     //  "              "                "
T0CON=0b10000111;     //configurar o timer0
a=0; x=0;  q=0;  w=0;  e=0;  d=0;  f=0;  z=0;  y=0;  s=0;
r=0;  g=0;  p=0;  j=0;  b=0;  carro=0; RA0=0; RA1=1; RA2=0;
RA3=0; RA4=0; RA5=1;
while(1)//***************** CICLO INFINITO***************
      {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Para entrar --> [#]");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"Para sair --> [*]");
```

```
            lcd_goto(40);
            lcd_puts(str);
            d=0;
            a=ler_tecla();
            switch(a)
            {
                  Case 10://****SAIR **************************
                        do
                              {
                                    lcd_clear();
                                    DelayMs(50);
                                    sprintf(str,"Inserir codigo:");
                                    lcd_goto(0);
                                    lcd_puts(str);
                                    for(q=0;q<5;q++)
                                    {
                                          w=ler_tecla();
                                          if(w==10 || w==11)
//Se tecla igual a apagar ou confirmar espera por ler tecla de
novo
                                          {
                                                while(w==10 || w==11)
                                          {w=ler_tecla();}
                                          }
                                          if(q==0){z=w;}
                                          if(q==1){y=z*10+w;}
                                          if(q==2){s=y*10+w;}
                                          if(q==3){r=s*10+w;}
                                          if(q==4){g=r*10+w;}
                                          sprintf(str,"*");
                                          lcd_cursor(15+q);
                                          lcd_puts(str);
                                          DelayMs(50);
                                    }
                                    lcd_clear();
                                    DelayMs(50);
                                    sprintf(str,"Inserir codigo:");
                                    lcd_goto(0);
                                    lcd_puts(str);
                                    sprintf(str,"%ld",g);
                                    lcd_goto(15);
                                    lcd_puts(str);
                                    sprintf(str,"ok[#]");
                                    lcd_goto(55);
                                    lcd_puts(str);
                                    sprintf(str,"[*]apagar");
                                    lcd_goto(40);
                                    lcd_puts(str);
                                    e=ler_tecla();
                                    while(e!=11 && e!=10)
{e=ler_tecla();}
/*enquanto tecla diferente de apagar e confirmar espera por ler
tecla de novo*/
                                    if(e==11)// se confirmar
                                    {
                                          if(g==0000)
                                          {
                                                if(cont==1)
                                                {
                                    lcd_clear();
                                    DelayMs(50);
                                    sprintf(str,"Esteve %d carro",cont);
```

```
                lcd_goto(0);
                lcd_puts(str);
                sprintf(str,"no parque...");
                lcd_goto(40);
                lcd_puts(str);
                for(f=0;f<999999;f++){}
                lcd_clear();
                DelayMs(50);
                sprintf(str,"Esta %ld carro",carro);
                lcd_goto(0);
                lcd_puts(str);
                sprintf(str,"no parque...");
                lcd_goto(40);
                lcd_puts(str);
                                }
                                else
                                {
lcd_clear();
DelayMs(50);
sprintf(str,"Estiveram %d carros",cont);
lcd_goto(0);
lcd_puts(str);
sprintf(str,"no parque...");
lcd_goto(40);
lcd_puts(str);
for(f=0;f<999999;f++){}
lcd_clear();
DelayMs(50);
sprintf(str,"Estao %ld carros",carro);
lcd_goto(0);
lcd_puts(str);
sprintf(str,"no parque...");
lcd_goto(40);
lcd_puts(str);
                                }

                        for(f=0;f<999999;f++){}
                        e=11;
                        d=1;
                        break;
                        }
                        if(g<10000 || g>96400)
                        {
lcd_clear();
DelayMs(50);
sprintf(str,"Codigo invalido...");
lcd_goto(0);
lcd_puts(str);
for(f=0;f<999999;f++){}
e=11;
d=1;
break;
                        }
                        else
                        {
                                f=segundos;
                                u=f-g;
                        }
                        for(t=0;t<18;t++)
                        {
                                if(car[t]==g)
                                {
```

```
                                        n=t;
                                    }
                                }
                                    h=0;
                                    for(t=0;t<18;t++)
                                    {
                                        if(car[t]!=g){h++;}
                                    }
                                    if(h==18)
{
lcd_clear();
                            DelayMs(50);
                            sprintf(str,"Codigo invalido...");
                            lcd_goto(0);
                            lcd_puts(str);
                            for(f=0;f<999999;f++){}
                            e=11; d=1; a=10; break;
                                    }
                                    do
{
                            if(u<1667)// se estiver há pouco tempo
                            {
                            lcd_clear();
                            DelayMs(50);
                            sprintf(str,"Nao precisa pagar...");
                            lcd_goto(0);
                            lcd_puts(str);
                            sprintf(str,"Boa viagem...");
                            lcd_goto(40);
                            lcd_puts(str);
                            RA5=0; RA2=1;
                            car[n]=0;
                            cont++; carro--;
                            for(f=0;f<999999;f++){}
                            RA5=1; RA2=0; e=11; d=1; break;
                            }
                            if(u>1667)
//se estiver há tempo suficiente para pagar
                            {
                                p=u*0.006;
                                lcd_clear();
                                DelayMs(50);
                                sprintf(str,"Valor a pagar:");
                                lcd_goto(0);
                                lcd_puts(str);
                                if(p==1)
                                {
                                    sprintf(str,"%ld euro",p);
                                    lcd_goto(40);
                                    lcd_puts(str);
                                    sprintf(str,"ok[#]");
                                    lcd_goto(55);
                                    lcd_puts(str);
                                }
                                else
                                {
                                sprintf(str,"%ld euros",p);
                                lcd_goto(40);
                                lcd_puts(str);
                                sprintf(str,"ok[#]");
                                lcd_goto(55);
                                lcd_puts(str);
```

```
                }
        }
b=ler_tecla();
while(b!=11){b=ler_tecla();}
if(b==11)
{
      lcd_clear();
      DelayMs(50);
      for(q=0;q<10;q++)
      {
            sprintf(str,"Pagar:");
            lcd_goto(0);
            lcd_puts(str);
            w=ler_tecla();
            sprintf(str,"ok[#]");
            lcd_goto(55);
            lcd_puts(str);
            sprintf(str,"[*]apagar");
            lcd_goto(40);
            lcd_puts(str);
            if(w==10)// se apagar
            {
                  q=0;
                  lcd_clear();
                  DelayMs(50);
                  sprintf(str,"Pagar:");
                  lcd_goto(0);
                  lcd_puts(str);
                  w=ler_tecla();
                  sprintf(str,"ok[#]");
                  lcd_goto(55);
                  lcd_puts(str);
                  sprintf(str,"[*]apagar");
                  lcd_goto(40);
                  lcd_puts(str);
                  }
            if(w==11)// se confirmar
            {
      lcd_clear();
      DelayMs(50);
      sprintf(str,"Pagar:");
      lcd_goto(0);
      lcd_puts(str);
      sprintf(str,"%ld euros",k[q-1]);
      lcd_goto(6);
      lcd_puts(str);
      if(k[q-1]==p)
      {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Obrigado...");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"Boa viagem...");
            lcd_goto(40);
            lcd_puts(str);
            RA5=0;
            RA2=1;
            for(f=0;f<999999;f++){}
            RA5=1;
            RA2=0;
            cont++;
```

```
                carro--;
                car[n]=0;  e=11; d=1;
            }
                if(k[q-1]<p)
                {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"O valor inserido");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"e insuficiente...");
            lcd_goto(40);
            lcd_puts(str);
            for(f=0;f<999999;f++){}
            e=0;
                }
                if(k[q-1]>p)
                {
                        c=k[q-1]-p;
                        if(c==1)
                        {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Troco: %ld euro",c);
            lcd_goto(0);
            lcd_puts(str);
            for(f=0;f<999999;f++){}
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Obrigado...");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"Boa viagem...");
            lcd_goto(40);
            lcd_puts(str);
            RA5=0;RA2=1;
            for(f=0;f<999999;f++){}
            RA5=1;RA2=0;cont++;   carro--;
            car[n]=0;  e=11;d=1;
                }
            else
            {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Troco: %ld euros",c);
            lcd_goto(0);
            lcd_puts(str);
            for(f=0;f<999999;f++){}
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Obrigado...");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"Boa viagem...");
            lcd_goto(40);
            lcd_puts(str);
            RA5=0;RA2=1;
            for(f=0;f<999999;f++){}
            RA5=1;RA2=0;cont++;carro--;
            car[n]=0;e=11;d=1;
            }
    }
```

```
        q=11;
    }
    if(q==0)
    {
        k[0]=w;
        if(k[0]==1)
        {
            lcd_clear();
            DelayMs(50);
            sprintf(str,"Pagar:");
            lcd_goto(0);
            lcd_puts(str);
            sprintf(str,"%ld euro",k[0]);
            lcd_goto(6);
            lcd_puts(str);
            sprintf(str,"ok[#]");
            lcd_goto(55);
            lcd_puts(str);
            sprintf(str,"[*]apagar");
            lcd_goto(40);
            lcd_puts(str);
            }
            else
            {
                lcd_clear();
                DelayMs(50);
                sprintf(str,"Pagar:");
                lcd_goto(0);
                lcd_puts(str);
                sprintf(str,"%ld euros",k[0]);
                lcd_goto(6);
                lcd_puts(str);
                sprintf(str,"ok[#]");
                lcd_goto(55);
                lcd_puts(str);
                sprintf(str,"[*]apagar");
                lcd_goto(40);
                lcd_puts(str);
            }
        }
        if(q>0 && q<10)
        {
            k[q]=(k[q-1]*10)+w;
            if(k[q]==1)
            {
                lcd_clear();
                DelayMs(50);
                sprintf(str,"Pagar:");
                lcd_goto(0);
                lcd_puts(str);
                sprintf(str,"%ld euro",k[q]);
                lcd_goto(6);
                lcd_puts(str);
                sprintf(str,"ok[#]");
                lcd_goto(55);
                lcd_puts(str);
                sprintf(str,"[*]apagar");
                lcd_goto(40);
                lcd_puts(str);
            }
            else
            {
```

```
                                    lcd_clear();
                                    DelayMs(50);
                                    sprintf(str,"Pagar:");
                                    lcd_goto(0);
                                    lcd_puts(str);
                                    sprintf(str,"%ld euros",k[q]);
                                    lcd_goto(6);
                                    lcd_puts(str);
                                    sprintf(str,"ok[#]");
                                    lcd_goto(55);
                                    lcd_puts(str);
                                    sprintf(str,"[*]apagar");
                                    lcd_goto(40);
                                    lcd_puts(str);
                              }
                        }
                        }
                  }
            }
while(e!=11);
// termina quando a tecla for confirmar
}
}while(d!=1);
            break;
            case 11://******** ENTRAR **********************
            if(x>17)
            {
v=0;
                  i=0;
                  for(t=0;t<18;t++)
                  {
                        if(car[t]==0)
                        {
                  lcd_clear();
                  DelayMs(50);
                  car[t]=segundos;
                  sprintf(str,"O codigo: %ld   %d",car[t],t);
                  lcd_goto(0);
                  lcd_puts(str);
                  sprintf(str,"ok[#]");
                  lcd_goto(55);
                  lcd_puts(str);
                  sprintf(str,"[*]sair");
                  lcd_goto(40);
                  lcd_puts(str);
                  e=ler_tecla();
                  if(e!=10 && e!=11)
                  {
                        while(e!=10 && e!=11)
                        {
e=ler_tecla();
}
                  }
                  if(e==11)
                  {
                        carro++;v=1;RA1=0;RA3=1;
                        lcd_clear();
                        DelayMs(50);
                        sprintf(str,"Veiculo a entrar...");
                        lcd_goto(0);
                        lcd_puts(str);
                        sprintf(str,"Espere por favor...");
```

```
                    lcd_goto(40);
                    lcd_puts(str);
                    for(f=0;f<999999;f++){}
                    RA3=0;RA1=1;break;
              }
                    if(e==10)
                    {
                          v=1;a=11;car[t]=0;
                    }
              }
        }
        if(v!=1)
        {
              lcd_clear();
              DelayMs(50);
              sprintf(str,"Nao ha vagas");
              lcd_goto(0);
              lcd_puts(str);
              i=1;
              for(f=0;f<700000;f++){}
              break;
        }
}
if(v!=1 && i!=1)
{
        if(car[x]==0)
        {
              lcd_clear();
              DelayMs(50);
              car[x]=segundos;
              sprintf(str,"O codigo: %ld   %d",car[x],x);
              lcd_goto(0);
              lcd_puts(str);
              sprintf(str,"ok[#]");
              lcd_goto(55);
              lcd_puts(str);
              sprintf(str,"[*]sair");
              lcd_goto(40);
              lcd_puts(str);
        }
        e=ler_tecla();
        if(e!=10 && e!=11)
        {
              while(e!=10 && e!=11){e=ler_tecla();}
        }
              if(e==11)
              {
                    carro++;
                    x++;RA1=0;RA3=1;
                    lcd_clear();
                    DelayMs(50);
                    sprintf(str,"Veiculo a entrar...");
                    lcd_goto(0);
                    lcd_puts(str);
                    sprintf(str,"Espere por favor...");
                    lcd_goto(40);
                    lcd_puts(str);
                    for(f=0;f<999999;f++){}
                    RA3=0;RA1=1;
                    break;
              }
              if(e==10)  {a=11;car[x]=0;}
```

```
            }
            break;
}
}/*Fim do ciclo infinito*/}/*Fim do programa*/
```

Anexo 11: Código fonte desenvolvido para controlo dos *displays* de sete segmentos

```
#include <pic18.h>

//        Definiu-se que:
//        RB4 = bit 1 --> DA
//        RB5 = bit 2 --> DB
//        RB6 = bit 3 --> DC
//        RB7 = bit 4 --> DD
//        RB0 = Interruptor p/ decrementar-> Display 1
//        RB1 = Interruptor p/ incrementar-> Display 1
//        RB2 = Interruptor p/ decrementar-> Display 2
//        RB3 = Interruptor p/ incrementar-> Display 2
//        RC5 = Interruptor p/ on/off enable

// FUNÇÕES PARA A REPRESENTAÇÃO BINÁRIA DOS NÚMEROS ENTRE 0
E 9
void nove()  {RB4=1;RB5=0;RB6=0;RB7=1;}
/* FUNÇÃO PARA O NÚMERO 9 */
void oito()  {RB4=0;RB5=0;RB6=0;RB7=1;}
/* FUNÇÃO PARA O NÚMERO 8 */
void sete()  {RB4=1;RB5=1;RB6=1;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 7 */
void seis()  {RB4=0;RB5=1;RB6=1;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 6 */
void cinco() {RB4=1;RB5=0;RB6=1;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 5 */
void quatro(){RB4=0;RB5=0;RB6=1;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 4 */
void tres()  {RB4=1;RB5=1;RB6=0;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 3 */
void dois()  {RB4=0;RB5=1;RB6=0;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 2 */
void um()    {RB4=1;RB5=0;RB6=0;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 1 */
void zero()  {RB4=0;RB5=0;RB6=0;RB7=0;}
/* FUNÇÃO PARA O NÚMERO 0 */
void display_1(int x)
{
     RC5=1;
     switch(x)
     {
          case 0 :zero();     break;
          case 1 :um();       break;
          case 2 :dois();     break;
          case 3 :tres();     break;
          case 4 :quatro();   break;
          case 5 :cinco();    break;
          case 6 :seis();     break;
          case 7 :sete();     break;
          case 8 :oito();     break;
          case 9 :nove();     break;
     }
}
```

```
void display_2(int x)
{
     RC5=0;
     switch(x)
     {
          case 0 :zero();     break;
          case 1 :um();       break;
          case 2 :dois();     break;
          case 3 :tres();     break;
          case 4 :quatro();   break;
          case 5 :cinco();    break;
          case 6 :seis();     break;
          case 7 :sete();     break;
          case 8 :oito();     break;
          case 9 :nove();     break;
     }
}
void main()
{
     unsigned long int atraso,en1,en2;
     int v1,v2,r1,r2,r3,r4;
     TRISB=0b11000011;
     TRISC=0b00000000;
     ADCON1=0b00001111;
     RB0=1; RB1=1; RB2=1; RB3=1;
     v1=9; v2=9; en1=0; en2=0;
     r1=0; r2=0; r3=0; r4=0;
     while(1)
     {
          for(atraso=0;atraso<60000;atraso++){}
          if(r1==0)
          {
               if(RB0==0)
               {
                    if(v1==9 || v1>9) v1=9;
                    else v1++;
               }
               else v1=v1;
/* Código correspondente ao display do 1º piso com a
variável "v1" responsável por guardar o número de vagas do
1º piso */
          }
          else v1=v1;
          if(r2==0)
          {
               if(RB1==0)
               {
                    if(v1==0 || v1<0) v1=0;
                    else v1--;
               }
               else v1=v1;
          }
          else v1=v1;
          if(r3==0)
          {
               if(RB2==0)
               {
```

```
                    if(v2==9 || v2>9) v2=9;
                    else v2++;
               }
               else v2=v2;
/* Código correspondente ao display do 2 piso com a variável
"v2" responsável por guardar o número de vagas do 2º piso */
          }

          else v2=v2;
          if(r4==0)
          {
               if(RB3==0)
               {
                    if(v2==0 || v2<0) v2=0;
                    else v2--;
               }
               else v2=v2;
          }
          else v2=v2;
          for(en1=0;en1<1000;en1++){} display_1(v1);
          for(en2=0;en2<1000;en2++){} display_2(v2);
          if(RB0==1) r1=0; else r1=1;
          if(RB1==1) r2=0; else r2=1;
          if(RB2==1) r3=0; else r3=1;
          if(RB3==1) r4=0; else r4=1;
     }/*Fim do ciclo infinito*/
}/*Fim do programa*/
```

Anexo 12: Orçamento do projecto

Descrição	Quantidade	Referência do fornecedor	Preço unitário	Preço final
LED verdes	1	708-8022	0,096 €	0,096 €
LED vermelhos	1	708-8035	0,083 €	0,083 €
LED amarelo	1	708-2771	0,101 €	0,101 €
Switches para pisos	4	706-4199	1,24 €	4,96 €
PIC 18F2520	2	623-0667	4,98 €	9,96 €
Teclado numérico	1	115-6031	8,62 €	8,62 €
Displays de 7 segmentos	2	235-8985	1,27 €	2,54 €
Resistências 270 Ω	14	707-7625	0,014 €	0,196 €
LCD 2x20	1	532-6600	12,54 €	12,54 €
Cristal 10 MHz	2	693-8790	0,57 €	1,14 €
Condensador 100 µF	2	520-0867	0,09 €	0,18 €
Condensador 27 µF	4	495-672	1,896 €	7,584 €
Condensador 10µF	2	703-5173	0,68 €	1,36 €
Max 232	2	540-5359	3,5 €	7 €
Condensador 1 µF	12	381-463	1,39 €	16,68 €
Resistência carvão 100 kΩ	2	707-7824	0,014 €	0,028 €
Resistência carvão 10 kΩ	6	707-7745	0,014 €	0,028 €
Díodo 1N4004	2	704-4004	0,051 €	0,102 €
LED da alimentação da placa	2	668-6492	0,36 €	0,72 €
Switches para placa	2	682-1471	0,66 €	1,32 €
Regulador de tensão	2	714-0792	0,368 €	0,736 €
Ligação RS232	2	692-7434	2,54 €	5,08 €
Ficha Molex (para 7 seg) M	2	679-5603	0,099 €	0,198 €
Ficha Molex (para 7 seg) F	2	687-7922	0,071 €	0,142 €
Fonte de alimentação	1	706-6079	43,18 €	43,18 €
Condutor 0,25mm^2	8	208-5479	0,36 €	2,88 €
Descodificador BCD	2	306-718	1,62 €	3,24 €
Socket 28 pinos	2	612-8227	0,15 €	0,3 €
Socket 16 pinos	2	612-8160	0,19 €	0,38 €
Total				131,4 €

Anexo 13: Mapa de Gantt

Id	Nome da tarefa	Duração	Início	Término
1	Introdução à utilização de PICs	16 dias	Qui 30-09-10	Qui 21-10-10
2	Planeamento da estrutura do parque	5 dias	Seg 04-10-10	Sex 08-10-10
3	Layout da maqueta	3 dias	Seg 11-10-10	Qua 13-10-10
4	Construção do diagrama de blocos	3 dias	Qua 13-10-10	Sex 15-10-10
5	Final do 1º Periodo	0 dias	Ter 19-10-10	Ter 19-10-10
6	Construção do fluxograma	3 dias	Seg 18-10-10	Qua 20-10-10
7	Elaboração da lista de material e orçamento	3 dias	Qua 20-10-10	Sex 22-10-10
8	Estudo de programação de PICs	19 dias	Qua 20-10-10	Seg 15-11-10
9	Desenvolvimento do esquema eléctrico e construção da placa de funcionamento da pic.	20 dias	Qui 21-10-10	Qua 17-11-10
10	Estudos de electrónica	18 dias	Seg 25-10-10	Qua 17-11-10
11	Construção de circuitos para testes em placa branca	20 dias	Qua 27-10-10	Ter 23-11-10
12	Resolução de problemas relacionados com as ligações do teclado e do lcd	29 dias	Seg 08-11-10	Qui 16-12-10
13	Final do 2º Periodo	0 dias	Ter 23-11-10	Ter 23-11-10
14	Programação do projecto	26 dias	Sex 03-12-10	Sáb 08-01-11
15	Testes, detecção de erros e sua correcção	25 dias	Seg 15-11-10	Sex 17-12-10
16	Férias do Natal	11 dias	Sex 17-12-10	Sex 31-12-10
17	Pesquisa para a elaboração do relatório	7 dias	Seg 20-12-10	Qui 30-12-10
18	Elaboração do relatório	23 dias	Sex 17-12-10	Ter 18-01-11
19	Construção da maquete	6 dias	Seg 10-01-11	Seg 17-01-11
20	Implementação dos circuitos na maquete	5 dias	Ter 11-01-11	Seg 17-01-11
21	Finalização do projecto	12 dias	Seg 03-01-11	Ter 18-01-11
22	Final do 3º Periodo	0 dias	Ter 18-01-11	Ter 18-01-11
23	Impressão e entrega do relatório	2 dias	Qua 19-01-11	Qui 20-01-11
24	**Reuniões**	**76 dias**	**Ter 05-10-10**	**Ter 18-01-11**

Bibliografia

- *NXP*, HEF4069UB, 9 de Dezembro de 2010, <http://www.nxp.com/documents/data_sheet/HEF4069UB.pdf>;
- *Datasheet Catalog*, HEF4001B gates Quadruple 2-input Nor Gate, 9 de Dezembro de 2010, <http://www.datasheetcatalog.org/datasheet/philips/HEF4001B.pdf>;
- *Datasheet Catalog*, HEF4081B gates Quadruple 2-input And Gate, 9 de Dezembro de 2010, <http://www.datasheetcatalog.org/datasheet/philips/HEF4081BT.pdf>;
- *Datasheet Catalog*, HEF4511B MSI BCD to 7-segment latch/decoder/driver, 18 de Novembro de 2010, <http://www.datasheet catalog.org/datasheet/philips/HEF4511BN.pdf>;
- *Microchip*, PIC18F2420/2520/4420/4520 DataSheet, 20 de Dezembro de 2010, <http://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf>;
- *JHD Products*, JM202A, 18 de Novembro de 2010, <http://www.jinghua-displays.com/pdf/JM202ACATA.pdf>;
- *Farnell*, Technical Data, 14 de Novembro de 2010, <http://www.farnell.com/datasheets/10874.pdf>;