

**Mini Projeto - Microprocessadores**

Licenciatura Engenharia Automóvel

Miguel João Agostinho Ribeiro

Leiria, março de 2020

Índice

[Lista de Figuras iv](#_Toc38705378)

[Lista de tabelas v](#_Toc38705379)

[Lista de siglas e acrónimos vi](#_Toc38705380)

[1. Preparação Miniprojecto – Módulo I/O 1](#_Toc38705381)

[1.1. Inputs e Outputs 1](#_Toc38705382)

[1.1.1. Inputs 1](#_Toc38705383)

[1.1.2. 1](#_Toc38705384)

[1.1.3. Outputs 1](#_Toc38705385)

[1.2. LCD 2](#_Toc38705386)

[1.2.1. Hardware - Comandos e Ligações 2](#_Toc38705387)

[1.3. LED’s 3](#_Toc38705388)

[1.4. Teclado Matricial 4x3 3](#_Toc38705389)

[1.5. BUZ 3](#_Toc38705390)

[2. Preparação Miniprojecto – Interrupções 4](#_Toc38705391)

[2.1. Teclado Matricial 4x3 4](#_Toc38705392)

[2.1.1. Fundamental & hardware 4](#_Toc38705393)

[2.1.2. Software 5](#_Toc38705394)

[2.2. Timers 5](#_Toc38705395)

[3. PWM 6](#_Toc38705396)

[4. ADC – Sensor de Temperatura 7](#_Toc38705397)

[4.1. Características 7](#_Toc38705398)

[5. Funcionalidades extra 8](#_Toc38705399)

[Bibliografia ou Referências Bibliográficas 9](#_Toc38705400)

[Anexos 10](#_Toc38705401)

# Lista de Figuras

Elemento a figurar, **quando aplicável**.

[Figura 2.1 - Texto ilustrativo da figura 1. 2](file:///C:\Users\Marta.Henriques\Documents\03_Arq_2019\Modelo_dissertação_relatorio_projeto.docx#_Toc1407571)

[Figura 3.1 - Texto ilustrativo da figura 2. 5](file:///C:\Users\Marta.Henriques\Documents\03_Arq_2019\Modelo_dissertação_relatorio_projeto.docx#_Toc1407572)

# Lista de tabelas

Elemento a figurar, **quando aplicável**.

Tabela 1.1 - Nomenclatura e ligações ao microcontrolador 1

# Lista de siglas e acrónimos

Elemento a figurar, **quando aplicável**.

|  |  |
| --- | --- |
| ESTG | Escola Superior de Tecnologia e Gestão |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Preparação Miniprojecto – Módulo I/O

## Inputs e Outputs

### Inputs

|  |  |
| --- | --- |
| **Pino** | **Funcionalidade** |
| RA0 | Sensor Temperatura [ADC] |
| RB0 | Teclado – coluna 1 |
| RB1 | Teclado – coluna 2 |
| RB2 | Teclado – coluna 3 |
| RC7 | Conexão Computador [RS232 - Rx] |

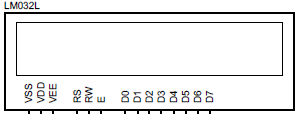
### 

### Outputs

|  |  |
| --- | --- |
| **Pino** | **Funcionalidade** |
| RA1 | LED |
| RB3 | Teclado – linha A |
| RB4 | Teclado – linha B |
| RB5 | Teclado – linha C |
| RB6 | Teclado – linha D |
| RC2 | BUZ [PWM] |
| RC6 | Conexão Computador [RS232 -Tx] |
| RD0 | LCD – D4 |
| RD1 | LCD – D5 |
| RD2 | LCD – D6 |
| RD3 | LCD – D7 |
| RD4 | LCD – RS |
| RD5 | LCD – RW |
| RD6 | LCD – E |
| RD7 | LCD - VDD |

## LCD

### Hardware - Comandos e Ligações

Iremos usar um LCD 20x2 (LM032L).

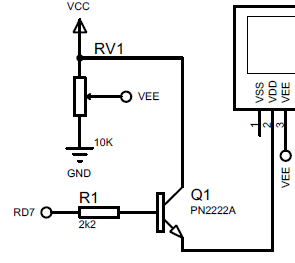
O controlo do módulo LCD efetua-se através de um sistema de processamento, normalmente

através de um barramento paralelo de 8 linhas de I/O bidirecionais (modo 8-*bit*).

Contudo, iremos usar um modo especial, designado modo de 4-*bit*, para o qual o barramento necessita apenas de 4 linhas de I/O bidirecionais.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pino No:** | **Nome** | **Função** | **Pino do Microcontrolador** |
| **1** | **VSS** | Massa (Ground) | - |
| **2** | **VDD** | Fonte de alimentação do pino positivo (5V DC) | - (controlado pelo RD7) |
| **3** | **VEE** | Ajuste de contraste | - (ex.: potenciómetro) |
| **4** | **RS** | Seleção do registo (Register selection) | RD4 |
| **5** | **R/W** | Ler ou Escrever (Read or Write) | RD5 |
| **6** | **E** | Enable | RD6 |
| **7** | **D0** | Data | - |
| **8** | **D1** | Data | - |
| **9** | **D2** | Data | - |
| **10** | **D3** | Data | - |
| **11** | **D4** | Data | RD0 |
| **12** | **D5** | Data | RD1 |
| **13** | **D6** | Data | RD2 |
| **14** | **D7** | Data | RD3 |

Tabela .1 – LCD - Nomenclatura e ligações ao microcontrolador

NOTA: O funcionamento de cada pino está esclarecido em detalhe no Anexo 1 (Tutorial - LCD da disciplina).

Para controlar o contraste do ecrã, é usado um potenciómetro, de forma a controlar a tensão no pino VEE.

A alimentação do LCD é controlada pelo pino RD7 do microcontrolador através de um transístor NPN bipolar Q1.

## LED’s

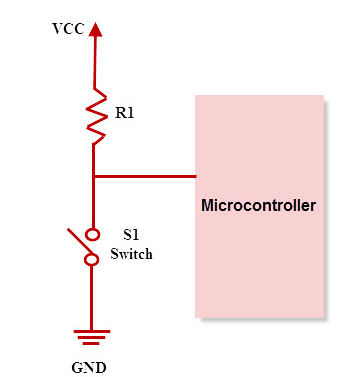
Considerando que o LED será usado como indicador de alarme,

usaremos um LED vermelho (Vfonte = 5V e Vled = 1,7 V; Iled=10mA) e uma resistência de 330 Ohms (calculada R= (Vfonte – Vled) / Iled).

Mas no simulador iremos usar uma resistência de 470 Ohms.

## Teclado Matricial 4x3

Tal como num botão, num teclado matricial também é necessário ter uma resistência de pull up.

Resistência de pull up são valores fixos, que normalmente estão conectados entre a fonte e o pino.

Circuitos lógicos digitais consistem em 3 estados: alto, baixo e variável (ou de impedância elevada). Quando um pino não está pulled para um valor up ou down, o estado de impedância elevada acontece. Quando este estado ocorre, o output do microcontrolador não é controlado pelo pino de input, ou seja, pelo 0 ou pelo 1 (considerando assim o output desconectado do circuito). As resistências pull up (ou pull down) são a solução puxando o valor de input para um estado alto ‘1’ ou baixo ‘0’.

No caso do microcontrolador que usamos, o port B tem integrado resistências pull-up (100kOhms), necessitando apenas de serem configurados (configurado com o registo WPUB).

## BUZ

O avisador sonoro utilizada comunicação PWM. Foi adicionado ao porto C por necessitar de um pino desse tipo.

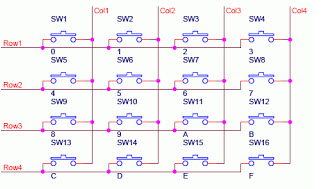
# Preparação Miniprojecto – Interrupções

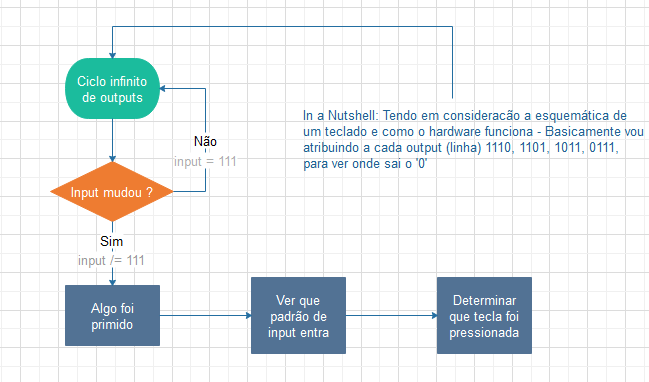
No nosso projeto, as únicas interrupções necessárias serão nos pinos conectados ao teclado matricial (4x3) e os timers.

## Teclado Matricial 4x3

### Fundamental & hardware

O teclado matricial pode ser conectado de várias formas ao microcontrolador, mas a lógica fundamental é as colunas agirem como input para o uC e as linhas como output para o uC.



Em termos de hardware, o teclado matricial consiste em 16 botões, mas que para funcionar apenas necessita de 8 pinos. Os botões estão em cima e quando se pressiona uma tecla (ex.: SW1) é criado um curto circuito entre a coluna 1 e a linha 1. Este curto circuito pode ser detetado pelo microcontrolador, sabendo assim que tecla foi premida:

Imaginando que metemos um output nas linhas ‘1’, a interpretação do funcionamento de um teclado matricial pode ser step by step. Primeiro, ao clicar num botão, uma linha e uma coluna ficam em curto circuito. Segundo, terá de haver software que deteta que coluna foi ativada pela pressão do botão (curto circuito), e isto irá acontecer detetando que input ficou a ‘1’

No nosso trabalho o output irá ser ao contrário (será asserido a ‘0’) pois iremos usar as resistências de pull-up internas do microcontrolador, de modo a poupar hardware e levar a complexidade para o software.

### Software – resumo de implementação

## Timers

# PWM

# ADC

No nosso mini projeto irá haver a necessidade de converter um sinal analógico de um sensor de temperatura num sinal que o microcontrolador consiga processar, ou seja, um sinal digital.

## Características e como funciona

### Sensor

Iremos usar o sensor MCP9701 da Microchip.

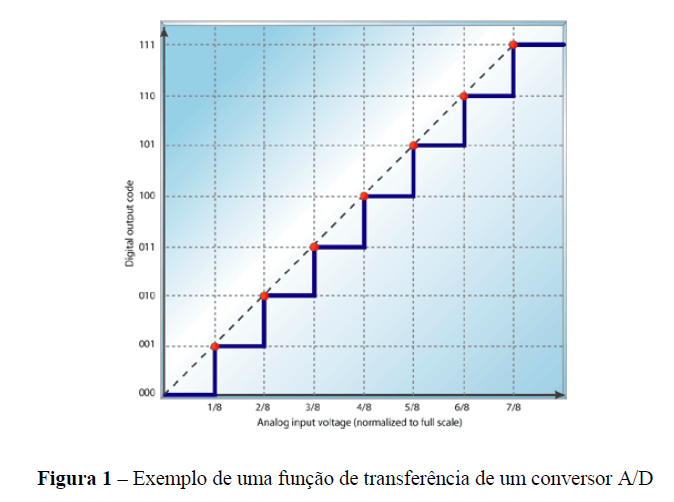
* **Gama de tensão** de alimentação: [3.1V; 5.5V]
* **Precisão**: + 4°C (max.). [0°C; +70°C].
* Otimizado para ADC (**sensibilidade**): 19.5mV/°C (típico).
* **Output**: TA = 0°C: V (0°C) = 400 mV

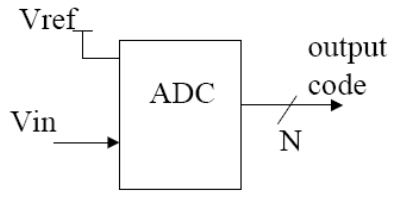
Gama do sensor: [-35; 150] °C -> [-285.50; 3013] mV

Nota: O sensor tem a capacidade de até -40°C mas o limite da tensão de referência para o microcontrolador é -0.3V (-300 mV).

### Microcontrolador

O Microcontrolador permite a conversão de um input analógico de ? Volts para um sinal representativo de no máximo 10 bits em binário, que é chamada de **palavra digital**. Quanto maior o número de bits, maior a **resolução** do conversor ADC (isto traduz-se na conversão mais exata do sinal analógico real, ou seja, o erro será menor entre o valor digital e o valor analógico que o código representa). O **número de níveis (N)** (No gráfico em baixo, um nível é cada ponto) é definido por N = 2bits.



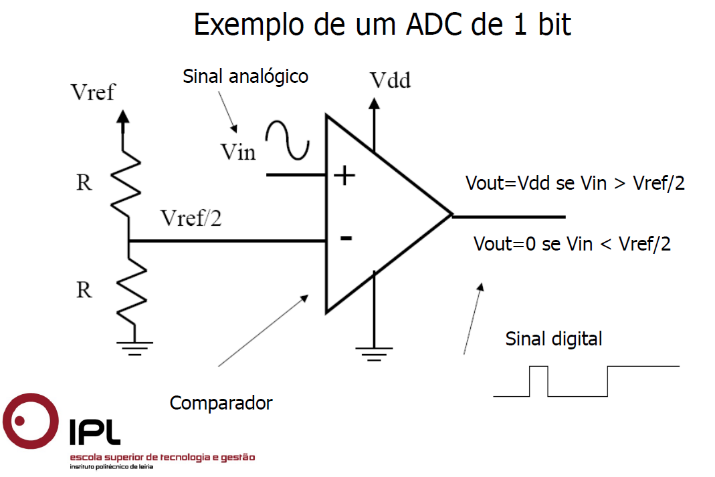


As **tensões de referência** (VREF+ e VREF-) do ADC são configuradas por software. Estas tensões de referência definem a gama de variação das tensões na entrada do conversor ADC e, tal como o número de bits, definem a resolução do conversor. Por exemplo, se estivermos a usar [0V; 5V] e o sensor apenas transmite [0V;2V] iremos perder uma resolução de 3V que não é aproveitada à custa de termos uma gama mais alta.

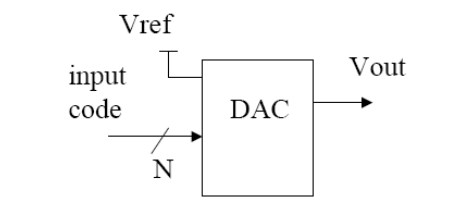
NOTA: O PIC18F45K22 Apenas permite um máximo VREF- = 3,3 V e de VREF+ = 5,3V (isto se a VDD = 5V e VSS = 0V).

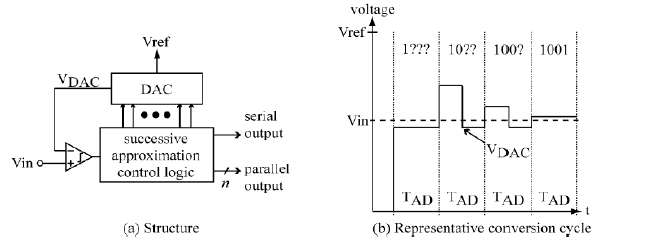
. LSB (***Least Significant Bit***) representa o valor mínimo de tensão na entrada do conversor ADC que pode ser convertido/lido (que é proveniente do sensor), ou seja, o valor em tensão que corresponde a uma variação do bit menos significativo da palavra digital. Isto é importante para determinar a **sensibilidade**.

Se fizermos o inverso, conseguimos determinar o valor medido da grandeza analógica (a tensão input Vin) em função do código de binário obtido. Ou seja, o valor da grandeza corresponde ao código digital (que corresponde ao número de LSB’s medidos) multiplicados pela tensão correspondente a cada LSB e deslocado da tensão de referência mais baixa:

Podemos entender isto de forma mais intuitiva olhando para o hardware do conversor ADC. Ele consiste num Amp Op comparador, em que os valores de input do sensor é comparado com os valores de referência definidos por software:

Resumindo as fórmulas da parte introdutória:

 NOTA: No caso do VDAC (nota que Vout é uma tensão de analógico):

Em termos do **tempo de conversão (**apesar de que o tempo de conversão para esta aplicação não seja um fator tão importante.**),** que depende do tipo de conversor em termos de hardware, o microcontrolador (sendo que existem vários tipos de conversores) usa o conversor de aproximações sucessivas:

O funcionamento é o seguinte: 1° colocar o bit mais significativo do código digital a ‘1’ – através do DAC converte-se esse código em analógico e compara-se esse valor do VDAC com o Vin proveniente do sensor. Se VDAC > Vin então o bit do código digital deve ficar a ‘0’, caso se VDAC < Vin o bit do código digital deverá ser ‘1’.

Depois é fazer este processo do bit mais significativo para o bit menos significativo até se obter o código digital que corresponde ao sinal Vin do sensor.

Ou seja, se Vin for de 3 V e tivermos uma VREF de 5V num conversor de 8 bits (ou seja, o código digital será 0b00000000), a 1ª iteração (0b10000000) irá ser = 2.5V (VDAC = 2.5V, ou seja, VREF/2). Como VDAC<Vin, o bit terá de se manter a ‘1’, **ou seja, agora só estamos numa metade da gama do VREF, neste caso, entre VREF/2 e VREF**. Na 2ª iteração, o código é 0b11000000 e VDAC = 3.75. Como VDAC>Vin o bit irá ficar a ‘0’. E assim sucessivamente….

Nota: Os valores da tensão de referência para, por exemplo 10 bits, correspondem ao código de binário, em decimal, da seguinte forma:

- VREF- = 0;

- VREF+ = 1023(210);

... Este processo é o que o conversor de ADC faz para achar o código digital correspondente ao valor de tensão vindo do sensor analógico.

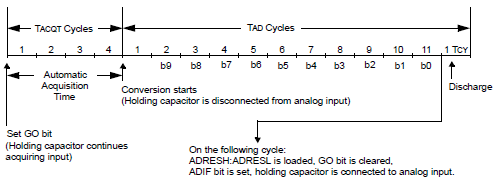
## Configuração

As configurações que vamos utilizar do módulo ADC estão associados aos seguintes registos (**consultar datasheet**):

* ADCON0:
  + Ativar o módulo ADC – logo no inicio;
  + Selecionar o canal em uso;
  + Habilitar o início de uma nova conversão, ou seja, mandar converter (GO\_nDONE) – Este é “flexivel”, ou seja, ativo manualmente.
* ADCON1:
  + Atribuir quais são as tensões de referência (internas ou externas);
* ADCON2:
  + Escolher 8 ou 10 bits – à direta são 10 bits, à esquerda 8 bits;
  + Valores do TACQ e TAD;

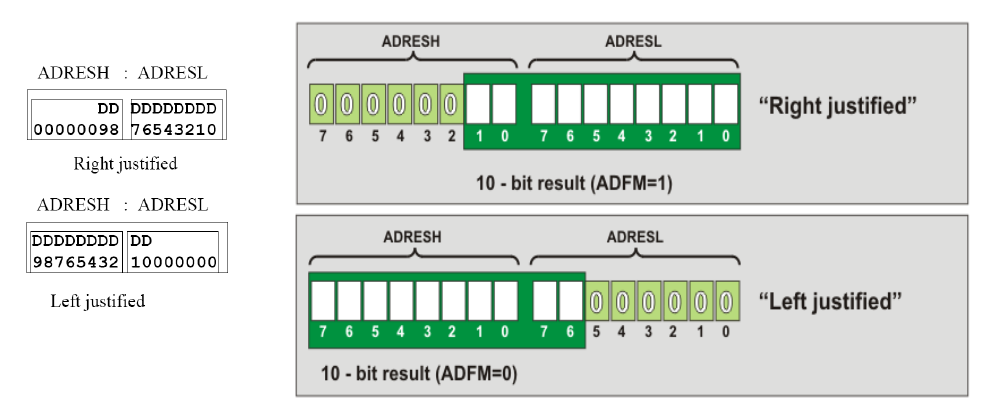
TACQ – Tempo de aquisição – período entre a seleção do canal analógico para o conversor ADC e o início da conversão do referido canal (estabilidade do sinal por causa da impedância do sinal).

TAD – período necessário para efetuar a conversão de 1 bit.



Nota: TCONV = 11 x TAD – tempo de conversão de 10 bits (10 bits + 1 que é do tempo inicial até a conversão volta a acontecer – TCY-TAD).

* ADRESH (ADC Result High Register) e ADRESL (ADC Result Low Register):
  + Registos auxiliares que contém o resultado da conversão digital do sinal analógico, respetivamente o *byte* mais significativo e o *byte* menos significativo da conversão.



* IPR1 – ADIP
  + Interrupção de alta prioridade;
* PIE1 – ADIE
  + Interrupção Enable;
* PIR1 – ADIF
  + Flag da interrupção;

# EUSART – comunicação em serie

Videos a explicar:

<https://www.youtube.com/watch?v=IyGwvGzrqp8>

<https://www.youtube.com/watch?v=XVEnxipCIJ0>

Essencialmente EUSART (Enhanced Universal Synchronous Asynchronous Transmitter Receiver) é um módulo que permite comunicações em série assíncronas e síncronas. Comunicações assíncronas não necessitam de um timer para enviar informação (o transmissor e o recetor têm, um protocolo em que são configurados da mesma maneira de forma a se perceberem um ao outro) – comunicação entre dispositivos, computadores, etc. – e os síncronos usam um timer para comunicar – sensores digitais e microprocessadores, etc.

No nosso miniprojecto apenas iremos usar comunicações assíncronas.

# Funcionalidades extra

# Bibliografia ou Referências Bibliográficas

[1] – [OpenLab Pro](https://openlabpro.com)

# Anexos

[Anexo 1](../2.%20Lab/Ficheiros%20-%20Aulas_Lab/PL1/Tutorial_-_LCD.pdf)

Anexo 2 – [Transístor Bipolar](../../../1%20Sem/IE/Transistor%20Bipolar.pdf)