



INFORMÁTICA INDUSTRIAL 2022/2023

CAPÍTULO INTRODUÇÃO

INTRODUÇÃO

J.P.Santos jps@ua.pt
J.Almeida almeida.j@ua.pt

Guião

Código no paco 42535, Área científica: Engenharia Mecânica. Créditos 6
Escolaridade: aulas teóricas (T) - 2 horas/semana. Aulas práticas laboratoriais (PL) - 2 horas/semana
Idioma(s) de lecionação: Português. Responsável: José Paulo Oliveira Santos

Objectivos

Pretende-se que o aluno seja capaz de controlar e coordenar o funcionamento de equipamentos industriais (shop floor control), aprendendo a utilizar e a desenvolver aplicações informáticas de controlo e supervisão.

Competências

No final, o aluno deve ser capaz de desenvolver os programas de comunicação que permitam o controlo de máquinas-ferramenta, transportadores e robôs, a partir do computador central do laboratório de Sistemas Flexíveis de Produção SFP. Deve ainda ser capaz de implementar, nas aulas práticas, parte da infra-estrutura tecnológica de controlo e comunicação necessária a um sistema de fabrico flexível (Flexible Manufacturing System-FMS).

Conteúdos

1. Programação: Aprendizagem da linguagem de programação Visual Studio 2017 para o desenvolvimento de programas para controlo e supervisão da instalação fabril.
2. Protocolos de comunicação: estudo dos protocolos Rs232, Ethernet, TCP/IP, HTTP.
3. Bases de Dados: Aprendizagem da linguagem SQL. Estudo de bases de dados (ex.MySQL).
4. Interfaces WEB: Estudo das linguagens HTML e PHP para o desenvolvimento de páginas WEB que permitam o controlo e a supervisão remota de equipamentos fabris.

A disciplina tem duas componentes importantes:

T – Teórica. Esta componente é lecionada nas aulas teóricas, no auditório ou remotamente via Zoom, e é posteriormente avaliada através de provas escritas realizadas nas datas oficiais (época normal e época de recurso/melhoria). Os exames da época normal e da época de recurso/melhoria serão provas escritas presenciais, ou via Elearning.

P – Prática. Esta componente é lecionada no laboratório onde são realizados os trabalhos práticos. Cada trabalho prático é avaliado através de um questionário escrito individual. A nota da componente prática provirá da avaliação prática obtida pelo aluno nos questionários escritos, realizados em 3 momentos de avaliação.

AULAS PRÁTICAS

Em 2022/2023, as práticas e os questionários:

(P1,P2,P3)

Trab1 – Vbasic, Editor de texto	(13 Fev)
Trab2 – Vbasic, Máquina de calcular	(20 Fev)
Trab3 – VBasic, Comunicação Rs232 entre computadores	(27 Fev)
<i>1º Momento (Questionários 1, 2 e 3 realizados na Aula Teórica– Programação em Visual Basic e Rs232)</i>	(10Mar)
Trab4 - Programa PLC/Ladder para controlo do Reservatório (Rs232)	(06 Mar)
Trab5 - VBasic, Controlo e supervisão remota do PLC/Reservatório (Rs232)	(13 Mar)
Trab6 – OPC, Controlo e supervisão de um PLC usando um servidor OPC	(20 Mar)
<i>2º Momento (Questionários 4, 5, 6 na Aula Teórica - VBasic e Ladder c/ Rs232, OPC)</i>	(31Mar)
Trab7- TCP/IP, comunicação entre computadores	(27 Mar)
Trab8 - TCPIP+OPC, Supervisão e controlo remoto do PLC	(17 Abr)
Trab9 - VBasic, Servidor WEB/HTTP (VBasic + HTML)	(08 Mai)
Trab10- SCADA remoto do PLC via BrowserWEB (HTML/HTTP/TCPIP/VBasic/OPC	(15 Mai)
Trab11- SCADA remoto do PLC com Bases de Dados MySQL (2 semanas) 11b	(22 Mai) (05 Jun)
<i>3º Momento (Questionários 7,8, 9,10,11 – TCP/IP, HTTP/HTML, MySQL)</i>	(02 Jun)

AULAS TEÓRICAS

T1- Apresentação: Matéria Teórica, Prática, Projetos, Bibliografia, Avaliação.	(17 Fev)
T2- Apresentação: do VBasic (Forms e módulos); do Protocolo EIA 232 e de exemplos em VBasic	(24 Fev)
T3- Supervisão Controlo e a Aquisição de Dados (SCADA) e Especificação OPC	(03 Mar)
<i>T4 - 1º Momento (Questionários 1, 2 e 3 realizados na Aula Teórica– Programação em Visual Basic)</i>	(10 Mar)
T5- Especificação OPC	(17 Mar)
T6- Relembrar os Protocolos de comunicação: Ethernet, IP e TCP	(24 Mar)
<i>T7- 2º Momento (Questionários 4, 5, 6 na Aula Teórica - VBasic e Ladder c/ Rs232, OPC)</i>	(31 Mar)
Férias da Páscoa (3-10 Abr)	
T8- Protocolo HTTP e a linguagem WEB HTML	(14 Abr)
T9- Bases de dados BD - Diagrama de Dependências Funcionais DDF e a normalização da BD	(21 Abr)
Semana académica (24-28 Abr)	
T10- Bases de dados BD - Diagramas de Entidade Relacionamento (DER), normalização da BD	(05 Mai)
T11- Bases de dados BD - Unified Modelling Language (UML - Class Diagram)	(19 Mai)
T12- Processos de negócio de fabrico: Informatização e Automatização, SCADA, MES, Indústria 4.0	(26 Mai)
<i>T13 - 3º Momento (Questionários 7, 8, 9, 10 – TCP/IP, HTTP/HTML)</i>	(02 Jun)

AVALIAÇÃO

Nota Final = 50% T + 50% P

Nota Final = 50% Exame T + 50% P (*1º momento * 3 + 2º momento * 3 + 3º momento * 3*) / 9

T= Nota obtida no exame teórico

P= Média dos 3 momentos de avaliação/questionários.

T≥6.0 e P≥6.0, caso contrário não há lugar a aprovação!

Componente prática de anos anteriores é aceitável desde que seja superior ou igual a 9,5 valores em 20 (>= 9.5)

Obs: As classificações parciais serão expressas com arredondamento às décimas.

Quem não tiver realizado um dos momentos de avaliação (questionários) por razões de saúde, sobreposição com outro momento de avaliação a outra UC, ou outra razão válida, se quiser poderá realizar o(s) questionários em falta no dia do exame da época normal

Quem quiser antecipar a realização do exame teórico da época normal, para o dia 02 de Jun (última aula teórica), poderá fazê-lo. Quem antecipar, já não o fará na época normal.

Em síntese, no dia 02 Jun, ou na época normal, será dado o exame teórico, ou os questionários em falta, conforme preferirem.

Importante é que até à época normal, inclusive, todos tenham sido avaliados à componente teórica (exame teórico) e à componente prática (3 momentos de avaliação baseados em questionários escritos).

Calendário Escolar 2022/2023

			2.º Semestre																																																												
2		FEV												MAR												ABR																																					
0		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28																																		
2																																																															
3																																																															
Férias da Páscoa																																																															
Semana Académica																																																															
MAI		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																															
JUN		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30																																
Época Normal																																																															
Época de Recurso																																																															
Época de Recurso																																																															
			S	T	Q	Q	S	S	D	S	T	Q	Q	S	S	D	S	T	Q	Q	S	S	D	S	T	Q	Q	S	S	D	S	T																															

9:30		Segunda		Terça		Quarta		Quinta		Sexta	
10:00	GIT ANF. 10.1.3 TP1 (TP)	PME 22.3.20 P1 (P)	SVPI 23.1.6 T (T)	PSA 22.4.13 - LEICA P2 (P)	MFC ANF. 22.3.2 TP (TP)	SIAI 22.1.13 - LEICA T (T)	PSA 22.3.20 TP1 (TP)	PME ANF. 22.3.21 TP1 (TP)	EDP ANF. 11.1.10 T (T)	PCP 10.3.2 (EGI) TP2 (TP)	SES 28.02.29 TP (TP)
10:30			SVPI 22.3.9 P1 (P)			SVPI 22.3.9 P3 (P)	PSA 22.3.20 TP2 (TP)	PME ANF. 22.3.21 TP2 (TP)	EDP 22.3.14 P1 (P)		PME 22.3.20 P2 (P)
11:00											
11:30											
12:00											
12:30											
13:00											
13:30											
14:00	SAT 23.3.4 TP (TP)	II 22.3.14 P1 (P)					PSA 22.3.20 TP3 (TP)	PME ANF. 22.3.21 TP3 (TP)	PCP 10.3.2 (EGI) TP1 (TP)	PME 22.3.20 P4 (P)	EDP 22.3.14 P2 (P)
14:30											
15:00											
15:30											
16:00	TPNESC 23.3.10 T (T)	II 22.3.14 P2 (P)	PME 28.01.10 P5 (P)	OENG 23.3.6 TP (TP)	EDEE ANF. 11.1.12 T (T)	PME 23.3.11 P3 (P)		TES 22.3.20 T (T)	EDEE ANF. 22.3.2 P2 (P)	PSA 22.1.13 - LEICA P4 (P)	COMB 22.3.6 TP (TP)
16:30											
17:00	TPNESC 23.3.10 P (P)	II 22.3.14 P3 (P)		OENG 23.3.6 P (P)	EDEE 12.2.8 P1 (P)	PSA 22.1.13 - LEICA P6 (P)		TES 22.3.20 P (P)			
18:30											
19:00											
19:30											
20:00											

Software

VisualStudio

Para instalar o visual studio utilizar o link:

<https://visualstudio.microsoft.com/downloads/>

Selecionar a opção “Community” e depois “.NET development”

Drivers USB/RS232

ATEN USB-RS232 Serial Converter

<http://www.aten.com/global/en/products/usb-&-thunderbolt/usb-converters/uc232a/#.WdIvjGhSyW9>

Prolific PL2303

http://www.prolific.com.tw/US>ShowProduct.aspx?p_id=225&pcid=41

CH340

<https://sparks.gogo.co.nz/ch340.html>

PLC

Software para criar o programa do Autómato (WinProladder):

https://elearning.ua.pt/pluginfile.php/4726299/mod_folder/content/0/WProLad330-24518-EN%281%29.zip?forcedownload=1

Manual do WinProladder:

https://elearning.ua.pt/pluginfile.php/4726299/mod_folder/content/0/WinProladder_Manual_en_v330.pdf?forcedownload=1

Manual do Simulador

https://elearning.ua.pt/pluginfile.php/1556974/mod_folder/content/0/AulaT12_WinProladder_simulation_enu.pdf?forcedownload=1

Módulo de Digital input http://www.esea.cz/support/fatek/FBe_Manual/Hardware/Chapter_5.pdf

Módulo de Digital Output http://www.esea.cz/support/fatek/FBe_Manual/Hardware/Chapter_6.pdf

Manual do módulo Ethernet do PLC: http://file.fatek.com/en/PLC/FBs/FBs_CM25E/FBs_CM25E_Datasheet_en.pdf

- Ethernet Module User's Manual**, informações gerais sobre a carta de expansão.
<https://www.esea.cz/support/fatek/Ethernet%20Module/fbs-ether-enu.pdf>
- Advanced Function Chapter 12: The Communication Function of FBs-PLC**, para configurar os parâmetros de comunicação RS232:
https://www.esea.cz/support/fatek/Supplement_FBs/Chapter12-Comm-Modem.pdf
- Advanced Function Chapter 13: The Applications of FBs-PLC Communication Link**, para ver que registos é necessário configurar:
https://www.esea.cz/support/fatek/FBs_Manual/Manual_2/Chapter_13.pdf

FACON Server for Win8

Fatek FACONSrv (programa)

http://www.fatek.com/en/download.php?f=data/ftp//PLC/Fatek_Server/FaSvr116-16523_en.zip

Fatek FACONSrv (Manual)

http://www.fatek.com/en/data%2Fftp%2FPLC%2FFatek_Server%2FFatekServerActiveX_enu.pdf

Tools for creating UML diagrams

Licença de estudante para usar o LucidChart,

- Free Education (Pro) Accounts for Teachers and Students
- <https://www.lucidchart.com/user/110206479#/education>

- Violet (free)

- <http://horstmann.com/violet/>

- Rational Rose

- <http://www.rational.com/>

- Visual Paradigm UML Suite (trial)

- <http://www.visual-paradigm.com/>

- (nearly) direct download link:

- <http://www.visual-paradigm.com/vp/download.jsp?product=vpuml&edition=ce>

Bibliografia

Visual Basic Tutorial 2017	https://youtu.be/3FkWddODLno
INDUSTRY 4.0 - The Fourth Industrial Revolution based on Smart Factories (38 min)	https://www.youtube.com/watch?v=vWrbBcY2-Ms
Intelligent Factory (5 min)	https://www.youtube.com/watch?v=HPRURtORNis
Industry 4.0 - Next Steps (13 min)	https://www.youtube.com/watch?v=XZF10XrowGU
Smart Manufacturing: The Brilliant Factory (5 min)	https://www.youtube.com/watch?v=SfVUkGoCA7s
Serial Communication (7 min)	https://www.youtube.com/watch?v=cu8iFT7xARA
Serial Communication RS232 & RS485 (12 min)	https://www.youtube.com/watch?v=2DQdEHvnqvI
Back to Basics: SCADA (7 min)	https://www.youtube.com/watch?v=bfxr5DikdP0
E-Learning SCADA Lesson 1- What is SCADA? (8 min)	https://www.youtube.com/watch?v=5ZiIA-kMV8M
http://electrical-engineering-portal.com/an-introduction-to-scada-for-electrical-engineers-beginners (5 min)	
Lecture 4 - SCADA Concepts - Part 1 (Power grid) (12 min)	https://www.youtube.com/watch?v=szXCj4iQOQM
Lecture 4 - SCADA Concepts - Part 2 (Telemetry) (13 min)	https://www.youtube.com/watch?v=xewRu1RFp04
Movicon 11 - SCADA Software PowerPoint (28 min)	https://www.youtube.com/watch?v=2AgJCPj0wN8
Enterprise Resource Planning (ERP)	
What is SAP? Why do we need ERP? (8 min)	https://www.youtube.com/watch?v=lYCEQqSM08I
SAP to Shop Floor (3 min)	https://www.youtube.com/watch?v=z_Auifovjcc
Production Planning & Control (PP&C)	
SAP Production Planning & Manufacturing;	
Introduction to SAP PP, SAP Production Planning & Control (27 min)	
https://www.youtube.com/watch?v=geDTPFB2JEQ	
SAP tutorial for beginners (34 min)	https://www.youtube.com/watch?v=Nfq3OC6B-CU
SAP PM - CURSO BÁSICO GRÁTIS	https://www.youtube.com/watch?v=xtnicsu6Qwc
- Os sistemas de execução da produção (MES)	
Manufacturing Execution System (40 min)	https://www.youtube.com/watch?v=pKjRiHF0Obw
SAP Manufacturing Execution: Integrate and Optimize Operations	http://www.sap.com/solution/lob/manufacturing/ software/execution/index.html
Manufacturing Execution Service Offering SAP ME	https://www.sap.com/products/execution-mes.html
Sobre OPC	
OPC Foundation	https://opcfoundation.org
Introduction to OPC for Factory Automation and Plant Process Control	https://www.youtube.com/embed/16fM_7vZ_FE
What is OPC? Part 1: OPC UA Overview (1 min)	https://www.youtube.com/watch?v=tDGzwsBokY
What is OPC UA	http://www.matrikonopc.com/opc-ua/index.aspx
Exemplo em VBasic	
https://social.microsoft.com/Forums/en-US/608798b5-986d-418d-a7c0-1d6d2211843d/connecting-a-opcsERVER-in-vbnet?forum=vblanguage	
Similar issue: OPC Automation	
http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=324042&SiteID=1	
http://www.cnblogs.com/Phoenix-Rock/archive/2007/11/07/541696.html	
http://www.codeproject.com/Articles/18525/OPCWare-DA-Automation-Wrapper-VB-NET	
Matrikon	
http://www.matrikonopc.com	
http://www.matrikonopc.com/training/opc-multimedia-tutorial-new/index.html	

Programas (demo)
<http://www.matrikonopc.com/products/opc-desktop-tools/index.aspx>
<http://www.matrikonopc.com/training/opc-demo-tutorial.pdf>

Kepware
<https://www.kepware.com>
<https://www.kepware.com/support/videos/>

Programas(demo)
<https://info.kepware.com/opc-foundation-kepserverex-download>
<https://www.kepware.com/products/kepserverex/documents/kepserverex-manual.pdf>
Free OPC servers <http://www.opcconnect.com/freesrv.php>
Free OPC Clients <http://www.opcconnect.com/freecli.php>
Manufacturing Plant Floor to SAP Connectivity using OPC <https://www.youtube.com/watch?v=DNVc-2cUjdo>
OPC Client driver/server (free), instalar e experimentar. <https://opcxpert.com/>

Sobre TCP/IP

Normas TCP / IP

IP protocol specification <http://www.rfc-editor.org/rfc/rfc791.txt>
TCP protocol specification <https://www.rfc-editor.org/rfc/rfc793.txt>
TCP/IP tutorial <https://tools.ietf.org/html/rfc1180>

01 - Understanding Local Area Networking (60 min). LAN: Data, nodes, Client, Server, Peer, Net adapter, hub, switch, router, meio, Transport protocol, Largura de banda, wireless access point, serial, data rate transfer, IP address, Virtual LAN, Topologia, Ethernet, Frames, Tipos de Servidores <https://www.youtube.com/watch?v=t9TmvFvYfWw>

02 - Defining Networks with the OSI Model (41 min)

Standards organization: ISO, ANSI, IEEE, ISO , OSI Seven layers <https://www.youtube.com/watch?v=HOI6gKO5QcY>

03 - Understanding Wired and Wireless Networks (48 min)

Twisted-pair (Crossover or Straight), RJ45, Atenuação, interferência, STP – Shielded Twisted-Pair, Fibra ótica
Wireless, Wireless Access Point, Wireless Router, Wireless access modes, Encryption, AP Settings <https://www.youtube.com/watch?v=i25aRHLnZTs>

04 - Understanding Internet Protocol (50 min). IP address: classes, loopback, Binary to decimal, public and private address, DNS, Gateway, NAT, IPv4, IPv6, subnetting

OSI Model Explained (animated)

Hub, Switch, or Router? Network Devices Explained (7 min)

How Network Address Translation Works (10 min)

Automatic IP Address Assignment: How DHCP Works

Inside the Domain Name System (13 min)

ARP Explained - Address Resolution Protocol

Address Resolution Protocol (ARP) Explained

Learn Address Resolution Protocol (ARP) in just 7 Minutes – ARP Tutorial. TCP/IP Explained <https://www.youtube.com/watch?v=ULpPIVln6nI>

Packet Traveling - How Packets Move Through a Network <https://www.youtube.com/watch?v=rYodcvhh7b8>

Routers, Switches, Packets and Frames <https://www.youtube.com/watch?v=zhlMLRNY5-4>

[Farrel] Farrel, A. - *The Internet and its protocols: A comparative approach*, Elsevier,

[Blank] Blank, G.A. - *TCP/IP Foundations*, SYBEX, 2004

[Clark] Clark, G.,Reynders.D., Wright. E., - *Practical Modern SCADA protocols: DNP3, 60879.5 and related systems.* Newnes – Elsevier, 2004
capítulo 2 – Fundamentals of SCADA communications
capítulo 12 – Ethernet and TCP/IP networks
capítulo 13 – Fieldbus and SCADA communication systems

[Mackay] Mackay, S., et.al. - *Practical Industrial Data Networks: Design, Installation and Troubleshooting*, Elsevier, 2004
capítulo 3 – EIA-232 overview
capítulo 4 – EIA-485 overview
capítulo 7 – Modbus overview

capítulo 13 – Profibus PA/DP/FMS overview
capítulo 15 – Industrial Ethernet overview
capítulo 16 – TCP/IP overview

Sobre HTTP

HTTP Crash Course & Exploration (38 min)	https://www.youtube.com/watch?v=iYM2zFP3Zn0
http-protocol.wmv (25 min)	https://www.youtube.com/watch?v=WGJrLqtX7As
Understanding HTTP Request Response Messages (12 min)	https://www.youtube.com/watch?v=sxiRFwQ1RJ4
Introdução ao HTML	http://www.w3schools.com

Sobre PHP

PHP Tutorial 1 - Introduction (PHP For Beginners) (5 min)	https://www.youtube.com/watch?v=kY5P9sZqFas
Web Design and Programming Pt 1 PHP (14 min)	https://www.youtube.com/watch?v=l21g8dJmD7U
Site PHP:	http://www.php.net , Manual de PHP
Site Apache:	http://www.apache.org
Carlos Serrão, Joaquim Marques - Programação com PHP 3 e 4, FCA – Editora de Informática Ida. – Setembro 2000. (cota 681.3.06A.214)	

Sobre SQL

Introduction to SQL http://www.w3schools.com/sql/sql_intro.asp

Sobre MySQL

- MySQL Stored Procedure Programming, Steven F., Guy H., 2006.
Chapter 13
- Teach.Yourself.MySQL.in.10.Minutes, By Sams.Sams, 2006.

Entity Relationship Diagram (ERD) Tutorial - Part 1

<https://www.youtube.com/watch?v=QpdhBUYk7Kk> (7 min - Lucid Chart)
<https://www.youtube.com/watch?v=-CuY5ADwn24> (14 min - Lucid Chart)

Stanford Dbclass - UML-data-modeling (25 min)

<https://www.youtube.com/watch?v=LmS4Y99fNaQ>
Tutorial de Diagrama de Classe UML (lucid chart 10 min)
<https://www.youtube.com/watch?v=UI6lqHOVHic>
Design and UML Class Diagrams
<https://courses.cs.washington.edu/courses/cse403/11sp/lectures/lecture08-uml1.pdf>
A UML Class Diagram Tutorial
<http://faculty.chemeketa.edu/ascholer/cs162/Lectures/Week01/UMLClassDiagramTutorial.pdf>
UML Use Case Diagram Tutorial
<https://www.youtube.com/watch?v=zid-MVo7M-E>
Use Case Diagram - Step by Step Tutorial with Example
<https://www.youtube.com/watch?v=sQgoFjxSdxo>

Organização do documento

No capítulo 2 são apresentadas as atividades industriais, os processos de negócio, a sua automatização e informatização.

No capítulo 3 é apresentado e utilizado o VBasic 2022, os conceitos gerais e alguns exemplos aplicados.

O capítulo 4 descreve o protocolo EIA232, apresenta exemplos em VBasic para interagir com um autómato programável (Programmable Logic Controller - PLC) através de uma ligação EIA232. Também é proposto um trabalho prático no final deste capítulo que consiste no desenvolvimento de um programa do tipo “chat” que permite enviar texto entre computadores através de uma ligação Rs232.

O capítulo 5 relembra a programação de autómatos (Fatek), bem como a configuração das suas interfaces Rs232.

O capítulo 6 descreve a especificação OPC, apresenta exemplos de como é possível em Vbasic utilizar clientes OPC para interagir com recursos fabris, através dos seus servidores OPC.

Os capítulos 7, 8, e 9 apresentam os protocolos *Ethernet*, *Internet Protocol – IP*, e o *Transmission Control Protocol – TCP*. São apresentados exemplos de como é possível desenvolver em Vbasic aplicações clientes e aplicações servidoras que comunicam através da “Internet”.

Os capítulos 10 e 11 apresentam o protocolo HTTP e a linguagem HTML. Cada vez mais, as interfaces de controlo e supervisão recorrem à “Internet”, mesmo em ambientes fabris.

São apresentados exemplos de páginas html e exemplos que permitem desenvolver um “servidor Web”, elementar, em VBasic.

O capítulo 12 aborda as Bases de Dados – BD. Começa por apresentar alguns modelos de bases de dados, descreve como projetar e normalizar as BD. É também apresentada a linguagem SQL e exemplos em Vbasic de como aceder às mesmas, nomeadamente através de ligações ODBC ou diretamente a servidores MySQL.

O capítulo 13, tendo em vista o desenvolvimento de páginas WEB dinâmicas, apresenta a linguagem PHP. São também apresentados exemplos em PHP de como interagir com o gestor de bases de dados MySQL.

Conteúdo

INTRODUÇÃO	i
Guião.....	i
Calendário Escolar 2022/2023	iii
Software	iv
Bibliografia	vi
Organização do documento	ix
1. INTRODUÇÃO	1
1.1. Automação e Informática Industrial.....	4
1.2. Apresentação de algumas DPEs, em linha com Informática Industrial	6
Estudo de eficiência energética nos equipamentos de conformação, na Vista Alegre, SA	6
Rastreabilidade de matérias primas e produtos, e a sua integração com o ERP da empresa, na Vista Alegre.	6
Bike Sharing - Plataforma de reservas e pagamentos.....	7
Proposta de Sistema Automático de Rastreabilidade, na OLI.....	7
Gestão remota de dispositivos (Bosch)	7
Proposta de um sistema automático de rastreabilidade, na indústria 4.0. (na Renault).....	8
Planeamento da manutenção preditiva no contexto da indústria 4.0. (na OLI).....	9
Uma solução para a Manutenção Industrial, na Indústria 4.0. (na Renault).....	10
Algoritmos para a manutenção preditiva, na Indústria 4.0. (na empresa Navigator)	11
Automatic booking system with smart door lock and PayPal payment (“Smart Locker”).....	11
Monitorização e controlo remoto de sistemas de rega agrícola.....	11
Desenvolvimento de módulos para aquisição de sinais para a Indústria 4.0. (na Renault)	11
1.3. Apresentação de algumas colaborações	12
2. PROCESSOS DE NEGÓCIO E DE FABRICO.....	13
2.1. Atividade de uma empresa de fabrico	17
2.1.1. Ambiente de Engenharia: Gestão, Concepção, Planeamento	19
2.1.2. Ambiente de produção: Fabrico, Montagem, Teste.....	20
2.2. Informatização e automatização das atividades da empresa	21
2.2.1. Informatização do ambiente de engenharia	24
2.2.2. Informatização e automatização dos processos de produção.....	29
2.3. Caracterização dos fluxos de informação	33
2.3.1. Fluxos de informação CAD – CAM	33
2.3.2. Fluxos de informação CAD/CAM -> PP&C	35
2.3.3. Fluxos de informação PP&C -> CAD/CAM	36
2.4. Considerações	38
2.5. Sistema Flexível de Produção	39
2.6. Indústria 4.0.....	42
3. INTRODUÇÃO AO VISUAL BASIC	43
3.1. Instalar o Visual Basic 2022	44
3.2. Ambiente integrado de desenvolvimento.....	46
3.3. Construção da interface gráfica dos programas	48
3.3.1. Exemplo “Calculadora_soma”	48
3.3.2. Exemplo “Seleção de imagens”.....	51
3.3.3. Exemplo “Lista de String”.....	53
3.3.4. Exemplo “Calendário”.....	54
3.3.5. Exemplo “Selecionar um ficheiro no disco”.....	55
3.3.6. Exemplo “Ler e gravar ficheiros de texto”.....	57
3.4. Trabalho nº 1 - “Editor de texto”	60
3.5. Linguagem de programação	61
3.5.1. Bases numéricas, prefixos	61
3.5.2. Variáveis.....	61
3.5.3. Declaração de Funções e Procedimentos.....	64
3.5.4. Tipos de operadores.....	66
3.5.5. Estruturas de controlo	68
3.5.6. Biblioteca de funções para manipulação de texto.....	70
3.5.7. Funções matemáticas.....	72
3.6. Projeto com vários Forms e Módulos	73

3.6.1.	Criar um segundo Form.....	73
3.6.2.	Alternar entre Forms.....	74
3.6.3.	Passar dados entre Forms	74
3.6.4.	Módulos com código e dados	76
3.6.5.	Exemplo 1 : Máquina de calcular com 4 operações matemáticas fundamentais	77
3.6.6.	Exemplo 2 : Máquina de calcular com registo das operações realizadas em ficheiro	79
3.7.	Trabalho nº 2 – “Máquina de Calcular”	81
3.7.1.	Introdução.....	81
3.7.2.	Objectivos.....	82
4.	EIA232 - PROTOCOLO DE COMUNICAÇÃO SÉRIE	85
4.1.	Protocolo de comunicação EIA232.....	85
4.1.1.	Topologia.....	88
4.1.2.	Meio comunicação.....	88
4.1.3.	Sinais eléctricos	88
4.1.4.	Formato da palavra série.....	89
4.1.5.	Detecção de erros de transmissão	89
4.1.6.	Tipo de diálogo.....	90
4.1.7.	Controlo do fluxo de dados	90
4.1.8.	Fichas EIA232	90
4.1.9.	Cabos Rs232	91
4.1.10.	Conversão dos níveis TTL para os níveis Rs232.....	92
4.2.	VBasic - Como utilizar e configurar a interface EIA232 do Computador.....	94
4.2.1.	Instalação dos drivers dos conversores USB-Rs232.....	94
4.2.2.	Instalação do com0com	95
4.2.3.	Software de comunicação série Termite	96
4.2.4.	Exemplo - programa “SerialPortTxRx”.....	96
4.2.5.	Objetos do tipo SerialPort.....	98
4.2.6.	Se o seu computador estiver ligado a um modem	100
4.2.7.	Exemplo: Programa simples de comunicação	102
4.3.	Trabalho nº 3 - VBasic, Comunicação Rs232 entre computadores	105
5.	Relembrar a programação de PLCs (Fatek).....	110
5.1.	Arquitectura do PLC	110
5.2.	Bornes do PLC	112
5.2.1.	Entradas digitais	113
5.2.2.	Saídas digitais.....	114
5.2.3.	Saídas e Entradas analógicas	115
5.3.	Memória Interna	117
5.4.	Editor Ladder (WinProladder)	118
5.4.1.	Um exemplo de um programa autómato (Ladder).....	119
5.5.	Módulo de comunicação Rs232, Rs485 e Ethernet da Fatek	122
5.5.1.	Configuração do baudrate , bit da dados, bit paridade, stopbits	122
5.5.2.	Criação da mensagem e envio, usando a função CLINK	124
5.5.3.	Mensagem recebida no PC/Termite (enviada pelo PLC)	125
5.5.4.	Recepção e tratamento da mensagem recebida pelo PLC	126
5.6.	Linguagem Ladder	127
5.6.1.	Tipos de dados Fatek	127
5.6.2.	Temporizador On-delay.....	128
5.6.3.	Temporizador Off-delay	128
5.6.4.	Temporizador (exemplo Fatek)	129
5.6.5.	Contadores	130
5.6.6.	Contadores (Exemplo Fatek).....	130
5.6.7.	Atribuir valores a posições de memória interna do PLC (Mov)	132
5.6.8.	Operações matemáticas (+ , - , / , *)	132
5.6.9.	Conversões	133
5.6.10.	Operações deslocamento	133
5.6.11.	Operações booleanas	134
5.6.12.	Operações Comparação	134
Trabalho nº 4 – Programa PLC/Ladder para controlo do Reservatório	136	
Descrição do trabalho.....	136	
Memória do PLC e mensagens enviadas.....	137	
Memória do PLC e mensagens recebidas.....	138	
Programa PLC.....	139	

Trabalho nº 5 – Controlo e supervisão do PLC/Reservatório, a partir do PC	143
Introdução	143
Objectivos	144
Programa do PLC.....	145
Programa do Computador	145
5.6.5. Apêndice – Programa VBasic.....	149
6. OPC - Ole for Process Control	151
6.1. Conceitos sobre OPC	151
6.2. FACONSRV Servidor da Fatek	155
6.2.1. Configuração do FACONSRV – exemplo.....	156
6.3. VBasic como cliente do servidor FACONSRV	159
6.3.1. Objetos do tipo FaconSrv	159
6.3.2. Exemplo: Cliente OPC – Facon server.....	160
6.3.3. Exemplo II: Cliente OPC – Facon server	161
Trabalho nº 6- Controlo e supervisão de um PLC através de um servidor OPC	162
6.3.4. Introdução.....	162
6.3.2. Objetivos	163
6.3.3. Descrição do trabalho (1 semana).....	164
6.3.4. Bibliografia.....	164
7. ETHERNET	165
7.1. Introdução	165
7.2. Camada física	166
7.2.1. 10Base2 - Meio de transmissão – Coaxial.....	166
7.2.2. 10Base5 - Meio de transmissão – Coaxial.....	168
7.2.3. 10Base-T: Meio de transmissão – UTP, sinais e topologia	168
7.3. Mensagem Ethernet.....	171
8. IP - INTERNET PROTOCOL	174
8.1. Conceitos sobre o protocolo IP	174
8.1.1. Mensagem Internet	175
8.1.2. Endereço IP	177
8.1.3. Internet Control Message Protocol (ICMP).....	178
8.1.4. Address Resolution Protocol (ARP).....	178
8.1.5. Reverse Address Resolution Protocol (RARP)	178
8.1.3. Routing	178
8.1.4. Endereços Internet Públicos e Privados.....	179
8.1.5. Network Address Translation (NAT)	180
8.1.6. Atribuição estática ou dinâmica de endereços IP aos equipamentos quando se ligam à rede	180
8.1.7. Domain Name Service (DNS)	181
9. TCP - TRANSMISSION CONTROL PROTOCOL	182
9.1. Conceitos sobre o protocolo TCP	182
9.1.1. Estabelecimento de uma ligação virtual	183
9.1.2. Estrutura da mensagem TCP	184
9.1.3. Envio das mensagens TCP	186
9.2. VBasic - como configurar e utilizar a interface TCP do computador.....	187
9.2.1. Exemplo de um servidor TCP/IP.....	189
9.2.2. Exemplo de um cliente TCP/IP	190
9.3. Trabalho prático nº 7 – TCPIP comunicação entre computadores.....	191
9.3.1. Descrição do trabalho	192
9.3.2. Exemplo de um Servidor TCPIP	193
9.3.3. Exemplo de um Cliente TCPIP	195
Trabalho prático nº 8 - TCP/IP e OPC, Supervisão e controlo remoto do PLC	197
10. HTTP – HiperText Transfer Protocol	200
10.1. Conceitos sobre HTTP	201
10.1.1. Localização dos recursos na Internet.....	201
10.1.2. Interacções HTTP	201
10.1.3. Mensagens HTTP	202
10.2. VBasic – HTTP	204
10.2.1. Exemplo – Servidor WEB elementar	205
10.2.2. Exemplo – Servidor WEB com duas páginas HTML.....	207

10.2.3.	Exemplo – Servidor WEB com três páginas HTML	210
10.2.4.	Exemplo – Servidor WEB específico para controlar um motor	213
10.2.5.	Exemplo – Servidor WEB específico para controlar um Motor II	214
10.3.	Trabalho prático nº 9 – Servidor WEB/HTTP.....	217
11.	HTML – HiperText Markup Language	225
11.1.	Conceitos sobre HTML	225
11.1.1.	Estrutura de um documento HTML.....	225
11.1.2.	Etiquetas HTML.....	226
11.2.	Exemplos de páginas HTML.....	226
11.2.1.	Estrutura de um documento HTML.....	226
11.2.2.	Formatar texto em HTML	226
11.2.3.	Hiperligações.....	227
11.2.4.	Inserir imagens	228
11.2.5.	Inserir imagens com hiperligação	228
11.2.6.	Inserir imagens com áreas de hiperligação.....	229
11.2.7.	Criar tabelas.....	229
11.2.8.	Criar formulários	230
11.2.9.	Criar frames	230
11.2.10.	Criar frames com targets	231
11.2.11.	Enviar um ficheiro para o servidor.....	232
Trabalho	prático nº 10 – SCADA remoto do PLC via Browser WEB (HTML/HTTP/TCP/IP/OPC)	234
12.	BASES de DADOS	245
12.1.	Conceitos	245
12.2.	Diagramas de Dependências Funcionais	250
12.2.1.	Exemplo - Normalização da BD Alunos	250
12.2.2.	Exemplo - Normalização de algumas relações	254
12.2.3.	Exemplo – Normalização de uma agenda telefónica.....	256
12.2.4.	Exemplo – Normalização de BD de Clientes com conta aberta numa loja	257
12.2.5.	Exemplo – Normalização de uma BD para uma biblioteca.....	258
12.2.6.	Exemplo – Normalização de BD para um Vídeo Clube.....	260
12.3.	Diagramas de Entidade Relacionamento	262
12.3.1.	Diagrama de Ocorrências	262
12.3.2.	Diagrama de Entidade - Relacionamento	262
12.3.3.	Relacionamentos binários 1:1.....	262
12.3.4.	Relacionamentos binários 1:N.....	263
12.3.5.	Relacionamentos binários M:N	263
12.3.6.	Relações necessárias para relacionamentos binários de Grau 1:1	265
12.3.7.	Relações necessárias para relacionamentos binários de Grau 1:N.....	267
12.3.8.	Relações necessárias para relacionamentos binários de Grau N:M	268
12.3.9.	Relações necessárias para relacionamentos de ordem superior	268
12.3.10.	Relações necessárias quando uma entidade tem vários papéis	269
12.3.11.	Exemplo Casas e Pintores	271
12.3.12.	Exemplo Garagem e Mecânicos.....	272
12.3.13.	Exemplo Distribuidor , Lojas e Fornecedores.....	273
12.3.14.	Exemplo Empresa de Investigação	274
12.3.15.	Exemplo - Competições desportivas	276
12.3.16.	Exemplo - Pesca nos Lagos com três associações binárias	279
12.3.17.	Exemplo - Pesca nos Lagos com uma associação ternária	280
12.4.	Diagramas de Entidade-Relacionamento (Não Peter Chen).....	282
12.4.1.	Exemplo - Cliente, Ordens e Produtos	283
12.5.	Linguagem SQL	286
12.6.	VBasic – ODBC	287
12.6.1.	Configuração de uma ligação ODBC para aceder às bases de dados	287
12.6.2.	Acesso à base de dados através de uma ligação ODBC	288
12.7.	VBasic – MySQL	292
12.7.1.	<i>Instalação do MySql</i>	292
12.7.2.	Exemplo I: Base de dados “Alunos” (Visual Basic/MySQL).....	294
12.7.3.	Exemplo II: Da BaseDados “Reservatorio” ao PLC (MySQL + FaconSrv)	300
12.7.4.	Exemplo III: Do BrowserWEB à BaseDados (HTTP/TCP/IP + MySQL)	304
12.7.5.	Exemplo IV: Do Browser ao PLC, com Base de dados	308
Trabalho Prático	nº 11 - SCADA remoto do PLC c/ Base de Dados MySql	314

13. PHP	318
13.1. Conceitos	318
13.2. Tipos de variáveis	320
13.3. Tipos de operadores.....	321
13.3.1. Operadores aritméticos	321
13.3.2. Operadores relacionais/comparação	321
13.3.3. Operadores lógicos	322
13.3.4. Operadores bit a bit	322
13.4. Estruturas de controlo.....	324
13.4.1. WHILE	324
13.4.2. FOR	324
13.4.3. IF	324
13.4.4. SWITCH.....	325
13.5. Funções e procedimentos.....	326
13.5.1. Passagem de parâmetros por valor	326
13.5.2. Passagem de parâmetros por referência.....	326
13.5.3. Retorno de valores.....	327
13.6. PHP - Calculadora	328
13.7. PHP – Acesso às bases de dados MySQL	329
13.7.1. Interação entre o Browser WEB, o Servidor WEB e o Servidor MySQL	329
13.7.2. Funções PHP para aceder ao Servidor MySQL.....	331



CAPÍTULO

1

1. INTRODUÇÃO

JPSantos

2023

Está em curso uma nova revolução industrial, a Universidade de Aveiro não pode deixar passar esta oportunidade e obrigação histórica. Tem de dar resposta à necessidade urgente e inevitável do País e das suas empresas terem recursos humanos formados para dar resposta a esta nova revolução. Os países que ficarem para trás nesta nova revolução tecnológica estão a condenar a sua população ao empobrecimento.

Diversos líderes Europeus e Mundiais, diversos estudos nacionais e internacionais, identificaram à saciedade que a nova revolução industrial, a 4^a revolução Industrial, atualmente designada por “Indústria 4.0” é a chave para o desenvolvimento das nações.

Em Portugal, “A CIP – Confederação Empresarial de Portugal apresentou o estudo “Automação e o Futuro Do Trabalho Em Portugal”, elaborado em parceria com o McKinsey Global Institute e a Nova School of Business and Economics. O estudo debruça-se sobre o impacto da automação no futuro do trabalho e mede o potencial de automação na economia portuguesa até 2030.”

<http://cip.org.pt/cip-apresenta-estudo-sobre-automacao-e-o-futuro-do-trabalho-em-portugal/>

Constam neste estudo afirmações como:

- “...Apenas a automação pode proporcionar a Portugal a injeção necessária de produtividade para o crescimento da mesma”.

POTENCIAL DE AUTOMAÇÃO

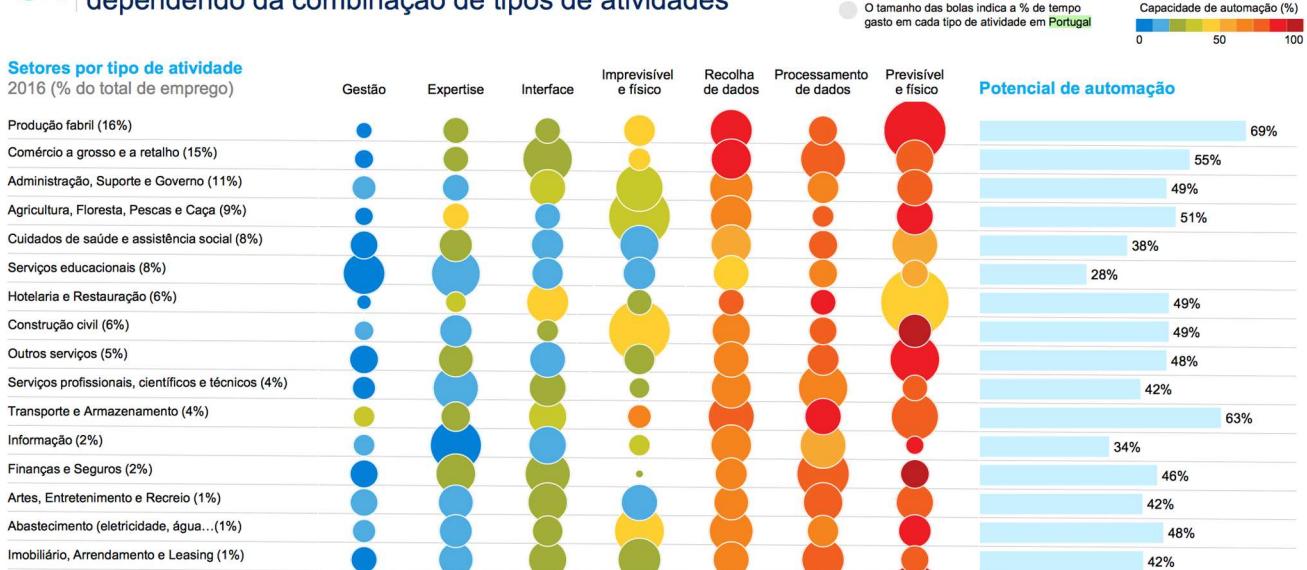
01 | 52% do tempo laboral em Portugal é despendido em tarefas repetitivas altamente automatizáveis, com mais de 70% de potencial de automação



As atividades repetitivas e mais automatizáveis correspondem a 52% do total das horas de trabalho em Portugal

- “0,7 Milhões de trabalhadores (15% do total da força de trabalho) terá de alterar as suas ocupações laborais atuais e adquirir novas capacidades até 2030” .

01 | O potencial de automação varia entre indústrias dependendo da combinação de tipos de atividades



- "...26% da automação potencial poderá ser adotada até 2030, deslocando 1,1 Milhões de trabalhadores".

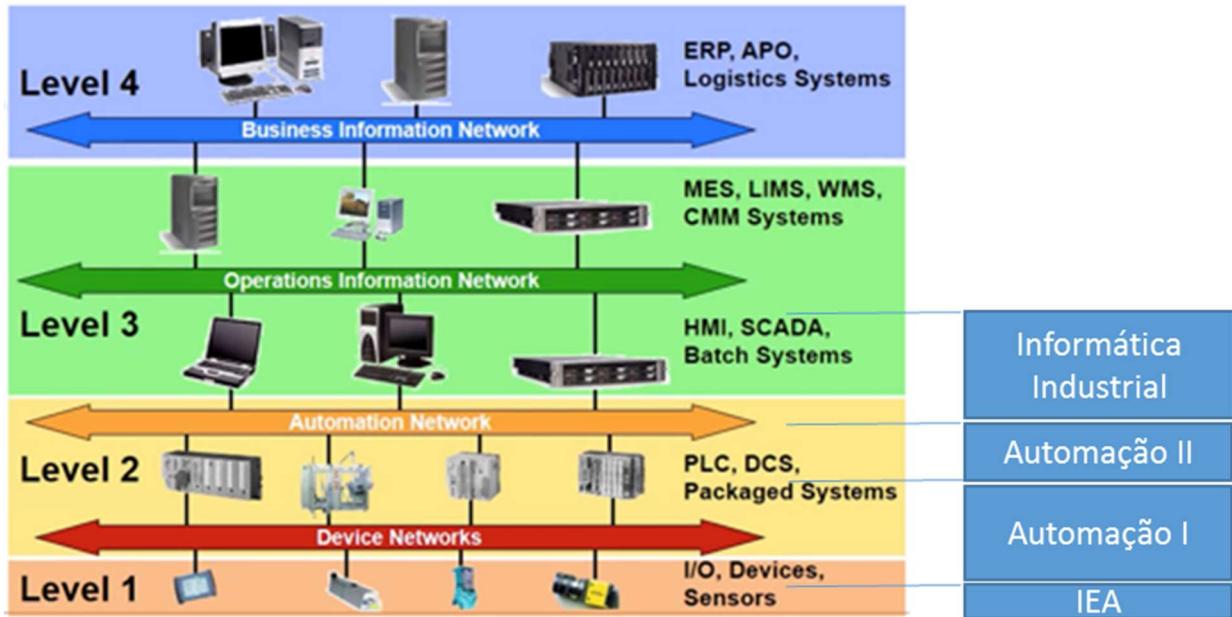
02 ..tornando redundantes cerca de 1,1 milhões de postos de trabalho em vários sectores, destacando-se a produção fabril e o comércio

Adoção de automação e deslocação, Cenário de adoção médio prazo (intermédio), 2030



1.1. Automação e Informática Industrial

A imagem seguinte, ilustra diversos sensores, equipamentos, protocolos de comunicação e programas habituais nas empresas industriais, desde o “chão de fábrica” até ao planeamento dos recursos e da produção na empresa.



Na base temos os equipamentos fabris: linhas de transferência, empilhadores, Automatic Guided vehicles (AGVs), centros de maquinagem, células de soldadura, fornos e muitos outros equipamentos de diversos tipos, com diversas interfaces *Human Machine Interfaces* (HMIs), muitos com quadros elétricos:

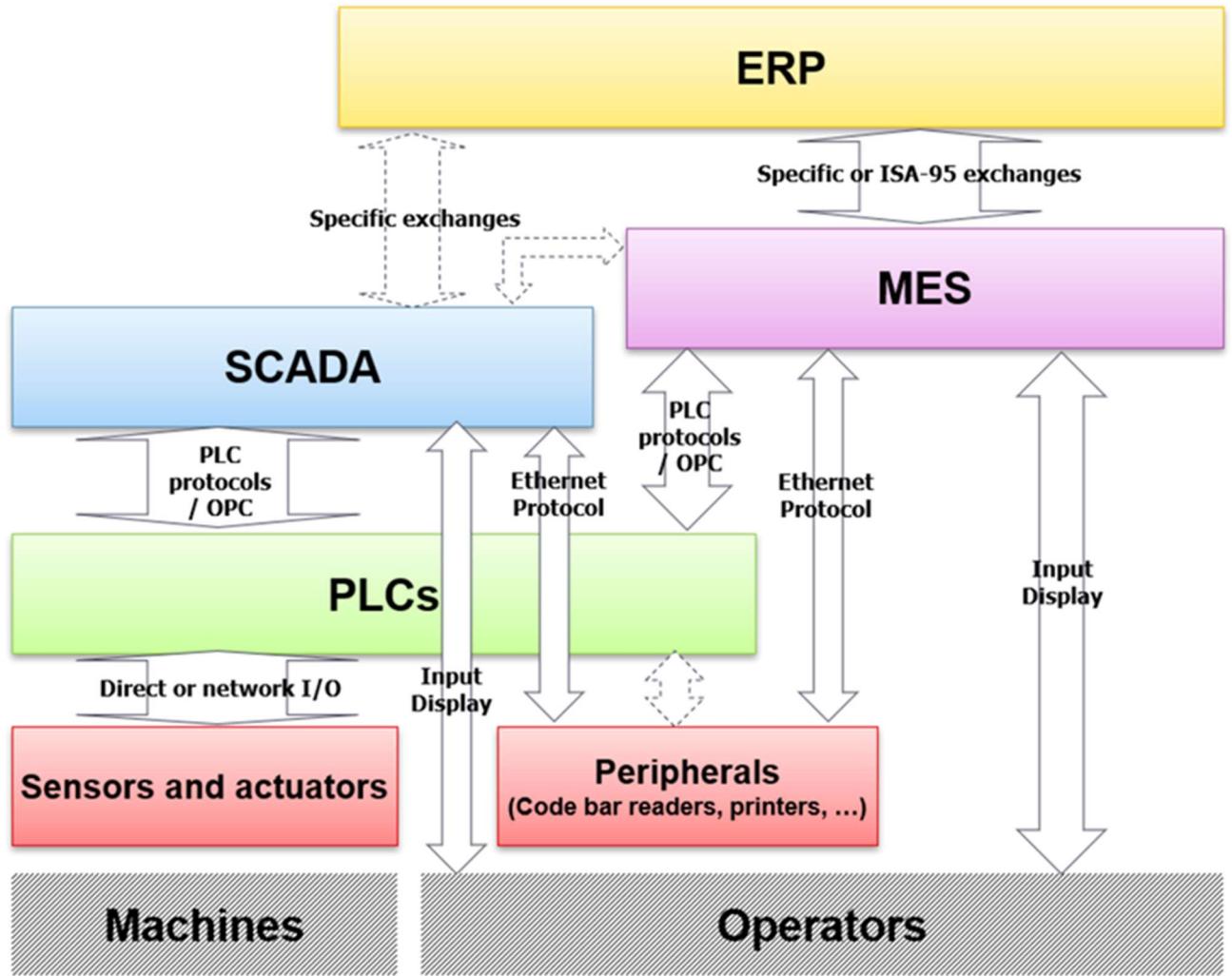
- com sensores e drives de potência que alimentam e permitem aos motores terem velocidade variável e inversão de sentido quando necessário (Level 1);
- com *Programmable Logic Controllers* (PLCs), *Computer Numerical Controllers* (CNCs), e outros equipamentos dedicados de controlo. (Level2)

Cada vez mais esses sensores, drives de potência, e controladores, possuem interfaces de comunicação em rede, como Ethercat, Sercos, Profibus, Profinet, IO-Link, ModBus, Rs485/Rs232, e muitas outras interfaces.

A camada nº3 da figura (Level 3), ilustra os programas de computador que comunicam com a instalação Fabril, desempenhando atividades de Supervisão, Controlo e Aquisição de Dados (SCADA), podendo também atuar como Interface Homem-Máquina (HMI) de um equipamento remoto, ou conjunto de equipamentos. Estes programas de computador podem também apoiar o *Planeamento da Produção e Controlo* (PP&C), ou atuar como um *Manufacturing Execution System* (MES) enviando e acompanhando a execução das ordens de fabrico na instalação fabril.

Essas aplicações de computador podem aceder à instalação fabril através dos diversos protocolos de comunicação referidos acima, e também através das especificações OPC-DA, OPC-AE, OPC-UA. Essas aplicações podem comunicar entre si, e com as aplicações de mais alto nível, por exemplo do tipo *Enterprise Resource Planning* (ERP), através de arquiteturas do tipo Broker MQTT, bases de dados partilhadas, WEB services baseados no *Simple Object Access Protocol* (SOAP), através de mensagens do tipo *Representational State Transition* (REST), dos *Remote Procedure Call* (RPC), entre outras.

Ao mais alto nível, também podem ser usados programas do tipo *Online Analytical Processing* (OLAP) para tratar grandes quantidades de dados e extrair informação útil.



1.2. Apresentação de algumas DPEs, em linha com Informática Industrial

Estudo de eficiência energética nos equipamentos de conformação, na Vista Alegre, SA

<https://www.youtube.com/watch?v=-WaLE47hIQ4&list=PLui9l53lO0XpeJ7aGSoDrY6lm-a49r9pm&index=7&t=0s>

O diagrama ilustra o fluxo de dados de um sistema de monitorização energética. No topo, uma interface de utilizador (Página Web e Relatórios) está conectada a um servidor. O servidor contém um Apache HTTP Server, uma Base de Dados, ficheiros (HTML, PHP, CSS) e um Node-RED com algoritmos. Um Broker MQTT conecta o servidor ao nível de Recolha de dados, que inclui um Micro-Controlador e um Contador. O Micro-Controlador recolhe dados de sensores.

Problema:

- Necessidade de monitorização do estado dos consumos energéticos (água, eletricidade, gás natural)

Solução:

- Sistema de monitorização de Energia
- Algoritmos de análise dos dados

Características:

- Capacidade de registo dos consumos e geração de relatórios
- Capacidade de visualização do estado em tempo real ou do histórico
- Avisos automáticos (SMS ou e-mail) sempre que os valores ultrapassem os limites definidos

Impacto Estimado:

Em 2019 o consumo de água foi de, aprox. 140 000m³
1% => 1 400m³, equivalente a cerca de 2 500€, segundo as tarifas da EPAL:
(<https://www.epal.pt/EPAL/menu/clientes/tarifário>, consultado em 26/02/2020)

Rastreabilidade de matérias primas e produtos, e a sua integração com o ERP da empresa, na Vista Alegre.

<https://www.youtube.com/watch?v=XdsV0Gg23Co&list=PLui9l53lO0XpeJ7aGSoDrY6lm-a49r9pm&index=9&t=0s>

O diagrama mostra a integração entre o sistema de rastreabilidade e o SAP ERP. Dados são enviados de dispositivos móveis e de colaboradores para um Node-RED, que os armazena em uma base de dados MySQL. O SAP ERP (SAP) é integrado ao sistema, permitindo a visualização de gráficos e dados para Engenheiros e Técnicos de Qualidade. O robô paletizador (etiquetado como (2)) também está integrado ao sistema, fornecendo dados para o rastreamento.

Problema: Registo e monitorização de dados e propriedades dos rolos de pasta utilizados para a produção de peças em porcelana e integração de um robô paletizador.

Solução pré implementação do robô (1):

- Aplicação online e em tempo real para os colaboradores submeterem as propriedades;
- Interface de monitorização dos dados recolhidos;

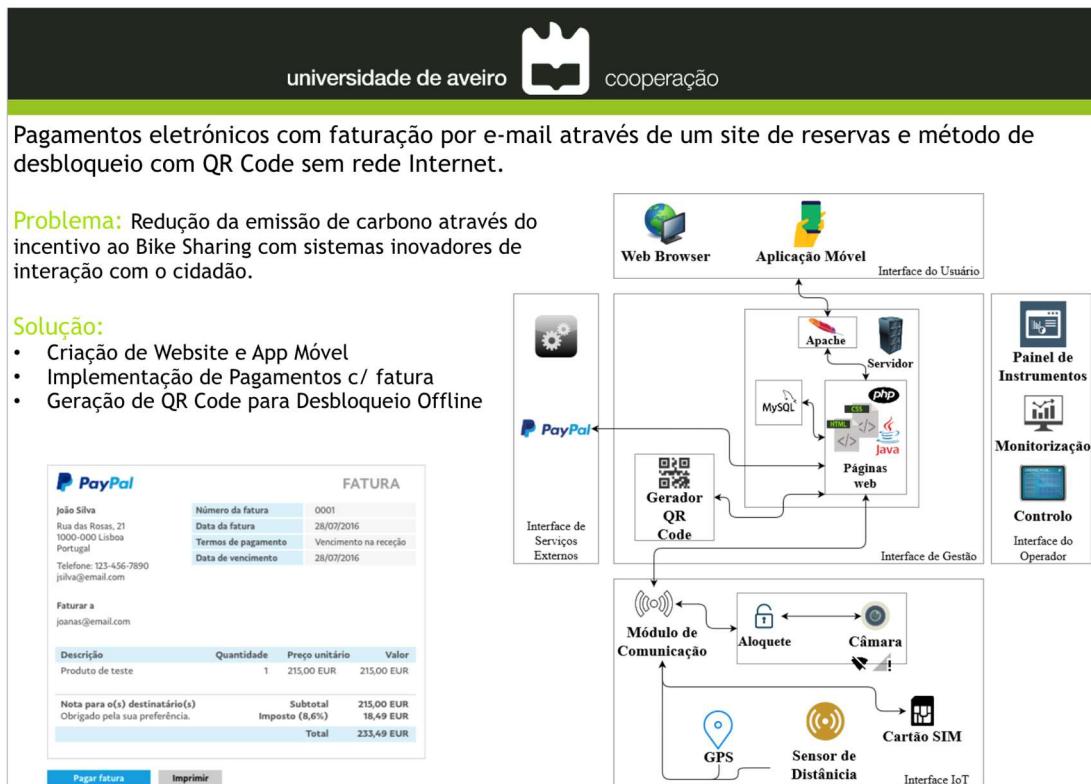
Solução pós implementação do robô (2):

- Implementação de um sistema de registo das propriedades autónomo ligado ao robô paletizador;

David Cabral
Orientador: Prof. José Paulo Santos
Projeto em colaboração com: Vista Alegre

Bike Sharing - Plataforma de reservas e pagamentos

https://www.youtube.com/watch?v=nZeP_bb4w6E&list=PLUi9l53lO0XpeJ7aGSoDrY6lm-a49r9pm&index=6&t=0s



Proposta de Sistema Automático de Rastreabilidade, na OLI

<https://www.youtube.com/watch?v=cvmHW7wzntM&list=PLUi9l53lO0XpeJ7aGSoDrY6lm-a49r9pm&index=10&t=0s>

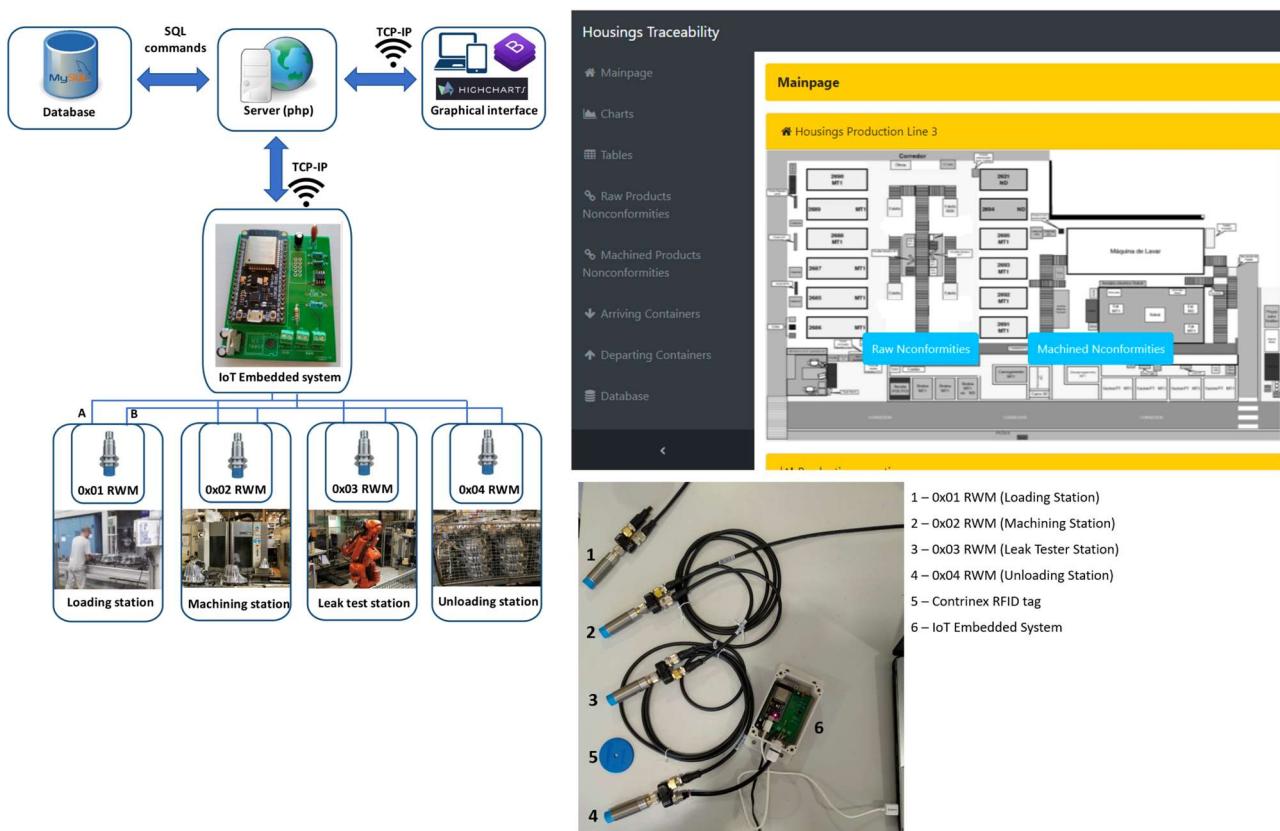
Gestão remota de dispositivos (Bosch)

<https://www.youtube.com/watch?v=BHDeMhrBzn4&list=PLUi9l53lO0XpeJ7aGSoDrY6lm-a49r9pm&index=11&t=0s>

Proposta de um sistema automático de rastreabilidade, na indústria 4.0. (na Renault)

<https://ria.ua.pt/handle/10773/26048>

Os sistemas de rastreabilidade são valiosos para as empresas fabricantes, permitindo manter um registo do histórico dos seus produtos. O surgimento da Indústria 4.0 avança ainda mais este conceito, integrando a Internet of Things na rastreabilidade ao criar objetos inteligentes capazes de comunicar pela Internet. A tecnologia RFID é a tecnologia com mais potencial na rastreabilidade devido às suas diversas vantagens quando comparada a tecnologias como os códigos de barra. A linha de produção de carters da Renault CACIA foi utilizada como um exemplo de um chão de fábrica no qual se aplica o sistema de rastreabilidade proposto. O sistema desta dissertação utiliza um microcontrolador de baixo custo, o ESP32, como o middleware responsável por processar as mensagens trocadas com os leitores RFID, sendo estes colocados em vários postos da linha de produção. As capacidades Wi-Fi do ESP32 são utilizadas para comunicar com um server que processa os dados e faz queries a uma base de dados MySQL. Esta base de dados foi normalizada e aplica-se especificamente à linha de carters, permitindo manter registos detalhados sobre cada carter produzido na fábrica. Foi desenvolvida uma interface gráfica web, cujas páginas mostram informação em tempo real sobre a linha, assim como informação sobre o histórico da produção. Estes serviços web podem ser acedidos por qualquer equipamento da IoT capaz de aceder ao server. A solução desenvolvida demonstra a viabilidade da integração de chips como o ESP32 em sistemas de RFID, permitindo que leitores e etiquetas RFID comuniquem pela Internet.



Planeamento da manutenção preditiva no contexto da indústria 4.0. (na OLI)

<http://hdl.handle.net/10773/26049> (UML ...)

Summary

The present work presents a solution that intends to help companies, in particular their maintenance departments, to initiate the implementation of a predictive maintenance program, through the development of a web platform.

The web platform must enable the real-time monitoring of the equipment's critical process parameters, which means that it was mandatory to develop all the hardware used to measure them. Additionally, the platform must be able to send real-time alerts, which warn the maintenance team of unusual operating conditions of the equipment.

This solution was tested under real and intensive operating conditions, around the clock.

This test exceeded expectations, since it measured the desired parameters correctly and didn't present any critical performance problems over a period of approximately four months.

This project was developed according to the needs of the company Oliveira & Irmão (OLI). The Mechanical Engineering Department has been collaborating with OLI since 2002 in several projects, including the publication of patents.

Keywords: Automation, embedded systems, Industry 4.0, Injection molding, Internet of things (IoT), Predictive maintenance, Web platform.

Description

The present work started with the determination of the injection molding machine's critical parameters. After this analysis, it was concluded that the parameters of interest for monitoring would be: temperature profile along the cylinder and the injection temperature and pressure. Accordingly, the necessary hardware was developed for the collection of these parameters and, subsequent storage in the created database. The developed hardware consists of a microcontroller that can interpret the values collected by the sensors and store them in the database through a wireless network.

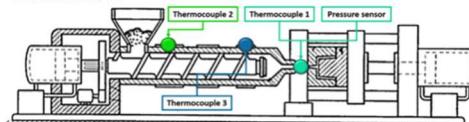


Fig 1 / Sensor location in the injection molding machine Maico Saving 260.

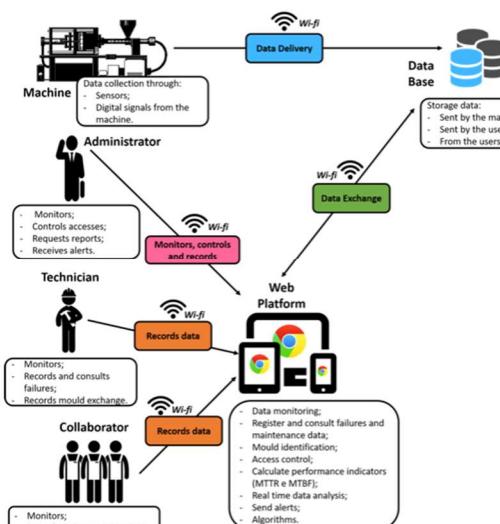


Fig 2 / Architecture of the proposed system.

The web platform was developed with the purpose of enabling the user to monitor, anywhere and in real time the parameters of interest. In addition, the platform controls the measured temperatures automatically, sending a warning email to the maintenance team when the temperatures go beyond the defined limits. It's important to note that an algorithm for the detection of lubricant oil leakage was also developed and implemented. Several tests were performed to verify the platform's robustness and efficiency, which, in the end, proved to be useful and promising for the implementation of a predictive maintenance plan.

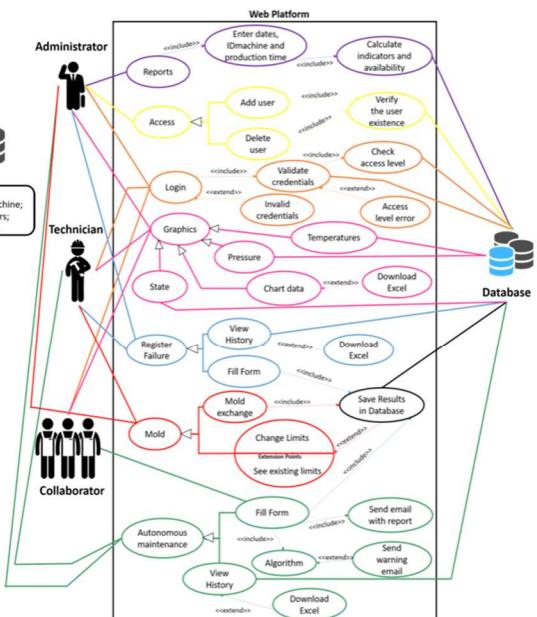
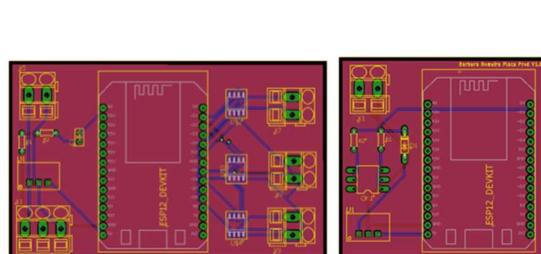


Fig 3 / Use Case diagram (in UML) for the developed platform.



OLI

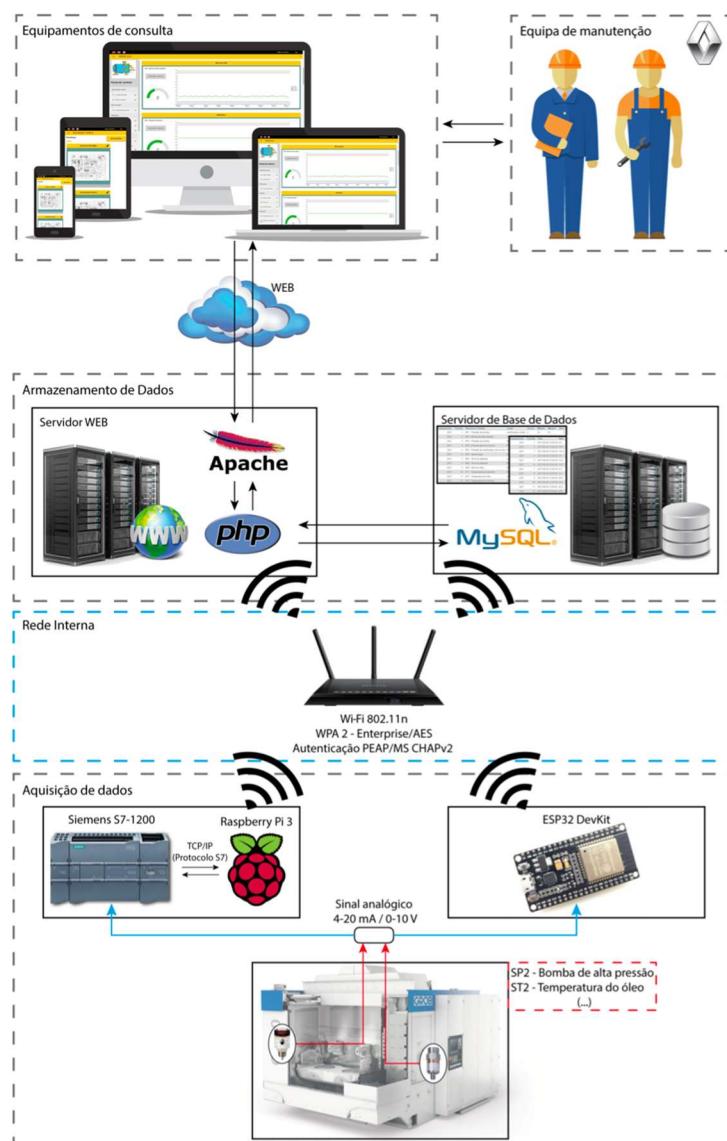
Machine N°91



Uma solução para a Manutenção Industrial, na Indústria 4.0. (na Renault)

<http://hdl.handle.net/10773/23677>

Atualmente, a manutenção industrial tem um papel estratégico fundamental nas organizações, uma vez que a sua gestão tem um impacto direto sobre as ações e proveitos retirados pelas mesmas. Deste modo, torna-se necessário dotar as organizações de mais e melhores mecanismos de monitorização e previsão do estado dos seus equipamentos, tornando possível o melhor planeamento das intervenções, a redução drásticas de custos associados a danos e/ou paragens e o aumento da eficiência do equipamento pela utilização de condições ótimas de funcionamento. Nesta linha, a manutenção preditiva permite a previsão de falhas antes destas ocorrem, tornando possível a diminuição do impacto dessa mesma falha no equipamento através de uma intervenção atempada e prematura. Neste relatório de projeto foi desenvolvida uma ferramenta de monitorização de equipamentos, com a Renault Cacia S.A., capaz de fornecer informações intuitivas aos operadores e responsáveis de manutenção sobre o estado dos mesmos. O objetivo passa por evitar paragens e/ou avarias por falta de uma correta supervisão, criando também um histórico de dados recolhidos que permite fazer a análise posterior dos mesmos. Foi desenvolvido ainda hardware para auxílio na implementação e utilização da ferramenta. Este trabalho serve de complemento a um outro trabalho responsável pela aquisição, processamento e envio de dados dos equipamentos. Os resultados obtidos foram congruentes com o esperado, uma vez que os objetivos pretendidos foram alcançados. Estes foram ainda demonstrados consecutivamente aos membros da fábrica Renault Cacia S.A., sendo que os comentários tecidos por esses membros demonstram que os objetivos alcançados encontram-se de acordo com as expectativas da empresa. O sistema desenvolvido encontrou-se a funcionar na totalidade nas instalações da fábrica, cumprindo todos os requisitos impostos. Refere-se ainda que o trabalho apresentado neste documento foi demonstrado ao diretor da fábrica Renault Cacia S.A., Juan Melgosa, a equipas de gestão europeias do Grupo Renault, ao chefe do projeto de hiper-competitividade no departamento de engenharia, manutenção e performance dos meios industriais no Grupo Renault, Stephane Gregoire, e ao vice-presidente de competitividade do Grupo Renault, Thierry Bolloré.



Algoritmos para a manutenção preditiva, na Indústria 4.0. (na empresa Navigator)

<https://ria.ua.pt/handle/10773/28536>

No âmbito da Indústria 4.0, pretende-se o desenvolvimento de algoritmos que permitam realizar a manutenção preditiva e apoiar na tomada de decisões face à calendarização de uma paragem de produção para reparações. Desta forma o planeamento de uma paragem será mais preciso, diminuindo custos, evitando a aquisição de material/equipamentos com entrega imediata (valor acrescido) e o overstock em armazém. Este projeto contempla um estudo sobre os vários indicadores que contribuem para a identificação de falha de rolamentos, como por exemplo a temperatura e as frequências inerentes aos constituintes do rolamento. Tem também uma análise de métodos preditivos focada em redes neurais artificiais, particularmente num algoritmo chamado de Multi-layer Perceptron Classifier. Este algoritmo é aplicado sobre vibrações sob a forma de frequência e medições de temperatura. A aceleração e a temperatura, são recebidas em tempo real de pontos estratégicos da máquina, e que demonstrem o estado atual de degradação dos rolamentos. Estas resultam do processamento de medições de aceleração com o algoritmo Fast computation algorithm for discrete Fourier Transform sendo as frequências e temperatura submetidas ao modelo que indica qual o tempo de vida que a máquina tem. Foram criadas duas rotinas de atualização do modelo de aprendizagem, uma que se repete semanalmente e outra que ocorre de aproximadamente de 16 em 16 horas. Por fim são apresentados vários resultados que comprovam a eficiência do algoritmo de machine-learning.

Automatic booking system with smart door lock and PayPal payment (“Smart Locker”)

<https://youtu.be/YemdMfx-o8c>

Monitorização e controlo remoto de sistemas de rega agrícola.

<http://hdl.handle.net/10773/23396>

Desenvolvimento de módulos para aquisição de sinais para a Indústria 4.0. (na Renault)

<http://hdl.handle.net/10773/23679>

1.3. Apresentação de algumas colaborações

SIGIP – Sistema Integrado de Gestão Inteligente de Produção (SI I&DT)

O projeto SIGIP visa o desenvolvimento de um sistema integrado de gestão inteligente de produção, configurando-se como um sistema baseado numa plataforma preditiva e integrada para otimização da produção, tendo como utilizador final a Indústria cerâmica. Enquanto sistema de gestão preditivo, os desenvolvimentos propostos assentam em algoritmos preditivos direcionados para a deteção de situações anómalas no sistema produtivo. Destes desenvolvimentos resultará um sistema integrado num módulo MES (Manufacturing Execution System), com algoritmos preditivos que permitirão a monitorização e correção de parâmetros, permitindo uma atuação direta sobre os equipamentos. Trata-se assim de um Sistema de manufatura preditiva, programado com inteligência que permita estimar sua própria condição, detetar a presença de falhas ou anomalias, inferir eventos futuros de falhas e diagnosticar possíveis causa dos problemas, permitindo uma maior eficiência dos processos produtivos desta indústria e a criação de valor pela maior disponibilidade dos meios produtivos.

INDÚSTRIA 4. 0 – AUGMENTED HUMANITY

PPS1 - Ergonomics and Robotics

PPS2 - Big Data and Predictive Analytics for i4.0

PPS3 - Industrial Internet of Things and connectivity

PPS4 - Artificial Vision & Augmented Reality

PPS5 - HR 4.0: Tools for better and healthier workers in i4.0 environments



CAPÍTULO

2

2. PROCESSOS DE NEGÓCIO E DE FABRICO

JPSantos

2023

Os processos de negócio e de fabrico podem ser entendidos como um conjunto de atividades paralelas e interdependentes, que concorrem de uma forma integrada e contínua para a consecução de um objetivo ou resultado comum. De acordo com a arquitetura de referência [CIMOSA'93], um *processo* caracteriza-se por um conjunto de *atividades* desenvolvidas na empresa, de acordo com um conjunto de *regras de procedimento*, regras essas inerentes ao negócio e à especificidade dos recursos da empresa. Por sua vez, uma *atividade* é composta por uma ou várias sequências de *operações funcionais*, entendendo-se por *operação funcional* uma operação elementar executada por um recurso da empresa.

Nas empresas industriais os *processos de negócio e de fabrico* compreendem atividades que vão desde a gestão de topo até à produção na instalação fabril. Os processos de negócio compreendem atividades de contabilidade, finanças, conceção e projeto de produtos, bem como de planeamento. Os processos de fabrico compreendem sobretudo atividades que decorrem ao nível da instalação fabril, ligadas às operações de fabrico propriamente ditas, i.e. transporte de material, transformação, montagem, armazenamento, preparação (“setup”), testes de qualidade, etc.

As empresas têm por objectivo o lucro, fornecendo para isso aos seus clientes serviços ou produtos por forma a gerarem valor acrescentado. Interessa produzir ao menor custo, o mais rapidamente possível, e com elevada qualidade os produtos encomendados de acordo com as especificações dos clientes. O aumento da concorrência, do nível de exigência dos consumidores e da rapidez com que as especificações dos produtos se alteram, têm todavia obrigado as empresas a encetar profundas transformações, obrigando-as a uma *reorganização constante dos seus processos de negócio e de fabrico*, bem como ao *recurso constante a novas tecnologias* capazes de suportar e integrar esses processos. O contexto atual de elevada competitividade tem obrigado as empresas a eliminar progressivamente todas as formas de desperdício de que resultem atrasos nas entregas, ou diminuição

de qualidade dos produtos, obrigando a uma reorganização constante dos processos de negócio e de fabrico.

A organização do trabalho nas empresas tem merecido a atenção de diversos autores, ao longo do tempo. Já em 1776 Adam Smith no seu livro “The Wealth of Nations” defendia a teoria de que o trabalho deveria ser decomposto em tantas tarefas elementares quanto possível, especializando-se cada operário na execução de uma delas, permitindo assim um aumento de produtividade nunca visto. Ao contrário das pequenas empresas artesanais, onde em geral, um só operário/dono executa todas ou quase todas as tarefas necessárias à produção, as empresas de pequena e grande dimensão, seguindo os princípios de Adam Smith, têm organizado a produção em *áreas funcionais* a que, no caso das empresas industriais, podemos chamar de *centros de produção*. Cada centro de produção agrupa um conjunto de máquinas, dispostas na proximidade física umas da outras, sendo controladas por um conjunto de operários, especializados na realização de um conjunto reduzido de operações que executam, de forma mais ou menos repetitiva sobre grandes lotes de uma mesma peça. A aplicação destes princípios à indústria metalomecânica, por exemplo, levou à construção de fábricas que se caracterizam pela disposição física de máquinas afins na proximidade umas das outras, ficando por exemplo todas as prensas de corte (ou guilhotinas) numa zona da instalação fabril, passando-se o mesmo com as máquinas quinadeiras, puncionadeiras, de rebarbar, de polir, tornos, soldadura, e outras, podendo também existir uma área reservada para a montagens de componentes.

Em cada centro de produção existe habitualmente um chefe de centro, responsável pela sua operação. Esta organização do trabalho leva no entanto à necessidade de encarregados responsáveis pela coordenação e supervisão dos vários centros de produção, assegurando a conformidade dos produtos às especificações do cliente.

Nas empresas de maior dimensão é necessária a criação de chefias de nível superior para coordenar os encarregados. Por sua vez, estas chefias são coordenadas por chefias de nível superior, criando-se desta forma uma cadeia de comando hierárquica. Está assim explicada a organização dos processos de negócio e de fabrico numa estrutura hierárquica. Estes sistemas tem habitualmente dois, três ou quatro níveis de controlo (figura 1.1 – [Rembold' 85]).

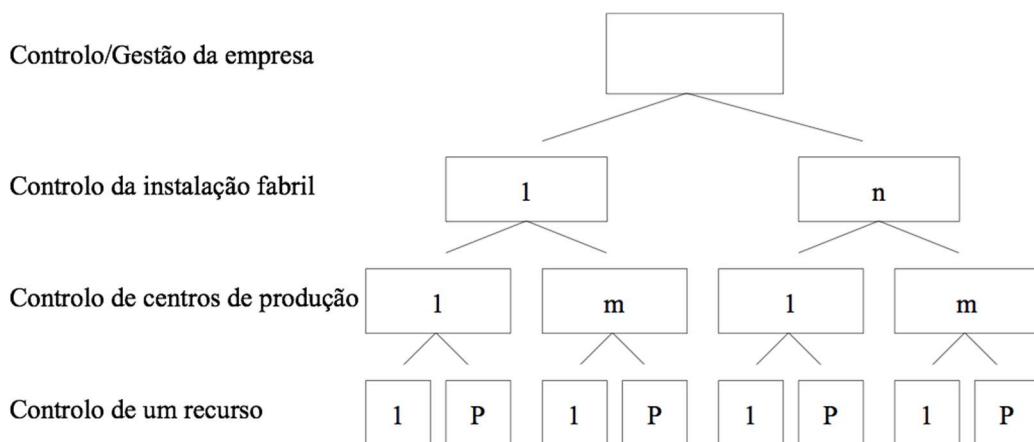


figura 1.1: Estrutura de controlo hierárquico

O 4º nível, “Controlo da Empresa/Gestão”, compreende funções de supervisão, gestão e planeamento do negócio, nomeadamente respeitantes a previsões, custos, publicidade, vendas, decisões estratégicas, planeamento dos processos (plano de produção) e planeamento/projeto dos produtos.

O 3º nível, “Controlo da instalação fabril”, compreende as funções de gestão da instalação fabril, nomeadamente a gestão quotidiana da execução das ordens de fabrico, e a supervisão das operações fabris. A função mais importante deste nível é o controlo da execução dos planos da produção. Este nível deve ainda optimizar os tempos e custos de produção, gerindo a atribuição de tarefas a cada recurso ou centro de produção.

O 2º nível, “Controlo de centros de produção”, compreende o controlo dos recursos fabris e desta forma dos processos produtivos. Compete-lhe supervisionar vários processos de fabrico controlando diversos recursos como máquinas ferramenta com comando numérico, sistemas de transporte de materiais, linhas de montagem e outros.

O 1º nível, “Controlo de um recurso”, como o nome indica assegura o controlo local de cada recurso de produção, podendo o controlo do recursos ser executado por um empregado da empresa ou por um dispositivo de controlo automático, integrado fisicamente no recurso a controlar. Alguns dispositivos de controlo automático são programáveis permitindo executar diretamente programas de controlo. Com o aumento do número de pessoas envolvidas nas várias tarefas da empresa e a crescente complexidade dos produtos a fabricar, surgiu a necessidade de criação de especificações escritas que descrevessem as diversas peças/produtos a produzir (desenho técnico, tolerâncias, materiais a usar), os processos de produção (planeamento prévio das atividades de produção) e ainda as diversas atividades administrativas ligadas ao fabrico e ao negócio (compras de material, gestão de armazéns, emissão de facturas a clientes, etc).

Esta organização do trabalho baseada na decomposição dos processos de negócio e de fabrico em tarefas elementares, coordenadas por uma cadeia hierárquica de comando, permite a utilização de mão de obra não especializada e portanto barata.

Neste tipo de organização das empresas (figura 1.1) os empregados são “encorajados” a seguir um conjunto de regras bem definidas, e a desprezar a sua iniciativa individual. Em suma, são encorajados a não questionar os métodos estabelecidos e a actuar de acordo com um conjunto de *regras de procedimento*.

A necessária organização, neste contexto, da empresa em áreas funcionais ou departamentos cria vários tipos de desperdício maioritariamente associados a dificuldades de integração: dificuldades de comunicação intra e inter-áreas, lacunas, redundâncias, descontinuidades nos fluxos de informação e material, etc. Esta situação limita, naturalmente, o desempenho da empresa em termos, nomeadamente dos tempos de resposta e custos de produção.

A agressividade da concorrência e as constantes alterações do mercado têm obrigado as empresas a uma reorganização constante dos seus processos de negócio e de fabrico, bem como ao recurso constante a novas tecnologias capazes de suportar e integrar esses processos com o objectivo de minorar, ou no limite, eliminar estes problemas. Para aumentar a sua competitividade, as empresas têm ao longo do tempo tentado melhorar três aspectos fundamentais: a sua estrutura organizacional, o suporte tecnológico às suas atividades e à sua estrutura organizacional, e ainda a formação e especialização dos seus recursos humanos, observando-se o seguinte:

A experiência, formação e perícia dos recursos humanos é um aspecto de crucial importância, mas que não será aqui abordado por não ser relevante no contexto deste trabalho.

No que diz respeito à estrutura organizacional das empresas, diversos princípios e técnicas de organização industrial têm sido propostos ao longo do tempo como forma de organizar as suas atividades, nomeadamente Just in Time – JIT, Total Quality Management – TQM, Total Productive Maintenance – TPM, Outsourcing, reengenharia de processos, etc. Mais tarde surgiram arquitecturas de suporte à integração de atividades, procurando ajudar a organizar e integrar as atividades da empresa

de acordo com diferentes estratégias e modelos organizacionais. Alguns exemplos destas arquiteturas serão apresentadas no capítulo seguinte, enquadrando o trabalho que é objeto desta tese.

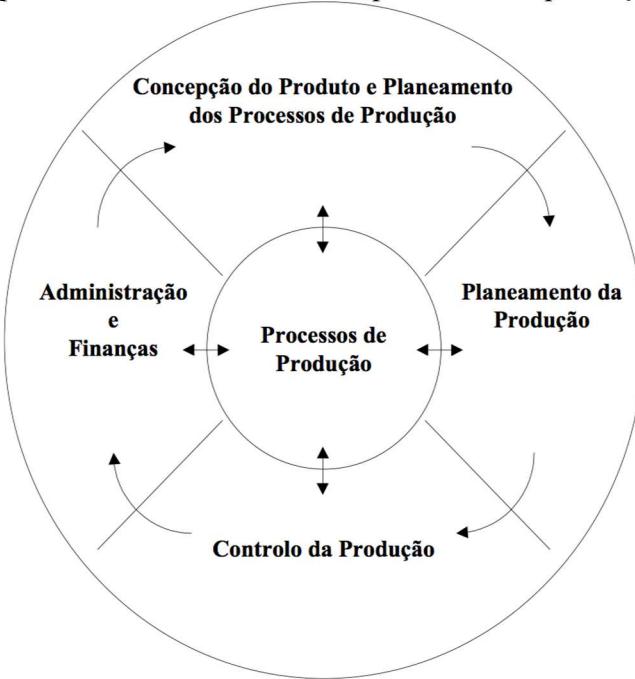
O recurso constante a novas tecnologias capazes de suportar e integrar os processos de negócio e de fabrico tem sido uma estratégia usada pelas empresas como forma de melhorar a sua competitividade. É paradigmático a aplicação das novas Tecnologias da Informação (TI) aos processos de negócio e de fabrico, nomeadamente na implementação dos sistemas de produção integrado por computador (CIM). Os sistemas CIM procuram aumentar a produtividade, qualidade e rapidez de resposta da empresa às solicitações do mercado, recorrendo aos computadores, às redes de comunicação e a outras soluções tecnológicas, para suportar e agilizar as atividades da empresa e a integração dos seus fluxos de informação e de material.

Este capítulo abordará em particular a última área, começando por apresentar, na secção 1.1, uma visão geral das atividades da empresa. Estas atividades podem ser enquadradas em dois grandes ambientes: o ambiente de engenharia (secção 1.1.1) e o ambiente de produção (secção 1.1.2). O ambiente de produção compreende as atividades de produção na instalação fabril, atividades essas que envolvem a transformação física das matérias primas em produtos acabados. Precedendo as atividades de produção efetiva na instalação fabril existem ainda um conjunto de atividades de concepção prévia dos produtos, planeamento e escalonamento da produção, bem como um conjunto de outras atividades que se podem enquadrar no chamado ambiente de engenharia da empresa.

Na secção 1.2 será apresentada a evolução das atividades nas empresas de fabrico proporcionada e acelerada pela sua *informatização e automatização*. Os computadores, os automatismos programáveis, e as redes de comunicação de dados, vieram de facto revolucionar a produtividade dos recursos e sistemas produtivos. No ambiente de engenharia (secção 1.2.1), esta evolução tem-se caracterizado pelo recurso aos computadores (Computer Aided), permitindo melhorar e integrar as atividades de concepção, projeto, planeamento e controlo. No ambiente de produção (secção 1.2.2), esta evolução tem-se manifestado pelo recurso à automatização/informatização dos recursos de produção, transformando-os em recursos programáveis (centros de maquinagem, autómatos programáveis, robôs, sistemas de transporte e manipulação de materiais, células, sistemas flexíveis de produção, etc.) e dessa forma melhorando significativamente os processos tecnológicos de produção. Estes recursos têm sido dotados de capacidade de comunicação em rede como forma de suportar os fluxos de informação ao nível da instalação fabril permitindo também sincronizar e coordenar as tarefas desempenhadas por cada um.

2.1. Atividade de uma empresa de fabrico

A figura [Mesquita' 97] apresenta os principais tipos de atividades de uma empresa de fabrico: atividades de *concepção*, *planeamento*, e *controlo* dos processos de produção na instalação fabril.



Todas estas atividades visam, em última análise, a transformação das matérias primas e componentes em produtos acabados da forma mais eficiente e eficaz possível. Por *processos de produção* entendem-se sequências de operações conducentes a uma efectiva alteração física das matérias primas e componentes em produtos acabados (processos de corte, soldadura, montagem, etc. Estes processos são executados pelos recursos de produção (recursos e pessoal) da instalação fabril. No entanto, antes dos processos de produção terem lugar na instalação fabril impõem-se o seu planeamento prévio. Existem basicamente dois tipos de planeamento: o *planeamento dos processos de produção* e o *planeamento da produção*.

Por *planeamento dos processos de produção* entende-se: a definição do tipo de processos e recursos a usar, das ferramentas, das sequências de operações, dos programas de controlo numérico, dos materiais que serão necessários e respectivas especificações (desenhos, materiais, tolerâncias) definidas durante a concepção do produto. O planeamento dos processos inclui também a previsão dos tempos envolvidos em cada fase do processo.

Por *planeamento da produção*: entende-se a coordenação e escalonamento dos vários processos de produção, bem como o planeamento e a partilha dos recursos da empresa que serão usados durante a produção, tendo em conta a necessidade de produzir em simultâneo vários tipos de produtos, encomendados pelos diversos clientes.

O planeamento da produção tem a seu cargo, nomeadamente, a definição das ordens de fabrico, a gestão de materiais, o plano director de produção e a estimativa de custos e tempos de produção na instalação fabril.

O *controlo da produção*: consiste no controlo e supervisão dos processos de produção, entendida aqui como fabrico, montagem, etc., tendo como base os planos de produção, as ordens de fabrico, os programas de Comando Numérico (CN), etc., definidos durante as fases de planeamento dos processos e planeamento da produção.

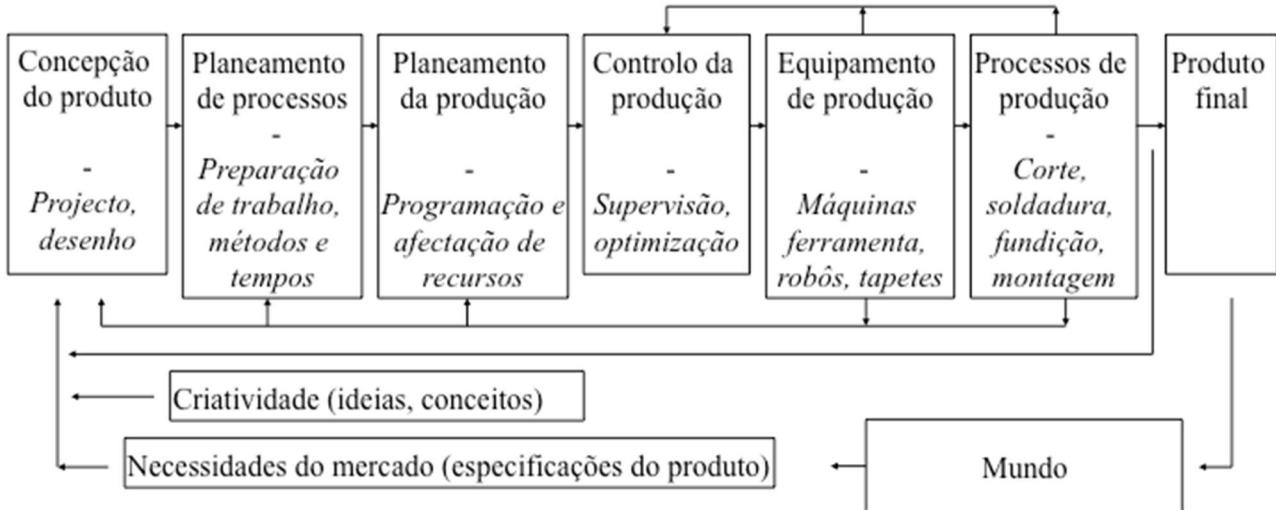


figura 1.2: Atividades de uma empresa de fabrico – introdução

Antecedendo ainda o planeamento dos processos de produção, é muitas vezes necessário conceber e projectar o produto de forma autónoma ou em colaboração (“co-design”) com clientes e/ou fornecedores. Esta fase inclui a elaboração do seu desenho técnico, com cotas, tolerâncias, materiais a usar, etc., atividades essas denominadas genericamente por *concepção e projeto do produto*.

As atividades de uma empresa de fabrico [Rembold’ 93] são bastante diversificadas. Pesquisas de mercado, contabilidade, planeamento estratégico de médio e longo prazo, e outras, que por não serem relevantes no contexto deste trabalho, não serão aqui apresentadas.

A figura 1.3 [Rembold’ 93] apresenta com maior detalhe, e de forma estruturada, as atividades da empresa mais relevantes para este trabalho. Podemos observar que cada bloco funcional agrupa atividades do mesmo tipo. Cada bloco funcional poderá corresponder a departamentos ou outras unidades organizacionais da empresa. A zona mais à direita da figura (ambiente de produção) mostra os fluxos das matérias primas e componentes desde a sua recepção até ao envio dos produtos acabados, passando pelos processos tecnológicos de produção. Estes fluxos de material são controlados pelos fluxos de informação criados nas unidades organizacionais do lado esquerdo da figura (ambiente de engenharia).

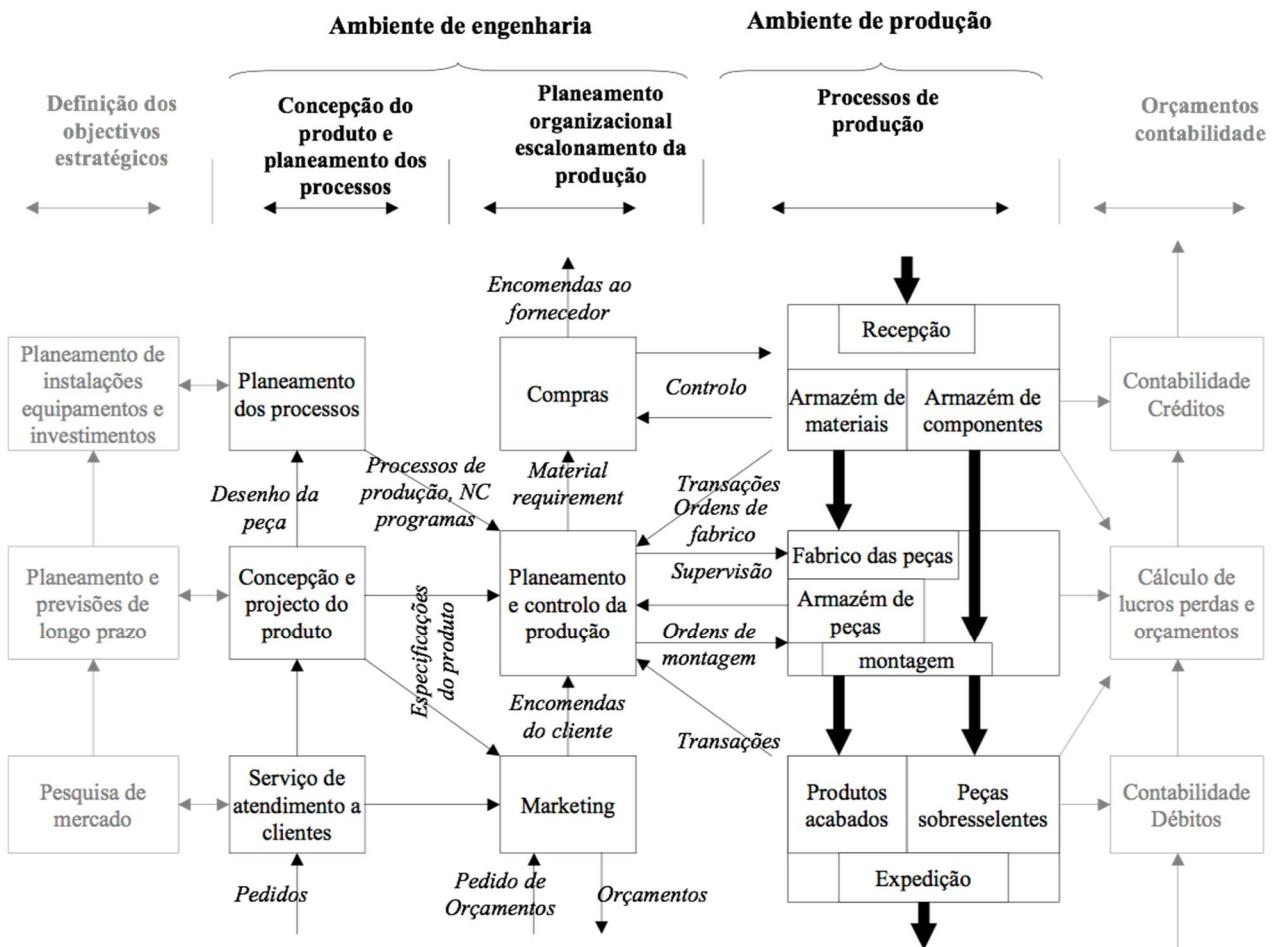


figura 1.3: Atividades de uma empresa de fabrico – detalhe

2.1.1. Ambiente de Engenharia: Gestão, Concepção, Planeamento

O ambiente referido de forma muito abrangente como de engenharia (figura 2.3 - lado esquerdo) comprehende múltiplas atividades: recepção/negociação de encomendas, concepção e projecto do produto de acordo com as especificações do cliente, atividades de planeamento prévio, quer dos processos de produção quer da produção, etc. Em suma, poderá dizer-se que as atividades de engenharia comprehendem todas as atividades que antecedem o início da produção fabril .

A encomenda do cliente pode ser recebida com menor ou maior rigor técnico e informativo (ex: lista de materiais, tolerâncias), podendo ser acompanhada com os desenhos técnicos das peças a produzir.

A unidade organizacional onde se leva a cabo a *concepção e projecto do produto* é responsável por produzir os desenhos técnicos, no caso de não terem sido já elaborados pelo cliente, por definir a lista de materiais, os componentes da peça a produzir, as tolerâncias, e os recursos que deverão ser usados. Esta unidade pode, se necessário, criar um protótipo funcional que pode ser utilizado para testes de aceitação e negociação com o cliente.

A unidade organizacional de *planeamento dos processos de produção* é responsável pela definição do Plano de Produção (PP) com as sequências das operações de maquinagem, montagem, etc., a que a matéria prima deverá ser sujeita em cada máquina ou recurso a fim de se obter o produto final pretendido. O PP é elaborado a partir dos desenhos técnicos da peça. Esta unidade pode ainda

desenvolver os programas para as máquinas ferramenta com comando numérico, robôs e outros automatismos programáveis.

Quando se inicia a produção de um novo componente ou produto é necessário reescalonar e coordenar a sua produção com a produção de outros componentes ou produtos que estejam a ser produzidos em simultâneo, sendo por isso necessário partilhar os recursos de produção. A unidade organizacional que inclui as áreas de *escalonamento das ordens de produção, supervisão e controlo da produção* tem a responsabilidade de “detalhar”, “escalonar” e “gerir” os tempos de produção de cada recurso, de forma a satisfazer as especificações pretendidas, gerando as ordens de fabrico adequadas. No que respeita à execução das operações de fabrico tem a função de decidir: que quantidades de produto, com que matérias primas, em que máquinas, com que operadores, com que ferramentas, e quando. A execução das ordens de fabrico depende dos próprios recursos de produção, da sua taxa de ocupação e dos processos tecnológicos envolvidos. Esta unidade deve ainda colaborar na previsão das datas de entrega, gestão dos atrasos e cálculo dos preços de produção.

Tal como mostra a figura, alguns componentes podem ser adquiridos a outras empresas e mais tarde integrados com os componentes produzidos na própria empresa, sendo da responsabilidade do departamento de *compras* o controlo da aquisição, recepção e armazenamento desses componentes, bem como da matéria prima em geral.

2.1.2. Ambiente de produção: Fabrico, Montagem, Teste

O ambiente de produção (figura 1.3 - lado direito) caracteriza-se pela transformação das matérias primas em produtos acabados. Ao conjunto de atividades e operações executadas pelos recursos de produção para transformar a matéria prima em produtos acabados, dá-se o nome de processos de produção. Estes processos obedecem às especificações elaboradas durante a concepção, projeto e planeamento. Os processos de produção incluem os processos tecnológicos de alteração de forma e propriedades físicas, químicas, etc., da matéria prima, os processos de aferição e controlo da qualidade e ainda atividades de transporte e armazenamento de matérias primas, componentes, produtos acabados e semi acabados.

Processos tecnológicos de alteração de forma

A figura 1.4 [Mesquita' 97] apresenta alguns processos tecnológicos que permitem alterar a forma dos materiais, nomeadamente os processos de corte, de ligação, e os processos de enformação.

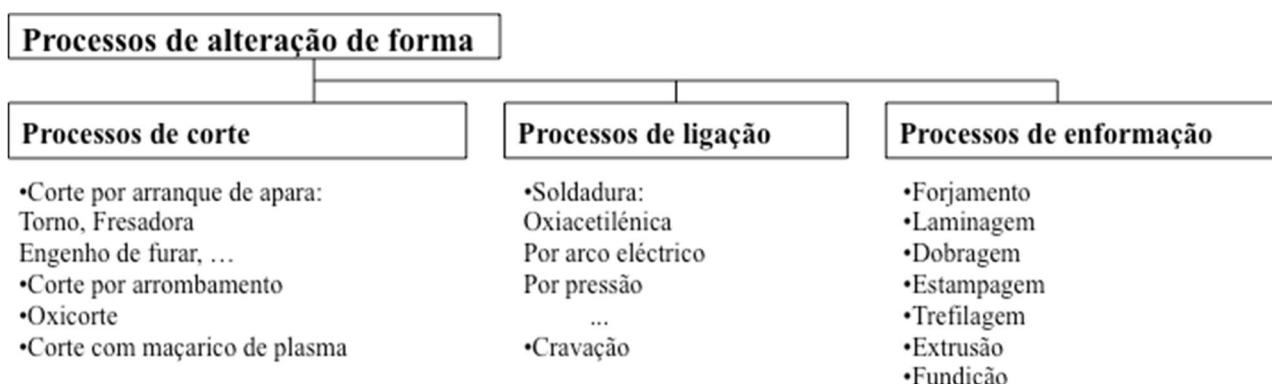


figura 1.4: Processos de alteração de forma

Os processos de corte permitem alterar a forma da matéria prima por remoção de material. O material é removido até se obter a forma pretendida; uma fresadora, por exemplo, permite executar um processo de alteração de forma da matéria prima por arranque de apara. Os processos de ligação permitem também alterar a forma das peças, mas neste caso por adição de material, por exemplo

soldando peças entre si. Os processos de enformação também permitem alterar a forma das peças sendo, neste caso, a alteração da forma conseguida por deformação plástica, ou por fusão e solidificação num molde.

Processos tecnológicos de alteração de propriedades

A figura 1.5 [Mesquita' 97] apresenta alguns exemplos de processos tecnológicos que permitem alterar as propriedades da matéria prima nomeadamente: a sua estrutura, a sua composição e o seu revestimento. Os processos de alteração de estrutura consistem em tratamentos térmicos ou mecânicos. Através destes processos é possível, por exemplo, alterar o estado de tensão interna ou as dimensões do grão do material sujeito a estes tratamentos.

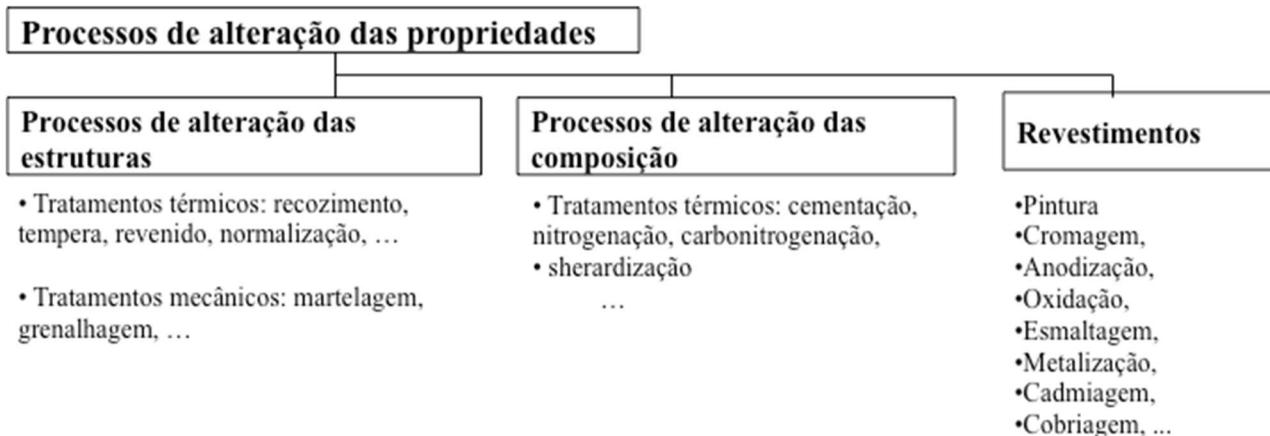


figura 1.5: Processos tecnológicos de alteração de propriedades

Os processos de alteração da composição permitem também alterar as propriedades do material sem no entanto alterar a sua forma. A alteração das propriedades consegue-se através de tratamentos térmicos, tais como a cementação, alterando a composição química da superfície do material. O revestimento de materiais com novos materiais de tipo diferente permite que as propriedades da superfície do material seja alterada.

No contexto deste trabalho são relevantes as atividades de planeamento e controlo dos processos tecnológicos de alteração de forma por corte, especificamente por arranque de apara, bem como as atividades de transporte, montagem e armazenamento.

2.2. Informatização e automatização das atividades da empresa

A aplicação das TI na produção mecânica tem sido a chave para o aumento da produtividade, qualidade e eficácia da produção. Nesta secção será feita uma breve descrição da evolução dos sistemas de fabrico, quer no que respeita ao seu ambiente de engenharia, quer ao ambiente de produção.

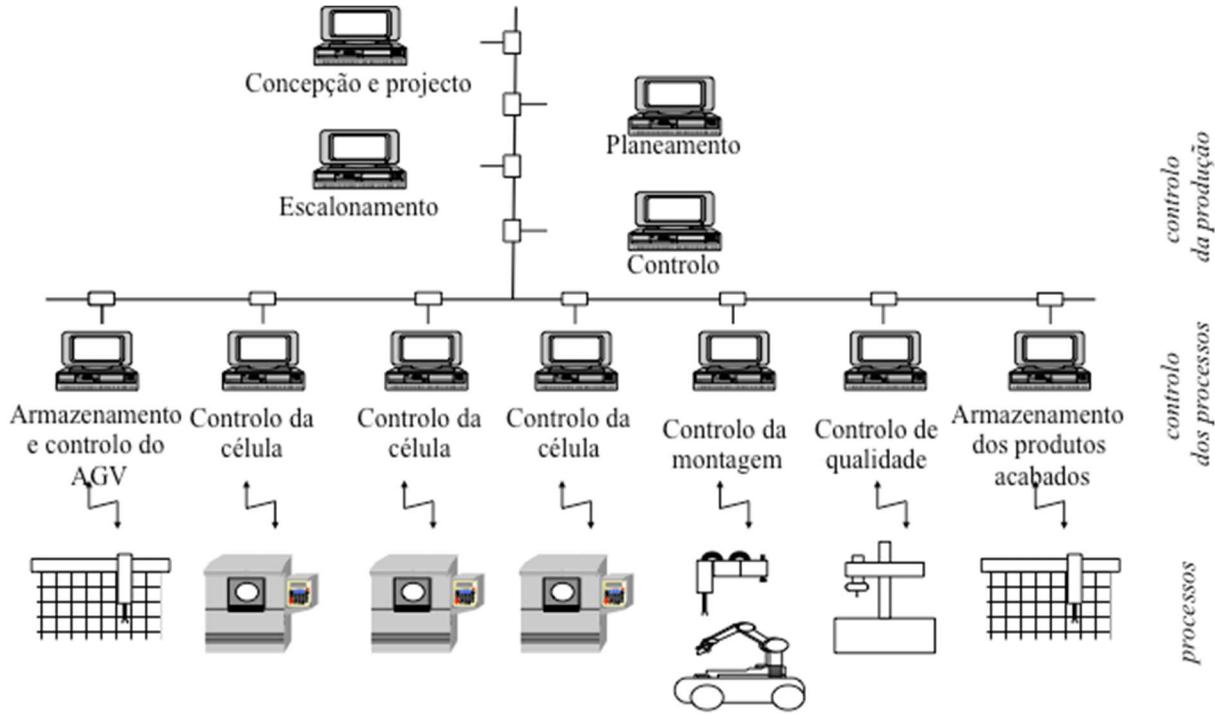


figura 1.6: Organização de uma empresa CIM

O controlo das atividades de produção usando computadores iniciou-se entre o final da década de 50 e o início da década de 60, com funções como a aquisição de dados, controlo da qualidade, supervisão das operações de fabrico e ainda controlo numérico de máquinas ferramenta. Os computadores, em conjunto com as redes de comunicação de dados, permitiram o controlo e supervisão em “tempo real” das operações de fabrico, recolhendo e processando (“on-line” e em “tempo real”) os fluxos de informação.

A informatização e automatização que se sucederam na década de setenta podem ser encaradas como um primeiro passo para o que nos anos oitenta e noventa se passou a designar por produção ou fabrico integrado por computador (CIM).

A figura 1.6 apresenta um exemplo simples de uma empresa em que algumas das atividades são suportadas e integradas recorrendo a computadores e redes de comunicação. Na parte de baixo da figura estão representados os recursos da instalação fabril (“shop-floor”): um armazém de matéria prima; três células de produção, cada uma constituída por um centro de maquinagem e um sistema automático de alimentação de peças, neste caso executando processos tecnológicos de alteração de forma por arranque de apara; um robô para a montagem de componentes maquinados na própria empresa e/ou componentes adquiridos ao exterior; uma estação de medição (controlo da qualidade); um armazém de produtos acabados; e um veículo guiado automaticamente (AGV).

A espinha dorsal do fluxo de materiais é, neste caso, assegurada pelo AGV que transporta as matérias primas ao longo da instalação fabril, desde o armazém de matéria prima (lado esquerdo da figura) até ao armazém de produtos acabados (lado direito da figura) passando pelos centros de maquinagem, montagem e qualidade.

Neste exemplo, as atividades de planeamento e controlo da produção são suportadas por aplicações informáticas baseadas em plataformas computacionais. Os recursos de produção são controlados a três níveis: controlados directamente por automatismos programáveis, ou computadores dedicados, instalados fisicamente nesses recursos. Controlados indirectamente, a partir de computadores que controlam cada centro de produção, cabendo a esses computadores a coordenação e sincronização dos recursos atribuídos ao seu centro respectivo. E ainda, um nível de controlo integrado de todos os centros de produção, cabendo a esse computador não só coordenar e sincronizar os vários centros de

produção mas também reescalonar as tarefas e os recursos a eles atribuídos, de acordo com as suas taxas de ocupação e tempos de produção.

Tanto os computadores de apoio às atividades de planeamento e controlo como os automatismos e computadores dedicados dos recursos de produção, encontram-se ligados entre si através de redes de comunicação de dados, partilha de bases de dados ou mesmo plataformas informáticas distribuídas, para dessa forma suportarem e agilizarem os fluxos de informação trocados entre si.

As decisões sobre a informatização e a automatização das atividades da empresa deverão todavia ser resultado de uma análise cuidadosa dos custos e benefícios inerentes.

A tabela 1.1 apresenta algumas soluções tecnológicas a adoptar consoante os processos da empresa que se pretende automatizar ou informatizar.

tabela 1.1 : Exemplos de soluções tecnológicas de suporte aos diferentes processos de produção e de engenharia

	Soluções tecnológicas
Processos de produção: fabrico, montagem, manipulação e armazenamento de materiais, inspecção e controlo da qualidade.	<ul style="list-style-type: none">. Robôs (de soldadura, pintura). <u>Autómatos Programáveis</u> (PLC- Programmable Logic Controllers). Recursos com controlo numérico.. Linhas de transferência (conveyors). Veículos Guiados Automaticamente (AGVs- Automated Guided Vehicles). Armazéns automáticos (ASRS – Automated Storaged and retrieval systems). Sistemas de visão por computador
Processos de negócio: concepção planeamento, projecto	<ul style="list-style-type: none">. CAE, CAD, CAM, CAP, CAQ, PP&C ,...

A informatização das atividades da empresa permitiu melhorar substancialmente a eficiência, a qualidade e a rapidez de execução de algumas atividades (figura 1.7), nomeadamente nas áreas de: projeto assistido por computador (CAD), planeamento dos processos da produção (CAPP), planeamento e controlo temporal da produção (PP&C) na instalação fabril, e ainda verificação da qualidade dos produtos (CAQ) . Na base da pirâmide está a instalação fabril e o controlo dos recursos de fabrico (CAM).

A filosofia da produção integrada (CIM) visa implementar um sistema de fabrico hierarquizado, controlado por computador, no qual os fluxos de informação gerados pelas atividades apoiadas por sistemas de CAD, CAP, CAM, CAQ e PP&C e os fluxos de material na instalação fabril se encontram integrados. Para isso, é imprescindível integrar os fluxos de material e de informação da empresa, o que exige, ao nível da instalação fabril, recursos de produção programáveis e capazes de comunicar através de redes de comunicação.

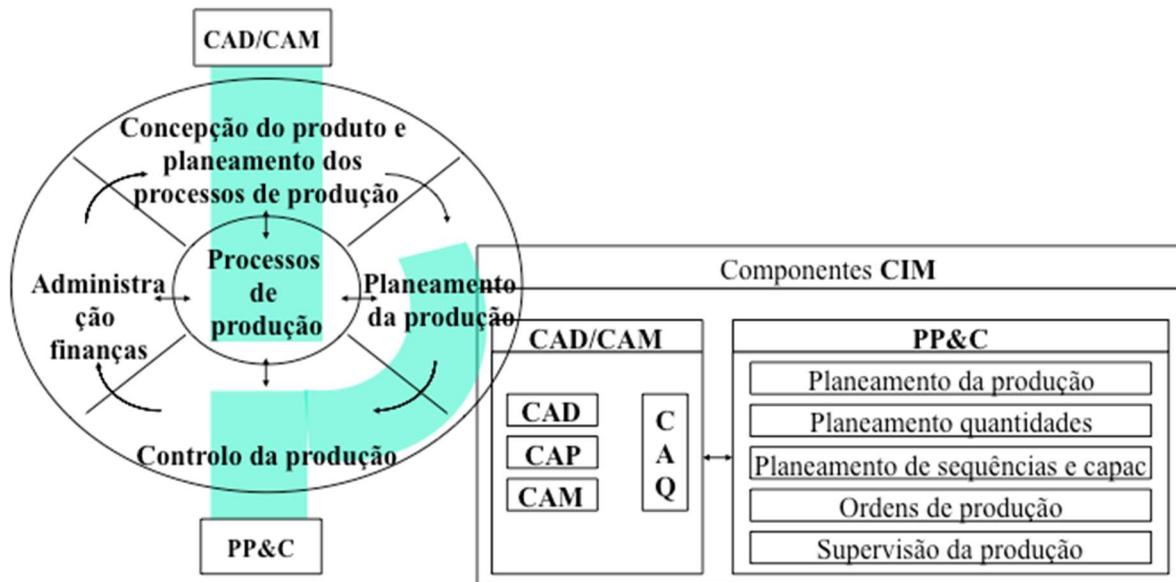


figura 1.7: Informatização das atividades de uma empresa - introdução

Podemos constatar, pelo que foi apresentado até agora, que a produção integrada por computador (**CIM**) comprehende o recurso aos computadores para informatizar as atividades de gestão, concepção e planeamento, bem como o recurso a computadores e automatismos programáveis para controlar os processos de produção (1.2.2). Além destes dois requisitos, um sistema **CIM** necessita ainda de integrar as atividades informatizadas, suportando de forma automática a transferências de informação e material entre componentes (fluxos de informação e de material). Impõe-se por isso e antes de mais, caracterizar esses fluxos, o que será feito na secção (1.2.3).

No capítulo seguinte, “Integração das atividades da empresa”, serão abordadas as arquiteturas de suporte à integração das atividades da empresa.

2.2.1. Informatização do ambiente de engenharia

O modelo em Y de [Scheer' 94] (figura 1.8) permite relacionar os processos e atividades da empresa entre si, mas também relacioná-los de forma mais detalhada com as ferramentas informáticas de apoio i.e. os sistemas de CAX (Computer Aided X). O braço esquerdo do Y representa as funções de organização e planeamento, enquanto que o braço direito apresenta as funções técnicas.

O **PP&C** (Production Planning & Control) abrange um conjunto de atividades tais como o planeamento dos recursos a utilizar durante a produção, a definição das ordens de fabrico e o seu escalonamento temporal, bem como a supervisão e controlo das operações na instalação fabril.

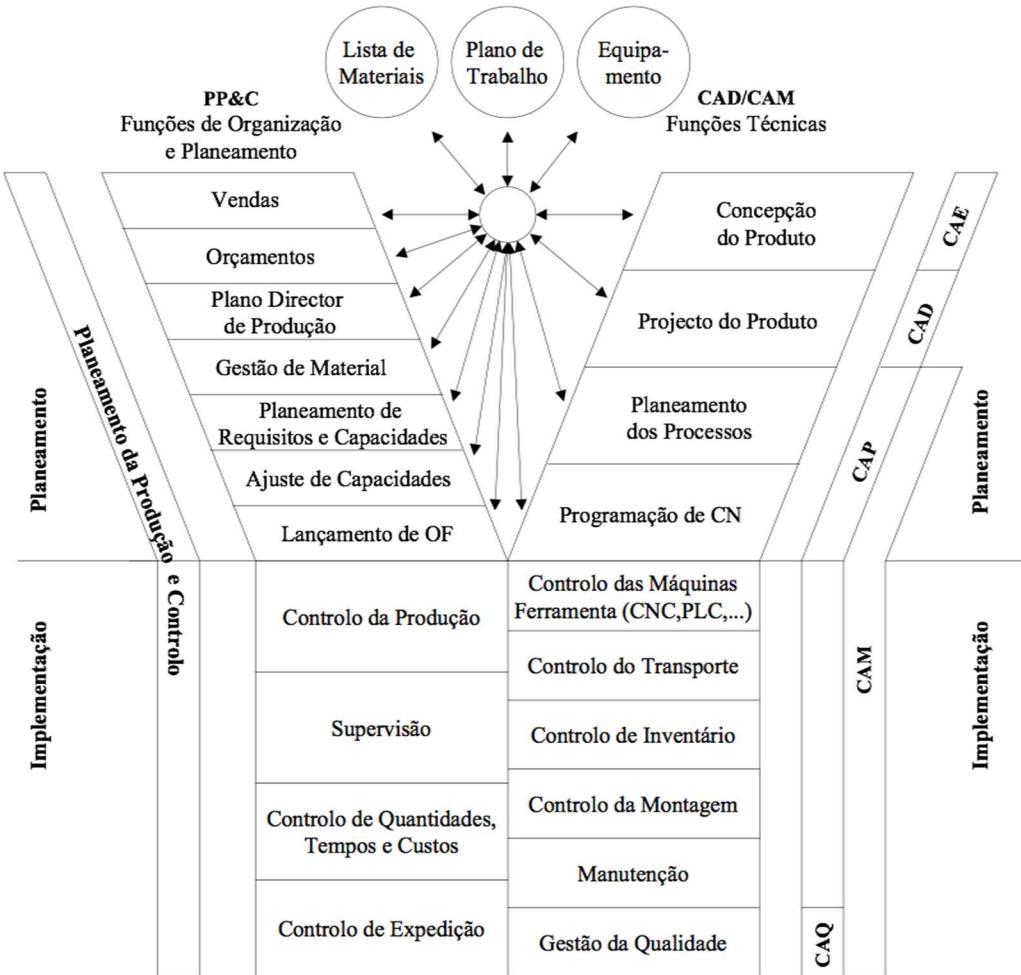


figura 1.8: Informatização das atividades de uma empresa - detalhe

O **CAD** (Computer Aided Design) consiste no projecto dos produtos a fabricar com recurso a aplicações informáticas. As empresas de fabrico com projecto próprio desenvolvem o desenho das peças a fabricar utilizando aplicações como o ProEngineer (módulo de CAD), o Solidworks, ou o AutoCAD, para a criação de ficheiros com o desenho técnico das peças a maquinar. Outra das atividades suportadas pelos sistemas informáticos de CAD consiste na definição da lista de materiais necessários à produção da peça.

Segundo [Scheer' 94] o CAD, além de incluir o desenvolvimento dos programas peça, inclui ainda o desenvolvimento de programas para controlar robôs ou outros automatismos programáveis. Outros autores consideram o desenvolvimento destes programas atividades de planeamento da produção (CAP), e há ainda quem considere como ferramentas de CAM as aplicações informáticas que convertem os ficheiros de CAD em programas peça. Segundo Scheer, as atividades de planeamento dos processos e o desenvolvimento dos programas para os recursos programáveis são atividades de CAP. Mas por sua vez e segundo Scheer as atividades de CAP fazem parte do CAM, reflectindo uma noção abrangente do fabrico assistido por computador.

O **CAP** (Computer Aided Planning) agrupa, como o seu nome indica as atividades de planeamento dos processos, incluindo a definição do Plano de Produção, e o planeamento dos materiais necessários à produção. O Plano de Produção descreve a sequência dos processos tecnológicos de fabrico e montagem que irão decorrer na instalação fabril para produzir um lote de peças. O CAP inclui também o desenvolvimento de programas para os comandos numéricos das máquinas ferramenta, robôs e outros recursos programáveis da instalação fabril.

O **CAQ** (Computer Aided Quality) agrupa, as atividades de: definição dos procedimentos para a avaliação e garantia da qualidade, criação de relatórios de qualidade, definição dos parâmetros de qualidade do sistema de fabrico, incluindo por exemplo os sistemas de inspecção, medida e testes.

O termo **CAM** (Computer Aided Manufacturing) tem significados ligeiramente diferentes de autor para autor, “*algumas vezes refere-se simplesmente ao controlo computorizado de linhas de transferência, armazéns, e outros recursos produtivos; outras vezes é bastante abrangente incluindo as atividades de planeamento e controlo da produção* [Scheer’ 94]. Segundo outra definição, CAM “*compreende o controlo por computador das atividades que decorrem na instalação fabril, incluindo o controlo directo dos recursos de produção e a gestão de materiais, ferramentas de corte, e atividades de manutenção* [Rembold’ 93]. Esta última definição inclui as atividades de controlo e supervisão da instalação fabril com recurso aos computadores, nomeadamente o controlo e supervisão em “tempo real” dos recursos de produção: máquinas ferramenta com comando numérico, robôs, linhas de transferência, etc., sincronizando e coordenando as suas operações.

Por vezes, são também consideradas como atividades de CAM o desenvolvimentos de programas para os comandos numéricos e outros recursos da instalação fabril (centros de maquinagem, robôs, ...) [Scheer’ 94].

O termo **CAD/CAM** é também utilizado frequentemente, designando as atividades de CAD, CAP, CAM e CAQ.

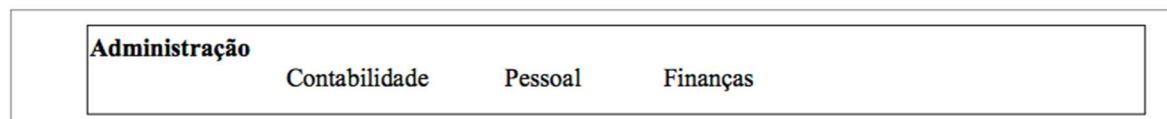
O termo **CIM** (Computer Integrated Manufacturing) comprehende a completa informatização e integração das atividades da empresa. Para isso um sistema CIM necessita por um lado de informatizar e automatizar as suas atividades (CAX), utilizando ferramentas informáticas e recursos de produção programáveis, e por outro lado de permitir que a informação gerada por essas aplicações e recursos, possam fluir rapidamente pela empresa, recorrendo para isso a redes de comunicação, bases de dados partilhadas, plataformas distribuídas ou outras soluções tecnológicas que permitam agilizar os fluxos de material e informação.

Exemplo I

A figura 1.9 [Rembold’ 93] apresenta um exemplo de como a Siemens interpreta os conceitos CIM e os complementa com o apoio computacional à organização da empresa nas restantes áreas do negócio.

Desta forma pretende-se dar um exemplo prático das definições de cada um dos componentes CIM. Para a Siemens, o CIM é encarado como uma estratégia e um conceito para alcançar os objectivos de mercado e da empresa. Os componentes CIM encontram-se interligados por fluxos de informação intensos, dando origem a requisitos de transferência e processamento de informação em “tempo real”, necessitando por isso a empresa de possuir infra estruturas de integração, baseadas nas TI, como por exemplo redes de comunicação, plataformas distribuídas e bases de dados normalizadas e partilhadas.

CAO - Computer aided organisation



Computer Integrated Manufacturing (CIM)

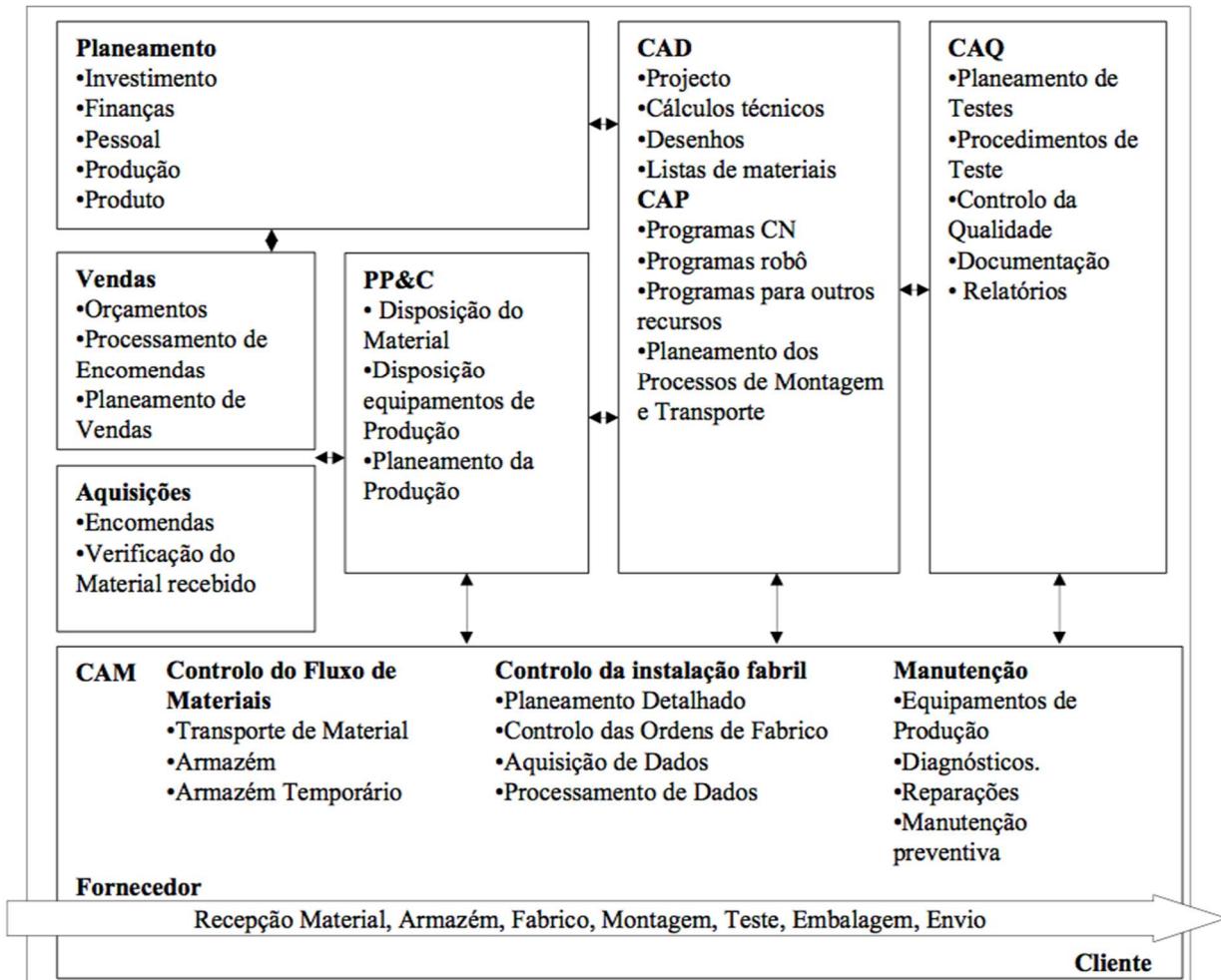


figura 1.9: SIEMENS - Componentes CIM

Exemplo II

A Digital Equipment Corporation (DEC) também procurou melhorar os processos de produção recorrendo aos computadores e à integração dos fluxos de informação. A figura 1.10 [Rembold' 93] apresenta uma solução de integração funcional através da qual dois recursos da empresa, independentes e fisicamente afastados, cada um executando uma atividade podem trocar informação entre si. A solução implementada pela DEC procura dar resposta a um conjunto específico de requisitos de um sistema CIM no que respeita aos sistemas de informação.

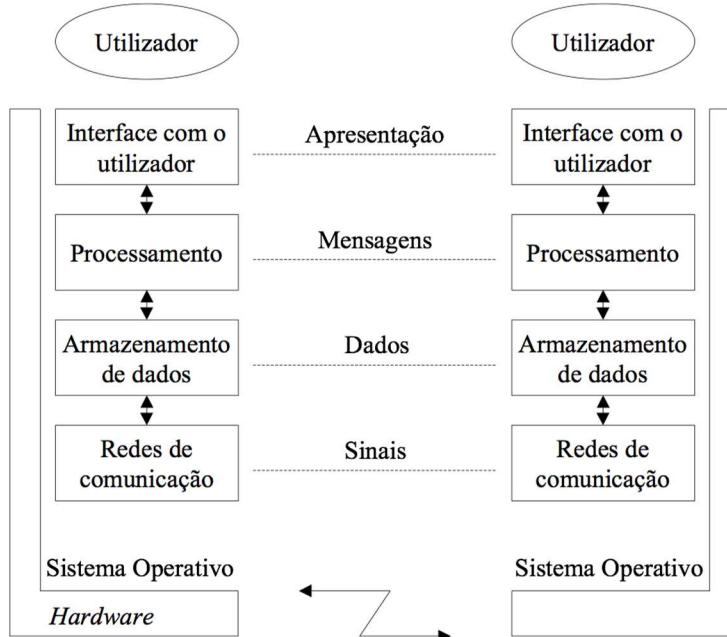


figura 1.10: DEC – Uma solução de integração

Segundo este modelo dois utilizadores, entendam-se recursos da empresa, podem comunicar entre si segundo uma arquitetura de quatro camadas suportada pelos sistemas operativos e respectivo hardware.

Interface com o utilizador

Esta camada permite que o seu utilizador (uma aplicação informática) possa transferir informação sem ter de se preocupar com a conversão prévia dessa informação em estruturas de dados normalizadas que sejam reconhecidas pelo seu interlocutor. Com esta arquitectura de comunicação, as normas de representação de dados são implementadas pelas camadas de “Interface com o utilizador”, tanto na camada local como na camada remota.

Processamento de dados

Esta camada permite que várias aplicações, ativas em simultâneo num só recurso, possam partilhar a mesma infra-estrutura de comunicações transferindo e processando mensagens em paralelo.

Armazenamento de dados

A camada de armazenamento de dados fornece todos os serviços necessários à gestão dos dados e ficheiros existentes na empresa, ainda que estes possam estar fisicamente distribuídos por diversos recursos de origens e fabricantes diferentes. Esta camada é também responsável por implementar mecanismos de controlo que garantem os acessos, a partilha, a fiabilidade e a redundância dos dados.

Redes de comunicação

Para que as camadas anteriores possam fornecer os serviços descritos, necessitam de comunicar com as suas camadas homólogas, existentes noutros recursos da empresa, recursos que se encontram fisicamente distribuídos. As redes de dados são responsáveis por assegurar essa comunicação, codificando os dados em sinais eléctricos, ópticos, rádio-eléctricos, ou outros, adequados ao meio físico de transmissão utilizado para interligar fisicamente os recursos da empresa. Além disso, esta camada é responsável por controlar o acesso dos recursos ao meio de transmissão, implementando protocolos de partilha do meio, protocolos de recuperação de erros de transmissão, protocolos de endereçamento e outros.

2.2.2. Informatização e automatização dos processos de produção

Durante as últimas décadas as empresas têm recorrido maciçamente às TI para ajudar a dar resposta às crescentes solicitações do mercado em termos de custos, qualidade, rapidez de resposta, etc.

Esta secção procura descrever as várias etapas da informatização e da automatização dos processos industriais, contínuos ou discretos, com especial ênfase nos recursos de produção e em particular nos seus sistemas de controlo.

Numa primeira fase, os recursos de produção eram controlados manualmente, o que limitava a cadência da produção, a qualidade e a fiabilidade dos processos. Procurando obviar a estes problemas os recursos de produção passaram gradualmente a utilizar sistemas de controlo automáticos (automatismos). Estes sistemas começaram por ser mecânicos e pneumáticos, posteriormente surgiram sistemas eléctricos (circuitos de relés, ...), e mais tarde ainda sistemas de controlo electrónicos (analógicos e digitais), bastante mais rápidos que os anteriores permitindo automatizar equipamentos muito complexos e capazes de assegurar elevadas cadências de produção.

Contudo, este tipo de sistemas de controlo, mesmo os digitais, não eram no inicio programáveis, pelo que era necessário projectar e construir novos recursos, e respectivos automatismos de controlo, sempre que era necessário alterar a produção (ex: um novo produto implicando operações ou sequências distintas). Numa segunda fase, esses automatismos passaram a ser programáveis, o que aumentou a sua flexibilidade, “bastando” alterar o programa em computador para, por exemplo, ser possível maquinar uma peça diferente. Numa terceira fase, passou a ser possível não só programar cada recurso isoladamente, mas também comunicar com o seu controlador local a partir de um computador, utilizando para o efeito redes de comunicação de dados. Dessa forma passou a ser possível coordenar e sincronizar vários recursos produtivos, utilizando para isso redes de comunicação de dados e plataformas informáticas, suportando o controlo e supervisão integrado da produção[\[JPS1\]](#).

No início deste século (1900-1920) o controlo dos processos, nomeadamente dos processos contínuos, era analógico recorrendo à tecnologia mecânica disponível na época. Os parâmetros dos processos eram visualizados através de indicadores (de pressão, caudal, temperatura, ...) dispostos na instalação fabril. O papel do operador humano consistia na visualização destes parâmetros e no ajuste manual das válvulas para controlo das variáveis dos processos.

Durante duas décadas, 1920 a 1940, surgiram sistemas de controlo realimentado de processos, baseados na tecnologia pneumática. O controlo analógico realimentado era mais fiável, pois permitia uma regulação “automática” de, por exemplo válvulas, em função de parâmetros do processo, como o caudal de um fluído. Neste tipo de controlo realimentado competia ao operador humano a definição e supervisão dos valores óptimos para os quais o sistema realimentado devia convergir. Mais tarde, surgiu o controlo integral, o qual reduziu a necessidade de intervenção do operador humano. Também o controlo derivativo foi desenvolvido durante este período, aparecendo então os primeiros sistemas de controlo PID (proporcional, integral e derivativo). Inicialmente os visualizadores e actuadores das variáveis dos processos encontravam-se junto aos recursos fabris, estando por isso dispersos pela instalação fabril. Em instalações fabris de grandes dimensões esta situação obrigava à contratação de um grande número de empregados para a supervisão e controlo dos processos, pelo que se resolveu centralizar grande parte desses visualizadores e actuadores num só local da instalação fabril. Para isso foram desenvolvidos “transmissores de informação”, pneumáticos, permitindo centralizar todo o controlo dos processos numa única sala (sala de comando ou sala de controlo).

Durante as décadas de 50 e 60 o controlo dos processos baseou-se fundamentalmente na tecnologia pneumática. O papel dos operadores consistia na definição dos valores óptimos das variáveis dos processos, supervisionando e actuando em caso de necessidade. Apesar da vantagem do controlo centralizado, a partir de grandes salas de controlo, continuaram a ser necessários operadores de campo,

junto aos recursos, podendo intervir rapidamente em caso de necessidade, comunicando entre si através de telefones ou rádios.

A concentração das funções de controlo em grandes “salas de controlo” aumentou com o desenvolvimento dos controladores electrónicos na década de 1950. Este controladores permitiram que a informação dos processos pudesse ser transmitida e processada mais rapidamente, além de permitirem distâncias maiores entre os recursos a controlar e a sala de controlo. Muitos transmissores de informação electrónicos são desenvolvidos nesta altura, bem como conversores pneumático-electrónico e electrónico-pneumático para controlo de válvulas, por exemplo.

Durante as décadas de 60, 70 e 80, para além da utilização da electrónica e da pneumática no controlo analógico de processos, começaram também a ser utilizados computadores. Inicialmente, os computadores foram utilizados apenas em instalações de média e grande dimensão na supervisão dos processos industriais, devido à sua flexibilidade, rapidez de aquisição de dados e cálculo. Os computadores digitais começaram então também a ser usados no controlo directo de alguns processos, implementando algoritmos PID que até então eram implementados por controladores analógicos.

Desde as décadas de 70 e 80 a utilização de computadores digitais, nomeadamente de microprocessadores, vulgarizou-se na indústria devido à sua fiabilidade e flexibilidade assim como baixo custo. Continuaram no entanto a ser utilizados alguns sistemas de controlo analógico, na aquisição de sinais e controlo de recursos analógicos, assim como alguns sistemas de controlo pneumático de válvulas.

Devido à elevada capacidade de processamento dos microprocessadores tornou-se possível adquirir e processar grandes quantidades de informação em “tempo real”, pelo que passaram a ser utilizadas redes de comunicação de dados para a transferência de informação entre sistemas de controlo de processos. As redes de comunicação permitiram o desenvolvimento de arquitecturas distribuídas de controlo e, juntamente com os computadores e restantes sistemas de controlo automatizado, permitiram uma maior integração das atividades do negócio e de fabrico.

No início do século a principal força motriz na indústria era obtida a partir de máquinas a vapor, turbinas hidráulicas, e animais. Esta energia era transmitida às máquinas através de veios que atravessavam horizontalmente a instalação fabril e dos quais, desciam correias até às máquinas respectivas.

A primeira **máquina ferramenta** accionada por um motor eléctrico individual foi construída em 1903 e gradualmente a principal força motriz passou a ser a energia eléctrica, dado que permitia melhorar substancialmente a qualidade da produção.

Mais tarde, durante a década de 40, surgiram as *linhas de transferência* as quais permitem o transporte automatizado (fluxo de materiais) num percurso pré-estabelecido e fixo, levando o material a ser processado até às máquinas respectivas.

Estes sistemas de transporte de material eram inicialmente controlados por circuitos de relés e sensores de posição. Actualmente, as linhas de transferência encontram-se largamente divulgadas na indústria, sendo controladas por recursos digitais programáveis (**autómatos programáveis ou Programmable Logic Controllers - PLCs**). Os primeiros PLCs foram desenvolvidos no final da década de 60 início da década de 70, dando resposta às crescentes solicitações da indústria automóvel. Em meados da década de 70 os PLC passaram a incorporar microprocessadores permitindo o cálculo aritmético e a manipulação de dados. Actualmente podem dispor de capacidade para comunicar em rede, podendo comunicar com computadores de mais alto nível e fazer o interface entre estes e os recursos de produção.

Na segunda metade deste século proliferou a utilização da electrónica analógica e sobretudo digital no controlo dos processos de produção. Grande parte da tecnologia desenvolvida durante a segunda guerra mundial para a indústria militar, foi posteriormente aplicada à indústria civil, nomeadamente no controlo das máquinas ferramenta e de outros recursos de produção. A partir da década de 50 o controlo

digital dos recursos de produção, seja o controlo local seja o controlo integrado de vários recursos de produção, expandiu-se fortemente. A utilização na instalação fabril de ferramentas programáveis (**Automatic Programmable Tools – APT**) controladas por dispositivos programáveis, permitindo melhorar a qualidade e flexibilidade da produção das empresas.

Na década de 60 surgiram as primeiras máquinas ferramenta controladas numericamente. Segundo a norma [ISO/CEI 2382-24:1995] e [ISO 2806:1994] o **Comando Numérico (CN)** é “*um dispositivo que faz uso de comandos em código numérico introduzidos em “tempo real” permitindo o controlo automático de máquinas ferramenta ou processos industriais*”. O controlo numérico (CN) das máquinas ferramenta baseou-se inicialmente em recursos electrónicos e mais tarde em computadores. Inicialmente as máquina ferramenta eram controladas numericamente, por sistemas electrónicos de controlo, analógico ou digital, que apenas estavam preparados para ler fitas perfuradas ou fitas magnéticas, previamente perfuradas ou gravadas noutro recurso, contendo o programa de maquinagem (programa peça) com as trajectórias das ferramentas, velocidades de avanço, de corte, etc.

A utilização do computador para o controlo numérico (**Computer Numerical Control - CNC**) de máquinas ferramenta foi um dos avanços mais significativos ao nível dos recursos de produção. O primeiro Comando Numérico baseado em computador desenvolvido para controlar uma máquina ferramenta surgiu em 1958, como resultado de um contracto estabelecido entre a força aérea e o MIT (Massachusetts Institute of Technology). O computador usado era o Whirlwind, baseava-se em componentes electrónicos de estado sólido (válvulas, diodos, tríodos, ...) e ocupava grande parte do laboratório. As máquinas ferramenta com o comando numérico por computador instalado fisicamente na máquina permitiram a escrita do programa peça directamente na consola da máquina, indicando ao recurso quais as trajectórias da ferramenta de corte, a sua rotação, bem como as estratégias de desbaste e acabamento. Contudo, para maquinar peças de geometria complexa era difícil definir todas as trajectórias directamente na consola, pelo que se começou, mais tarde, a recorrer a aplicações informáticas de CAD/CAM.

Paralelamente, em 1966, o sistema digital de CN das máquinas ferramenta passou a poder receber directamente os programa peça (**Direct Numeric Control - DNC**), desenvolvidos num computador externo, via porta série (RS232) sem necessitar de usar fitas perfuradas ou magnéticas.

No início da década de 60, as máquinas ferramenta com comando numérico foram equipadas com sistemas de troca automática de ferramenta sendo-lhes dado o nome de **centros de maquinagem**. Um centro de maquinagem pode assim executar um número muito mais elevado de operações sem intervenção humana, bastando para tal que o programa de CN descreva as operações a efectuar e as ferramentas adequadas a cada operação.

A utilização de **robôs manipuladores** na produção permitiu automatizar outros processos de produção, nomeadamente os processos de montagem, soldadura e pintura de componentes. Por definição, um robô manipulador é “um manipulador programável, projectado para mover peças ou dispositivos específicos com movimentos variáveis e programáveis, executando um conjunto variado de tarefas [Schlussel, 1985].

Os robôs começaram a ser desenvolvidos durante os anos 50 mas só começaram a ser usados na indústria nas décadas de 60, tendo passado a ser controlados por computador durante os anos 70. A utilização de computadores para controlar os movimentos dos robôs manipuladores permitiu dotá-los de uma enorme capacidade e flexibilidade. Na instalação fabril os robôs manipuladores podem ser dispostos de forma a alimentar centros de torneamento, sistemas de armazenamento e sistemas de transporte de material. Permitem uma maior flexibilidade de movimentos, num espaço mais restrito que os AGV ou as linhas de transferência.

A movimentação flexível de matérias primas, e componentes beneficiou fortemente com a introdução de **AGV's** (*Automatic Guided Vehicle- AGV*), veículos que permitem o transporte automatizado de materiais, componentes ou ferramentas na instalação fabril. Um AGV permite com alguma flexibilidade a alteração dos fluxos dos materiais, por exemplo alterando o trajecto de um fio metálico

ou de uma linha pintada no pavimento da instalação fabril, a linha traçada no pavimento que o AGV usa para se orientar (razão pela qual se denominam também veículos filoguiados).

Procurando aumentar a cadência de produção dos centros de maquinagem, polimento, etc., automatizou-se a colocação dos materiais nas suas mesas de trabalho dando origem às **células de fabrico flexível**. Segundo F.J.Martin, uma célula de fabrico flexível é um “*sistema formado por duas ou mais máquinas ferramenta com controlo numérico, integradas por um sistema automatizado de carga e descarga de materiais e ferramentas*”. Uma célula de fabrico pode ser dotada de armazéns para guardar temporariamente a matéria prima e as peças produzidas, podendo assim fabricar peças semelhantes com uma elevada cadência e durante períodos de tempo relativamente longos.

A área da armazenagem foi das últimas a beneficiar da introdução das tecnologias de automação e informatização. Os **Sistemas de armazenamento automatizados** permitem guardar ferramentas, peças ou matérias primas, sendo controlados por automatismos que lhes permitem gerir o seu espaço de armazenamento e alguns casos comunicar com outros recursos da empresa funcionando de forma integrada com todos os outros recursos produtivos.

Finalmente, a integração hierarquizada de recursos automatizados e informatizados permite conciliar dois conceitos inherentemente contraditórios: automação e flexibilidade.

O resultado são os **Sistemas Flexíveis de Produção (SFP)**, segundo a norma [ISO/CEI 2382-24:1995], “*são compostos por unidades de fabrico, controladas por comando numérico, e sistemas de transporte de material automatizado, ambos controlados por computador, permitindo alterar facilmente as tarefas de fabrico*”. Os SFP caracterizam-se por serem sistemas de produção controlados por computador em que apenas é necessário alterar a programação para que seja possível produzir um novo lote de produtos. O início de uma nova produção é também mais fácil e rápido, pois “apenas” é necessário alterar os programas de controlo e supervisão, bem como o(s) programas peça. Um SFP pode agrupar várias células flexíveis de produção, sendo de facto sistemas de processamento de fluxos de informação e de materiais: os sistemas físicos asseguram os fluxos de material e os sistemas de informação processam e asseguram os fluxos de informação. O controlo de todos os recursos é assegurado por computador sendo as trajectórias de material e o comportamento dos recursos definidas por ele.

No final dos anos 70 foi desenvolvido pela Cincinnati Milacron Company o primeiro sistema de flexível de produção, em inglês (**FMS-Flexible Manufacturing System**), tendo sido apelidado na altura de “Variable Mission”. Na altura esta solução era todavia demasiado avançada e cara para a época.

2.3. Caracterização dos fluxos de informação

Os processos de negócio e de fabrico de uma empresa são constituídos por redes de atividades, interligadas entre si, sendo a informação gerada por uma determinada atividade, numa determinada fase desse processo, frequentemente necessária às atividades seguintes.

Uma das formas de aumentar a eficiência da empresa consiste no recurso às tecnologias de informação para suportar e agilizar as transferências de informação entre atividades recorrendo à transferência de informação entre as aplicações informáticas que suportam cada uma dessas atividades. Dado que este trabalho se debruça sobre o suporte e a integração de parte destes fluxos de informação necessários ao controlo dos fluxos de materiais na instalação fabril, importa caracterizá-los com maior detalhe.

Nesta secção procurar-se-á apenas caracterizar os fluxos de informação gerados pelas atividades de CAD/CAM, e os fluxos de informação trocados entre estas e as atividades de PP&C.

2.3.1. Fluxos de informação CAD – CAM

Utilizando ferramentas informáticas é possível suportar e integrar as atividades de projecto e produção CAD/CAM (figura 1.11), gerando com maior rapidez ficheiros de controlo numérico a partir das peças desenhadas em computador. Contudo, os ficheiros de CAD não podem ser enviados directamente para os dispositivos de controlo, comandos numéricos ou outros recursos programáveis, pois estes não seriam capazes de os interpretar correctamente. Os comandos numéricos das máquinas ferramenta, por exemplo, necessitam de receber ficheiros com a informação das trajectórias da ferramenta de corte (interpolações lineares, circulares, ciclos de furação, roscação, etc.), das velocidades de avanço, e das velocidades corte, sendo essas trajectórias função das estratégias de desbaste, e acabamento, bem como das ferramentas escolhidas.

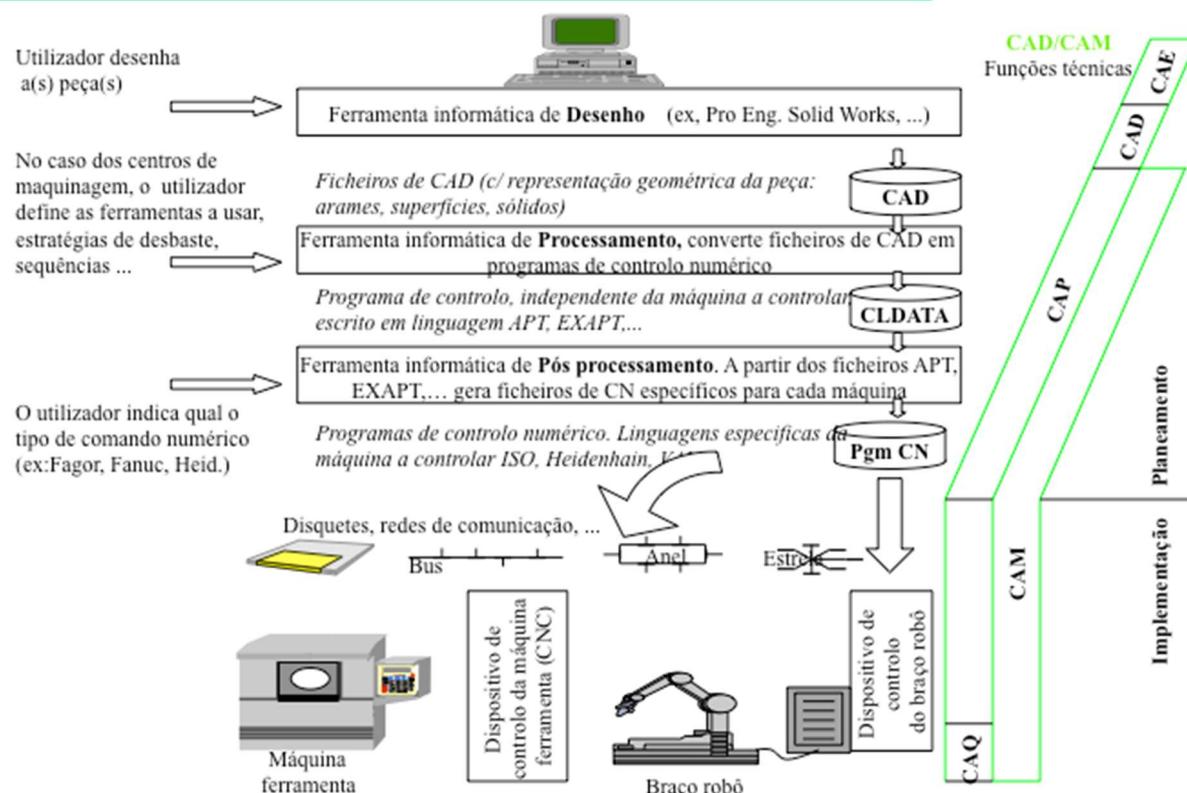


figura 1.11: CAD/CAM: Atividades e fluxos de informação

Assim sendo, os ficheiros de CAD necessitam de ser previamente convertidos em programas peça (ex: em linguagem ISO ou Heidenhain). A conversão dos ficheiros de CAD é feita em dois passos intermédios: processamento e pós-processamento. A partir dos desenhos criados pelo utilizador,

internamente, as ferramentas de CAD geram ficheiros com a representação em computador da peça desenhada (ficheiros de CAD contendo o modelo geométrico da peça).

Algumas ferramentas de CAD geram modelos da peça baseando-se nas suas arestas, outras nas suas superfícies e outras ainda à custa de sólidos elementares que conjugados permitem obter a forma da peça desenhada.

Uma vez que a representação interna da geometria da peça pode variar consoante a ferramenta de CAD usada (mesmo quando usam o mesmo tipo de modelo, por exemplo baseado em superfícies), a ANSI (“American National Standards Institute”) propôs a norma IGES (Initial Graphics Exchange Standards) para permitir que ferramentas informáticas de fabricantes diferentes possam trocar ficheiros de CAD. Os ficheiros IGES são compostos por conjuntos de caracteres ASCII (cada conjunto tem 80 caracteres). Esta norma foi inicialmente pensada para representar peças em duas dimensões, tendo sido posteriormente alterada para permitir representar peças em três dimensões.

As atividades de CAP permitem gerar os ficheiros de controlo numérico necessários às atividades de CAM, a partir dos ficheiros de CAD.

Num primeiro passo, além das dimensões geométricas da peça a maquinar (ficheiro de CAD), o utilizador necessita ainda de definir: as estratégias de desbaste e acabamento, as ferramentas a utilizar, as velocidades de corte e avanço e alguns parâmetros mais, para que seja gerado um ficheiro (*Cutter Location Data - CLDATA*) com a informação das trajectórias de corte. Os ficheiros CLDATA são ficheiros ASCII, escritos por exemplo em linguagem APT (*Automatic Programmable Tool - APT*).

Num segundo passo, o ficheiro CLDATA necessita ainda de ser pós-processado de forma a ser gerado um ficheiro (programa peça em ISO ou Heidenhain) na linguagem específica do CN da máquina ferramenta.

Uma vez criados, os programas de controlo numérico serão então utilizados pelas atividades de PP&C que em última análise os enviarão, via rede de comunicações, para os recursos fabris, de acordo com o escalonamento de tarefas previamente definido em função das suas taxas de ocupação, tempos e custos de produção, etc.

2.3.2. Fluxos de informação CAD/CAM -> PP&C

A figura 1.12 [Scheer' 94] apresenta de forma integrada os principais fluxos de informação trocados entre as atividades de concepção projecto e fabrício (CAD/CAM) e as atividades de planeamento e controlo da produção (PP&C).

Durante as fases de concepção e projeto de um novo produto são também elaboradas as lista de materiais necessários à sua produção. Com base nas listas de materiais e ou componentes utilizados na produção de outros produtos, pode-se estudar a hipótese de reutilizar desenhos ou componentes existentes.

Salienta-se também a transferência de programas de CN, entre das atividades de CAD/CAM para as atividades de PP&C, mais precisamente as atividades de lançamento de Ordens de Fabrico e de Controlo da Produção, que os irão enviar para os recursos fabris mais adequados, no momento adequado de acordo com o escalonamento elaborado.

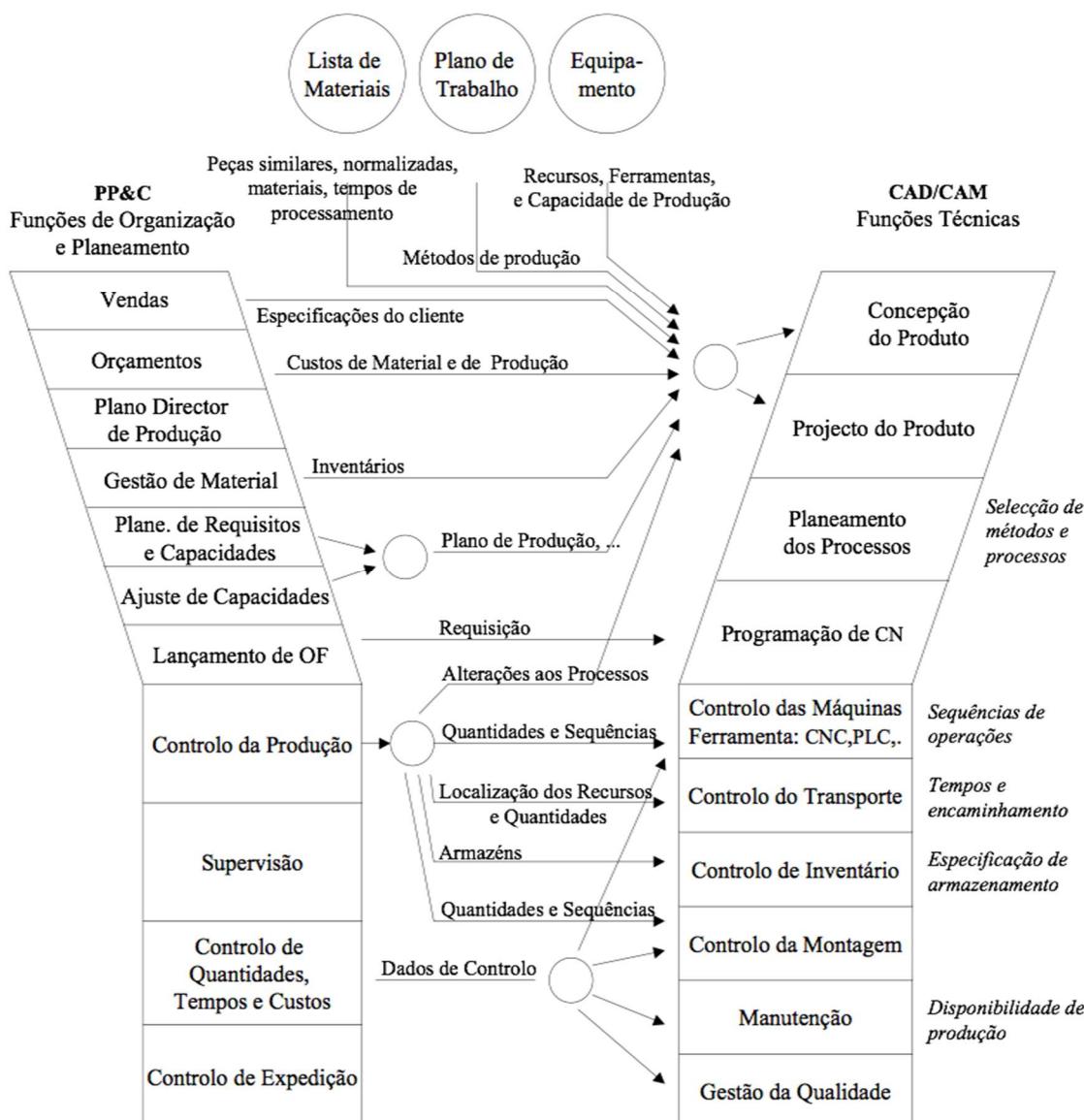


figura 1.12: Fluxos de dados do CAD/CAM para o PP&C [Scheer' 94]

2.3.3. Fluxos de informação PP&C → CAD/CAM

A figura 1.13 [Scheer' 94] apresenta os principais fluxos de informação trocados entre as atividades de PP&C e as atividades de CAD/CAM.

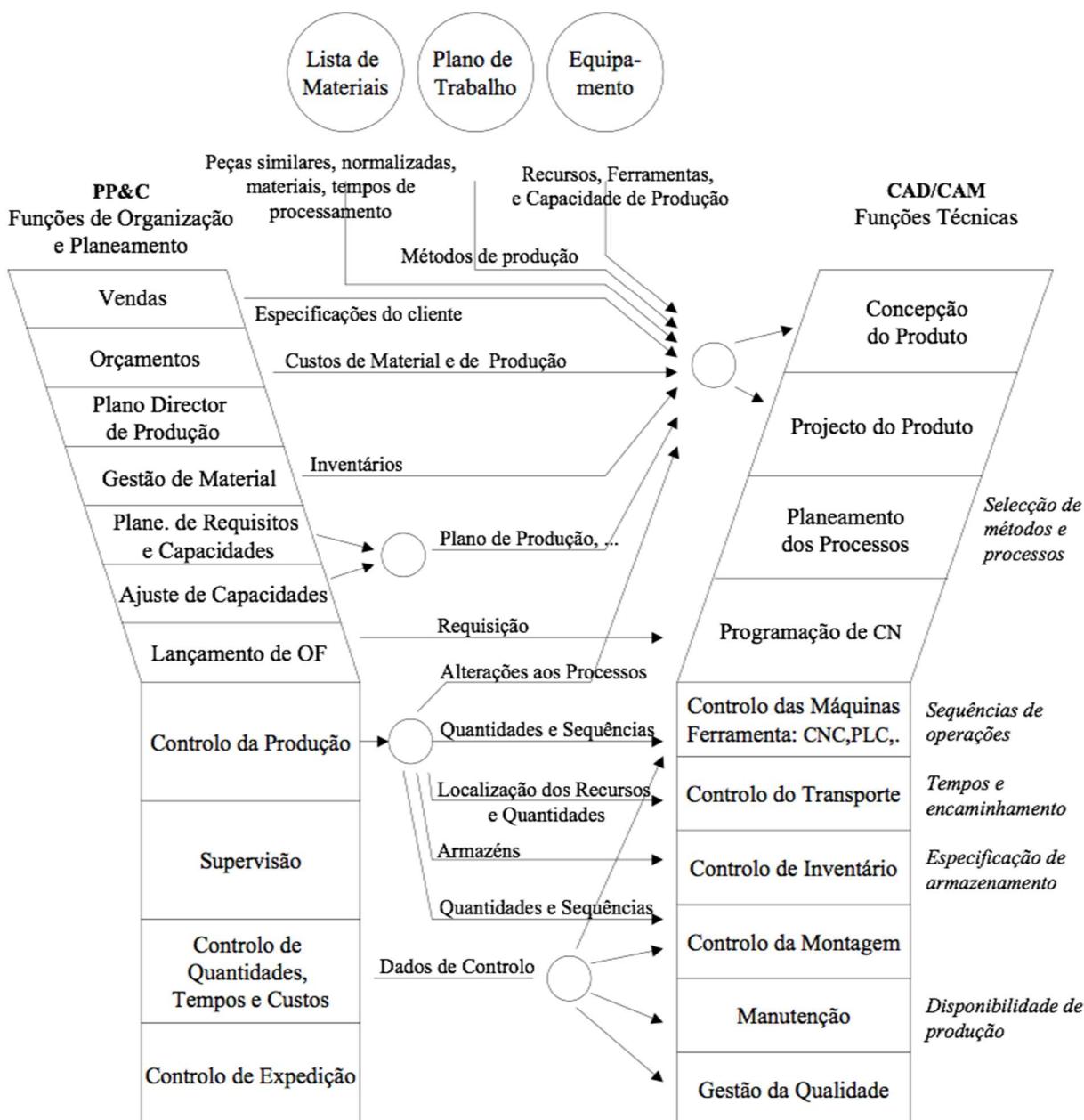


figura 1.13: Fluxos de dados do PP&C para o CAD/CAM

Para que seja possível prever os prazos de entrega e os custos de produção de um novo produto é necessário aceder a um conjunto variado de informações. Para além das especificações definidas pelo cliente, as atividades de concepção e projecto necessitam ainda de conhecer a taxa de ocupação dos recursos da empresa, bem como o plano director de produção, para poder com maior exactidão e fiabilidade estimar os tempos e custos de produção de um novo produto. Além disso, as atividades de CAE e CAD necessitam ainda de saber que materiais e componentes estão disponíveis em armazém e quais os que necessitam de adquirir, para ser possível estimar tempos de entrega da encomenda ao cliente, bem como os custos de produção. Por sua vez, o orçamento estimado não depende exclusivamente dos custos de produção, dependendo também de muitos outros factores, nomeadamente da procura do mercado e da concorrência.

Compete ao controlo da produção gerar e enviar informação de controlo para a instalação fabril: alterando, se necessário for, a sequência de ordens de fabrico previamente planeada; coordenando e sincronizando os recursos de produção; enviando informação sobre a localização de materiais e componentes, bem como dos trajectos a seguir pelos materiais ao longo da instalação fabril, e ainda sobre as quantidades a produzir por cada um dos recursos de produção.

2.4.Considerações

Neste capítulo procurou apresentar-se o contexto em que esta disciplina se situa, introduzindo um conjunto de conceitos necessários à sua compreensão.

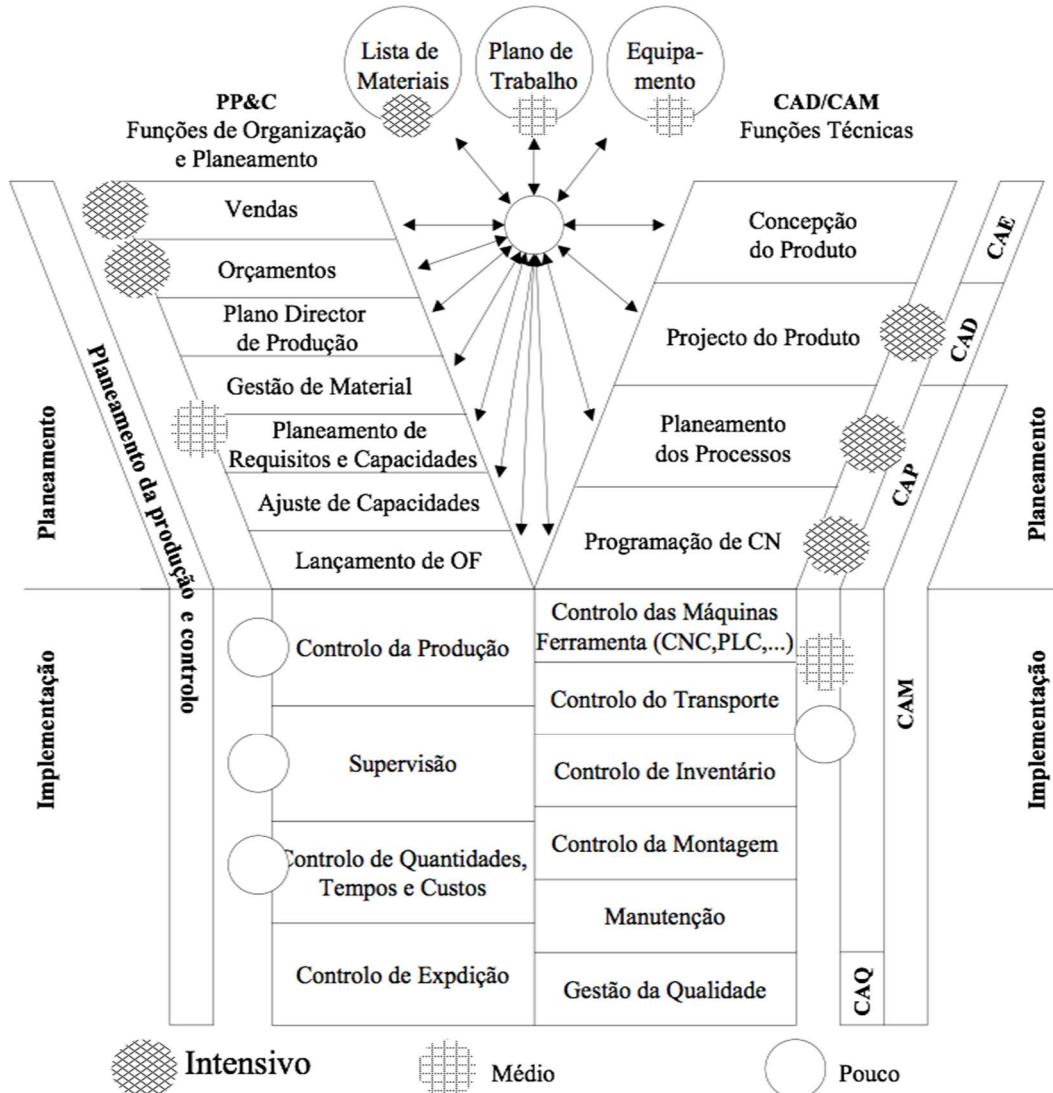


figura 1.14: Níveis de informatização e integração das empresas [Scheer]

Nomeadamente, procurou-se caracterizar: os processos de negócio e de fabrico de um empresa (atividades, operações, ambientes de engenharia e produção), o suporte tecnológico proporcionado pelas TI a essas atividades (CAX, CNC, PLC, robôs, células de fabrico, FMS e outros) e ainda alguns dos problemas que atualmente se colocam à integração de atividades da empresa.

Pode concluir-se que as atividades relacionadas com a gestão financeira, de materiais e recursos, bem como as atividades de CAD/CAM são habitualmente as mais informatizadas e integradas. As atividades de controlo e supervisão da produção, bem como o transporte e armazenamento de material na instalação fabril são, na maioria das empresas, executadas manualmente ou com intervenção humana direta. A figura 1.14 apresenta os diferentes níveis de informatização e integração implementados na maioria das empresas.

2.5. Sistema Flexível de Produção

Tal como já vimos, um sistema flexível de produção é segundo a norma [ISO/CEI 2382-24:1995], “é composto por unidades de fabrico, controladas por comando numérico, e sistemas de transporte de material automatizado, ambos controlados por computador, permitindo alterar facilmente as tarefas de fabrico”.

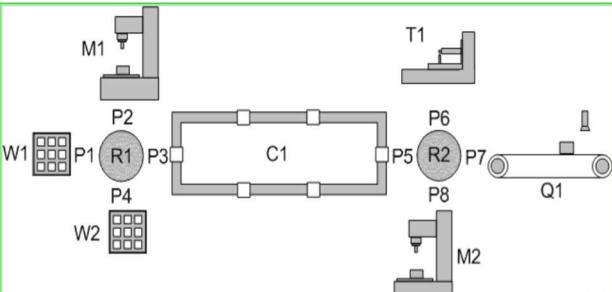
Um exemplo de um sistema flexível de produção, baseado no artigo [Santos et. Al , 2000]

JOSÉ P. O. SANTOS¹, J. J. PINTO FERREIRA², JOSÉ M. MENDONÇA, A modelling language for the design & execution of enterprise models, in manufacturing, International Journal of CIM, 2000.

No ano lectivo de 2004/2005, foi proposto aos alunos de Informática Industrial que desenvolvessem um sistema flexível de produção, nas últimas 4 semanas do semestre. Cada aluno ficou responsável por um recurso fabril. No seu conjunto foi possível monitorizar e controlar a partir de um browser WEB todos os recursos e controlar o seu funcionamento de forma integrada e coordenada.

Interface

A figura seguinte apresenta a página WEB disponível num computador remoto. Nessa página era possível visualizar o layout da instalação fabril/laboratorial, os dados de um cliente, da sua encomenda, e a lista das operações necessária para produzir a sua encomenda. No lado direito da imagem, era possível ver a lista de Clientes e respectivas encomendas. Era também possível fazer a supervisão em tempo real da execução das ordens de fabrico.



Controlo

Cliente nº 0001
nome _____ morada _____

Encomenda nº 0001
nº peças _____ Data Entrega _____

Lista de operações fabris

Nº	Recurso	Op	Param	Estado

Supervisão

Lista de clientes

#Clt	Nome	Morada

Clientes & Encomendas

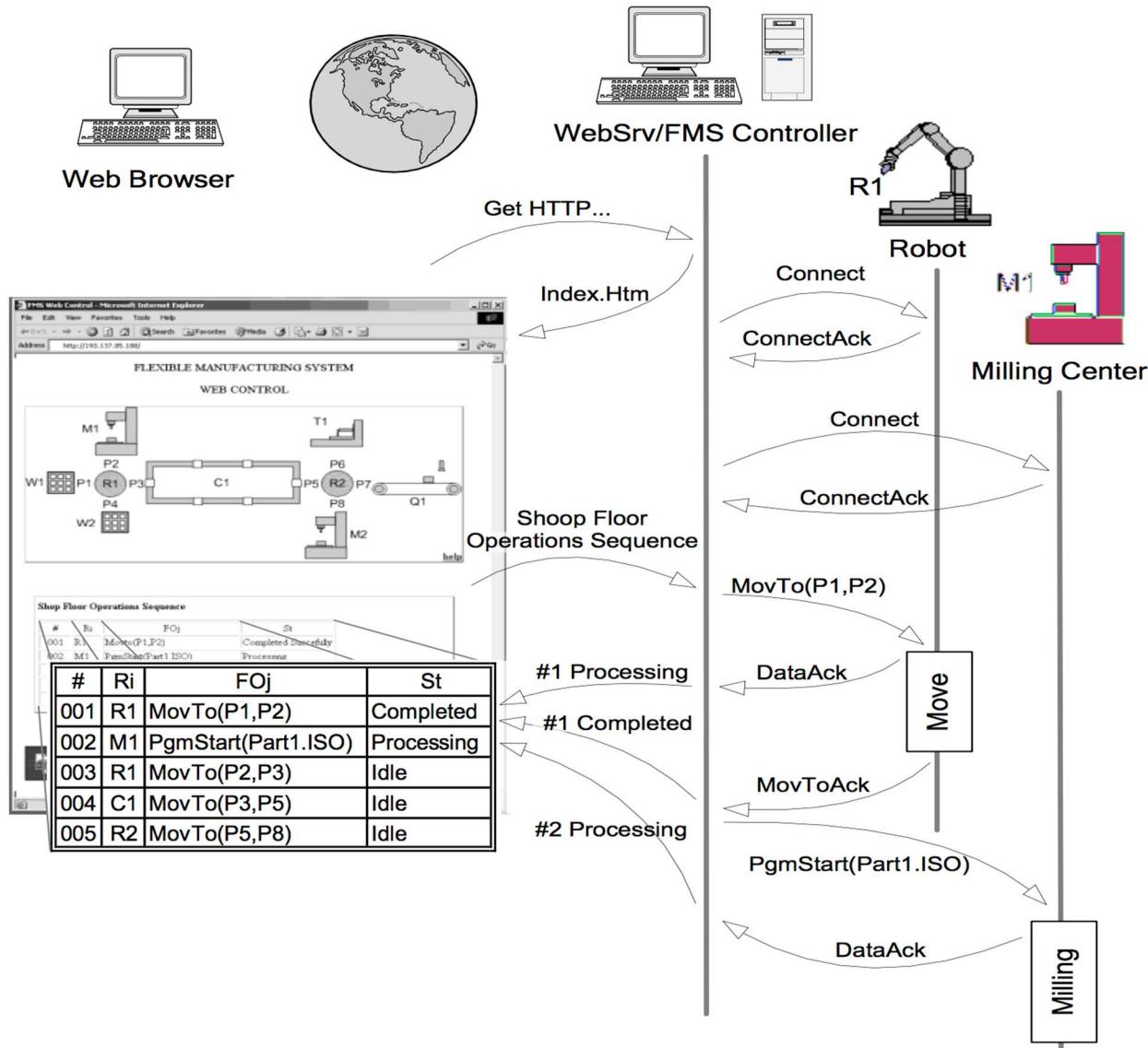
#Clt	#Enc	Nº peças	Entrega	Início	Fim

Ordens de fabrico

Nº	#Enc	Recurso	Op	Param	Estado

Interações

A figura seguinte ilustra as mensagens trocadas entre cada recurso fabril, o computador local (com a aplicação em VBasic e com o servidor WEB), e o browser remoto a partir do qual era possível monitorizar e controlar o SFP.



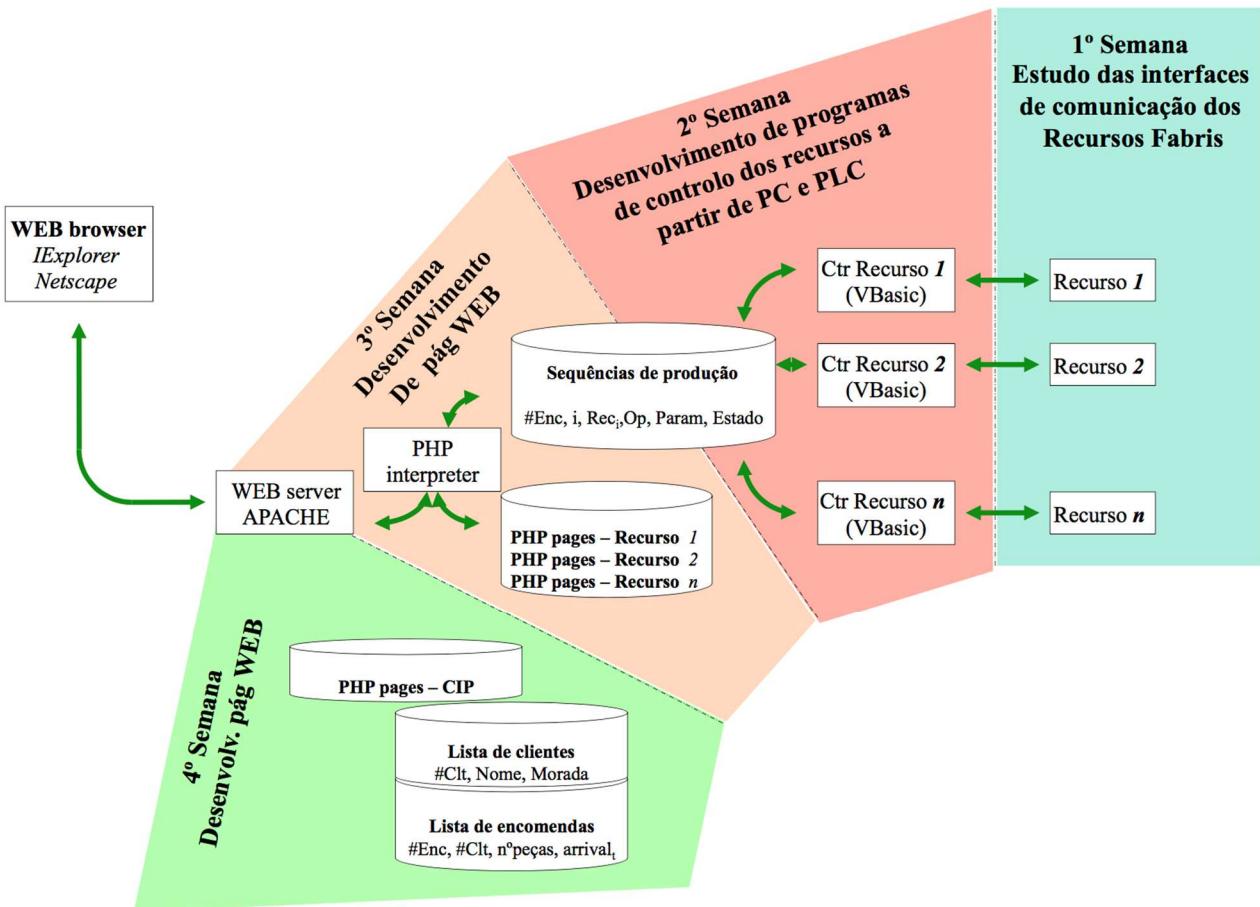
Implementação do SFP

Na primeira semana, cada aluno estudava um dos equipamentos disponíveis no laboratório, em particular a sua interface de comunicação, e desenvolvia em VBasic uma aplicação que permitia o seu controlo local.

Na segunda semana, cada aluno lia numa base de dados, na altura o MsAccess, as ordens de fabrico destinadas ao seu equipamento, e enviava-as sequencialmente para o seu equipamento através da sua interface de comunicação.

Na terceira semana, cada aluno desenvolvia as páginas WEB, em HTML e PHP, para ser possível monitorizar e controlar remotamente o seu equipamento a partir de um browser WEB.

Na quarta semana, todos os trabalhos eram integrados numa página WEB que permita o controlo integrado do SFP.



2.6. Indústria 4.0

Recentemente, o governo Alemão propôs uma estratégia para uma 4^a revolução industrial.

Chancellor Angela Merkel urged the German business elite at the World Economic Forum in Davos in 2015 to act swiftly: »We need to quickly master the amalgamation of the world of the internet with the world of industrial production because the current leaders in the digital area will otherwise take over industrial production.« (Merkel 2015).

WOLFGANG SCHROEDER (2016), Germany's Industry 4.0 strategy

Rhine capitalism in the age of digitalization. Friedrich-EbertStiftung (FES) London 2016

https://www.fes-london.org/fileadmin/user_upload/publications/files/FES-London_Schroeder_Germanys-Industrie-40-Strategy.pdf. Accessed on: 16.05.2018.

BASQUE INDUSTRY 4.0 - KEYNOTE, "Industrie 4.0: The Fourth Industrial Revolution based on Smart Factories", 2014, por Prof. Wolfgang Wahlster – DFKI GmbH

<https://www.youtube.com/watch?v=vWrbBcY2-Ms> Accessed on: 16.05.2018.

- A 1^a Revolução industrial baseou-se no uso de “novos” equipamentos acionados pela máquina a vapor. Foi uma revolução, dado que até então era usado o vento e o fluxo de água dos rios, recursos intermitentes, para accionar os engenhos de produção.
- A 2^a revolução permitiu a produção em massa, e baseou-se em equipamentos acionados por motores elétricos.
- A 3^a revolução foi potenciada pela electrónica em geral, e pelo microprocessador em especial, o que permitiu o aparecimento dos: Computadores, Programmable Logic Controllers (PLC), Comandos numéricos (CN), Robôs, e a automatização em geral.
- A 4^a revolução baseia-se no microcontrolador, “embedded” embutido/inserido em equipamentos e produtos, comunicando em rede (Internet of Things - IOT), e que em conjunto permitem a criação de “cyber-physical production systems”.

Embedded devices: Microcontroladores inseridos em equipamentos e produtos.

Network embedded devices: Embedded devices plus IOT

Cyber-physical: several “Embedded devices plus IOT” embedded in physical equipments, products, things in general. Cyber : computer + internet. Physical: equipments, products.

Big data: refers to the information and communication technologies trend of processing huge amounts of data in order to derive the appropriate data for rapid decision making for increased productivity.

Smart factory:

- *M2M*: Machine to Machine communications
- *P2M*: Product to machine communications.
- *Tag*: Embedded digital memories in products + RFID
- *MES* : Manufacturing Execution Systems

INDUSTRIE 4.0 is the name given to the German strategic initiative to establish Germany as a lead market and provider of advanced manufacturing solutions.

Indústria 4.0 -Exemplo 1: Fábrica de cereais Muesli (Prof. Wolfgang Wahlster)

O cliente pode pela internet, a partir do seu telemóvel, encomendar os cereais com a composição que quiser de entre 80 ingredientes diferentes (ex: nozes, banana, avelã) e recebe pelo correio a sua encomenda. A caixa “viaja” pela fábrica e “pede” a cada máquina a quantidade que precisa de cada ingrediente.



CAPÍTULO

VBASIC

2023

3. INTRODUÇÃO AO VISUAL BASIC

Longe vão os dias em que era necessário usar um editor de texto genérico para escrever o código e para o gravar num ficheiro em disco. Depois era necessário chamar um programa (compilador) que abria esse ficheiro de texto e gerava um segundo ficheiro com o código máquina (obj), depois o programador necessitava de chamar um terceiro programa “Linker” para juntar os vários ficheiros de código máquina gerados pelo compilador, com outros ficheiros com código máquina (lib) fornecidos pela empresa do “compilador”, para que finalmente fosse possível gerar em disco um ficheiro executável que fazia o que o programador pretendia. (quando tudo corria bem...)

Hoje em dia existem programas comerciais, como o *Visual Basic 2022* que integram todas estas funcionalidades, permitindo que o programador a partir de um ambiente integrado de desenvolvimento (IDE), possa editar vários ficheiros com o código fonte, associá-los a um mesmo projeto, compilar todos estes ficheiros, linkar e executar o programa sempre no mesmo ambiente. (*secção 2.2 Ambiente Integrado de desenvolvimento*).

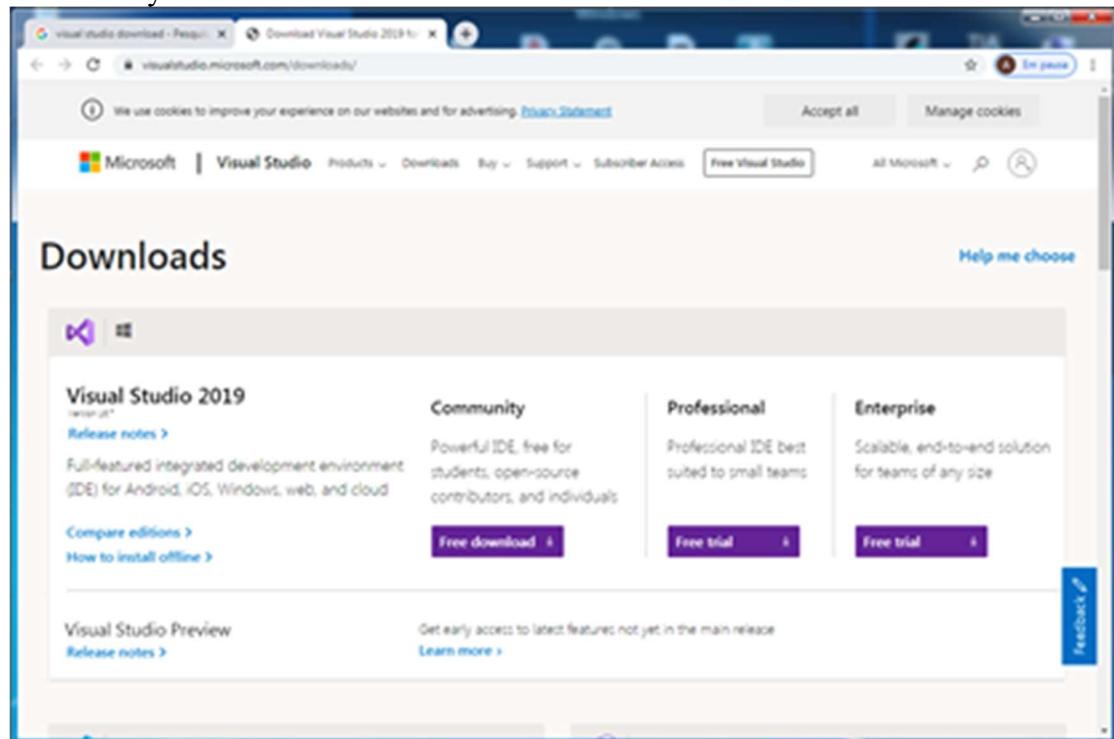
Por outro lado, todos os programas desenvolvidos por um programador necessitam de ter uma interface para que o utilizador possa interagir com o programa e visualizar os resultados. Uma das maiores vantagens do Visual Basic é a enorme facilidade e rapidez com que um programador pode criar novos programas com interfaces gráficas amigáveis (*secção 2.3 Construção da interface gráfica*). Naturalmente que a interface gráfica só por si não chega, é necessário saber desenvolver o código necessário para processar os dados recebidos e gerar os resultados (*secção 2.4 Linguagem de programação*)

3.1. Instalar o Visual Basic 2022

O **Visual Basic 2022** é gratuito e pode ser transferido a partir do sítio da Microsoft

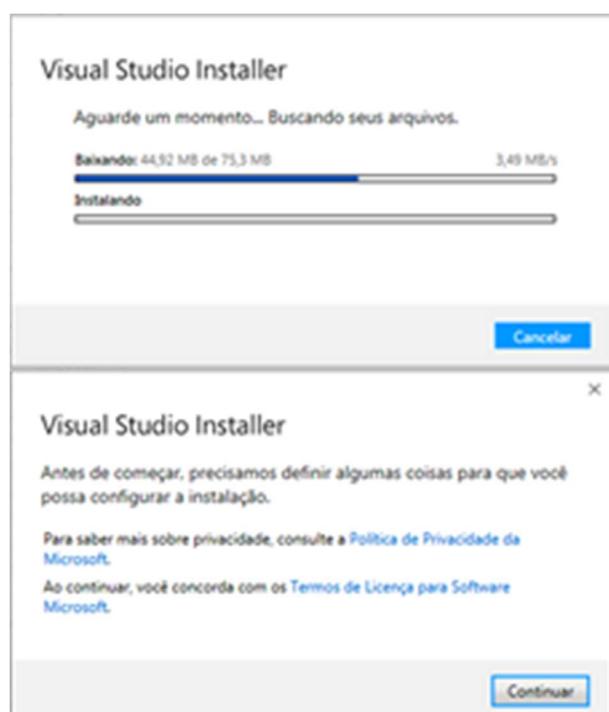
- Para instalar o visual studio utilizar o link:
<https://visualstudio.microsoft.com/downloads/>

- Selecionar a opção:
Community

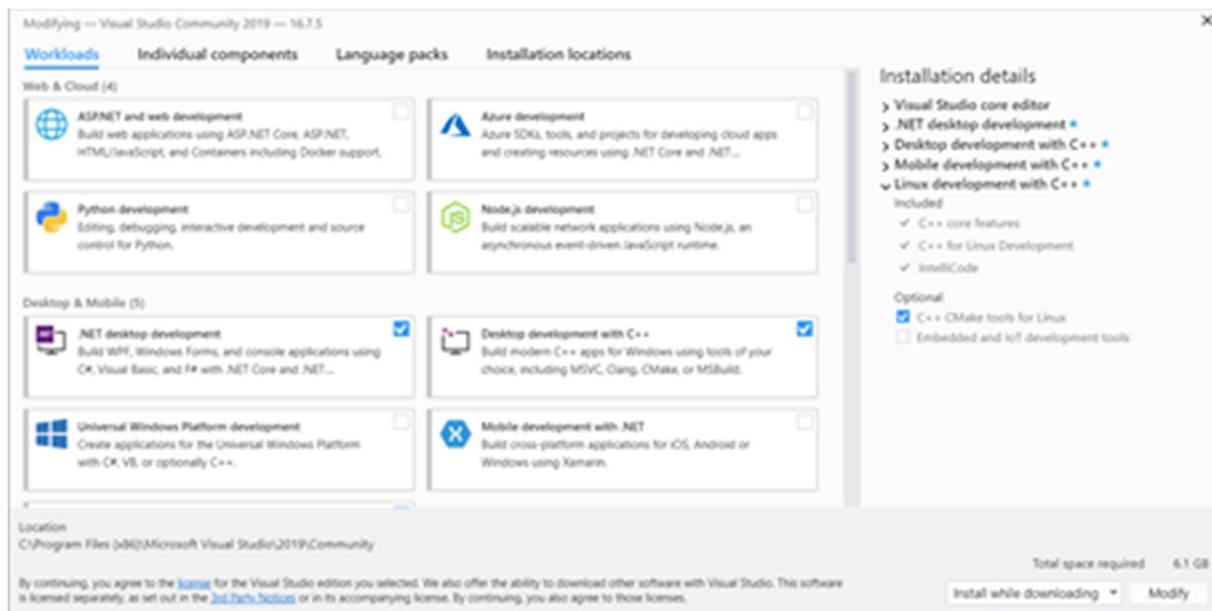


- É realizado o download do ficheiro:
vs_community_1801534470.1565969460.exe

- Aceder à pasta de transferências e Executar o ficheiro anterior, como administrador;



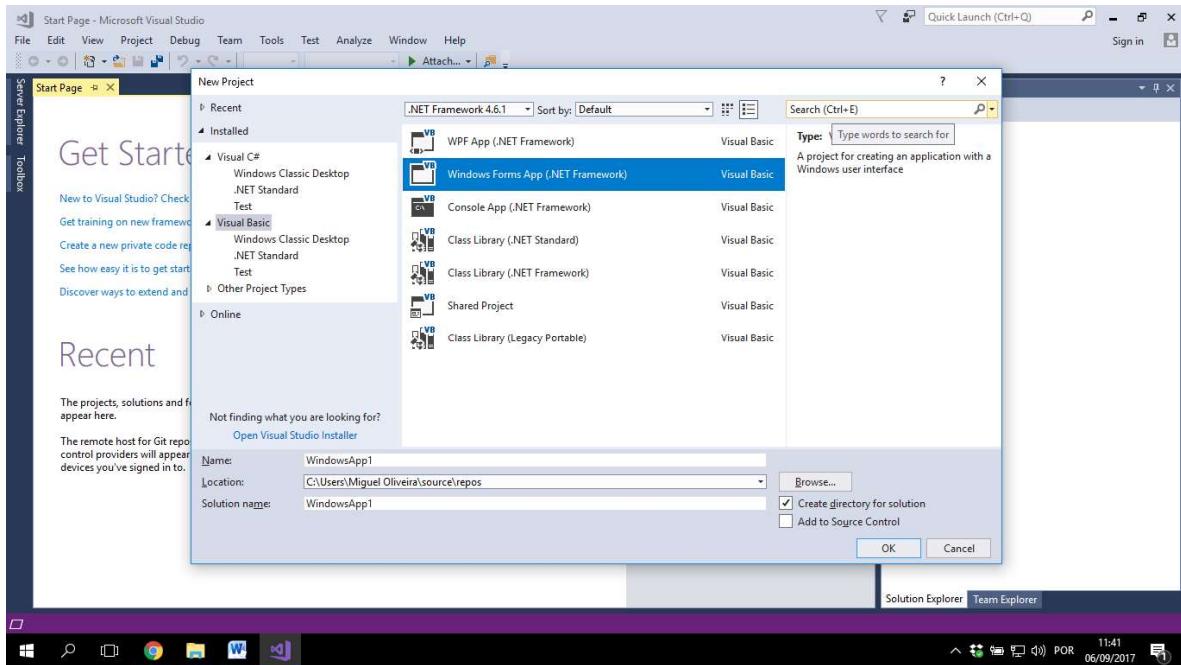
Selecionar a opção indicada .NET, instalando suporte para diferentes linguagens de programação: C#, Visual Basic, F# e C++;



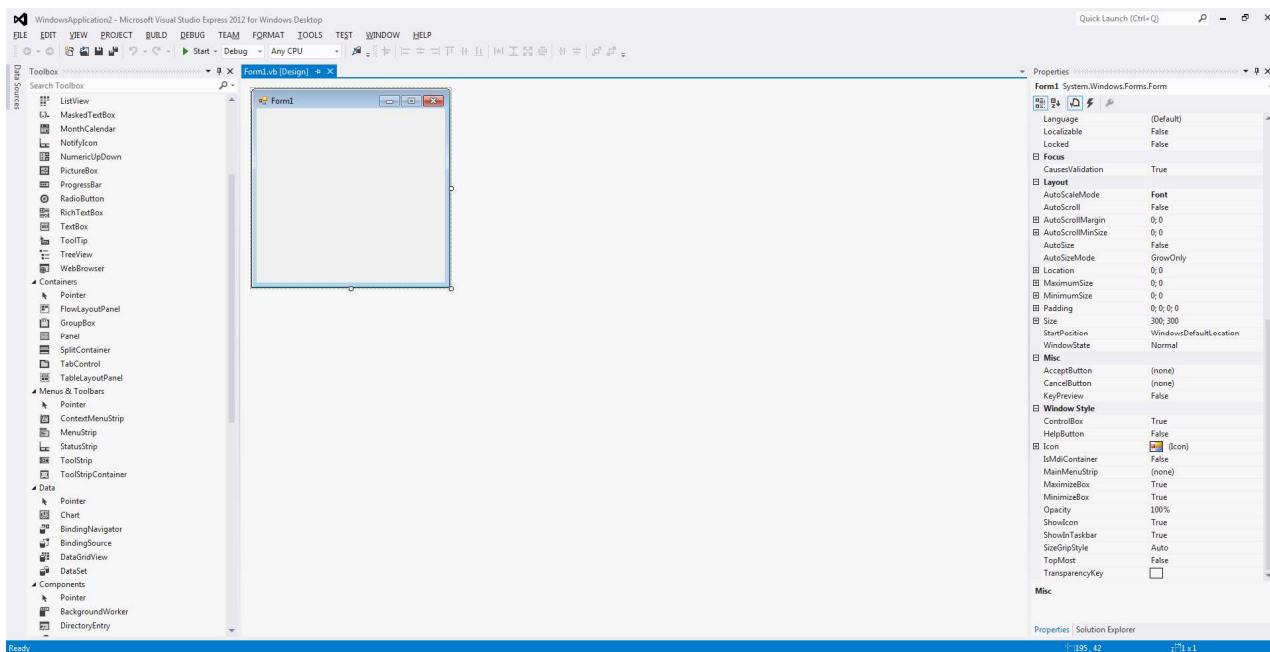
Na disciplina de Informática Industrial vamos utilizar apenas a linguagem de programação: Visual Basic;

3.2.Ambiente integrado de desenvolvimento

Para criar um novo projeto que permita desenvolver uma aplicação Windows, com interface, deve selecionar a opção “File – New Project”, selecionar a linguagem Visual Basic e a subopção “Windows Forms App” como a figura seguinte ilustra.



A figura seguinte apresenta a janela inicial do ambiente integrado de desenvolvimento IDE. Ao centro aparece a janela **Form1**, à esquerda aparece a janela **Toolbox** onde é possível ir buscar os objetos gráficos para colocar na janela Form1 (botões, caixas de texto,), à direita aparece a janela das **Properties**.



Todos os objetos gráficos (controlos) que colocarmos no Form, e o próprio Form, têm propriedades (cor, localização no ecrã, nome,...). Sempre que seleccionarmos um objecto do **Form** “clicando” sobre ele, a janela **Properties** apresenta as propriedades do objecto (controlo) selecionado.

Se alguma destas janelas não aparecer no écran, pode selecionar o menu ***View***, a opção ***OtherWindows*** e depois selecionar a(s) janelas que pretender ver : ***Toolbox***, ***Solution Explorer***, ***Properties Window***, ...

Para mais informações sobre o IDE deve consultar o livro recomendado.

3.3.Construção da interface gráfica dos programas

Uma das maiores vantagens do *VBasic* reside precisamente na facilidade e rapidez com que é possível desenvolver uma interface gráfica para o nosso programa.

Na janela “*Toolbox*” temos à disposição muitos objetos “*Controlos*” que podemos “arrastar” para a janela “*Form*”. É esta janela que o utilizador irá ver, quando executar o nosso programa.

Nesta secção iremos estudar vários controlos, este estudo baseia-se no princípio que “uma imagem vale mil palavras” por isso iremos analisar vários exemplos práticos. Será apresentado o código correspondente a cada exemplo, e estudado cada um dos controlos usados nesses exemplos.

No final desta secção, depois de realizar todos os exemplos, o leitor terá implementado um editor de texto com várias janelas/Forms (interfaces) e terá usados todos os controlos que a seguir de enumeram: *Form*, *Label*, *TextBox*, *Button*, *RadioButton*, *PictureBox*, *Combobox*, *OpenFileDialog*, *SaveFileDialog*, *StreamWriter*, *StreamReader*.

3.3.1. Exemplo “Calculadora_soma”

Vamos começar por um exemplo simples, intitulado *Calculadora_Soma*. Este programa permite calcular a soma de dois valores. O utilizador introduz dois valores, um em X, o outro em Y e quando premir o botão vermelho (+) aparece o resultado da soma em Z (figura seguinte).

Nesta interface *Form1* o programador utiliza três controlos do tipo *Label*, três controlos do tipo *Textbox* e um controlo do tipo *Button*.



*'Nesta classe é definido todo o código necessário para a execução da interface (Janela) Form1
Public Class Form1*

*' A subrotina Form1_Load é executada automaticamente, uma vez, quando a janela Form1 aparece no ecrã
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load*

```
Me.Text = "Calculadora"           ' A palavra Calculadora aparece na parte de cima da janela Form1
Me.Height = 200                  ' A janela Form1 ocupa 200 por 500 pixéis no ecrã
Me.Width = 500
Me.BackgroundImage = Image.FromFile("c:\windows\coffeebean.bmp") ' Esta passa a ser a imagem de fundo da
janela Form1
Me.Icon = New Icon("Alarm.ico")    ' Esta imagem "icon" aparece no canto superior da janela Form1
```

' A caixa de texto TextBox1, o seu canto superior esquerdo, tem as coordenadas X=50,Y=40

```
TextBox1.Location = New Point(50, 40)      ' A Textbox1 ocupa 100 por 30 pixéis dentro da Janela Form1
TextBox1.Width = 100
TextBox1.Height = 30
```

```

TextBox2.Location = New Point(50, 120)
TextBox2.Width = 100
TextBox2.Height = 30

TextBox3.Location = New Point(300, 75)
TextBox3.Width = 100
TextBox3.Height = 30

Button1.Text = "+"
Button1.BackColor = Color.Red      ' O botao passa a ter a cor de fundo vermelha
Button1.ForeColor = Color.Yellow   ' e o texto a amarelo
Button1.Width = 100
Button1.Height = 40
Button1.Visible = True

Label1.Text = "X"
Label1.BackColor = Color.Yellow    ' A cor de fundo do Label1 passa a amarelo
Label1.ForeColor = Color.Black     ' e o texto que aparece no Label1, neste caso a letra X, passa preto
Label1.Width = 100
Label1.Height = 40
Label1.Location = New Point(50, 20)
Label1.Visible = True

Label2.Text = "Y"
Label2.BackColor = Color.Yellow
Label2.ForeColor = Color.Black
Label2.Width = 100
Label2.Height = 40
Label2.Location = New Point(50, 100)
Label2.Visible = True

Label3.Text = "Z"
Label3.BackColor = Color.Yellow
Label3.ForeColor = Color.Black
Label3.Width = 100
Label3.Height = 40
Label3.Location = New Point(300, 55)
Label3.Visible = True
End Sub

```

```

' Quando o utilizador do programa permitir o botao (+) da interface Form1 esta subrotina será executada
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    ' Os conteúdos das caixas de texto da esquerda são convertidos em números inteiros
    ' e o resultado da sua soma é visulizado na caixa de texto da direita (textbox3)
    TextBox3.Text = CStr(CInt(TextBox1.Text) + CInt(TextBox2.Text))
End Sub

```

End Class

Depois de apresentarmos a interface e o código, vamos analisar com maior detalhe as propriedades de cada um dos controlos/objetos utilizados na interface.

Objeto “Form1”

Altere o titulo da janela “Form1” para “Calculadora”	Form1.Text = "Calculadora"
Altere a altura da janela “Form1” para 200 pixeis	Form1.Height = 200
Altere a largura da janela “Form1” para 500 pixeis	Form1.Width = 500
Altere a imagem do canto superior esquerdo da janela Form1	Form1.Icon = New Icon("Calc.ico")

Altere a imagem de fundo da janela Form1 **Form1.BackgroundImage = Image.FromFile("C:\windows\café.bmp")**

Objeto “Label”

Arraste um controlo do tipo *“Label”* para a janela *“Form1”*. A partir desse momento passa a ter um objeto *“Label1”* visível na interface. Este controlo tem o aspecto de uma linha de texto, permite ao utilizador ler as mensagens de texto que o programador previamente definiu.

Escreva a mensagem <i>“Caixa de texto”</i> na caixa de texto	Label1.text = "X"
Altere a cor de fundo da janela para amarelo	Label1.BackColor = Color.Yellow
Altere a cor de fundo da janela para amarelo	Label1.BackColor = Color.Grey
Altere a altura da caixa de texto para 20 pixels	Label1.height = 50
Altere a largura da caixa de texto para 20 pixels	Label1.Width = 100
Altere o nome deste objeto para <i>“txtJanela”</i>	Label1.Name = "lbX"

Objeto “TextBox”

Arraste um controlo do tipo *“TextBox”* para a janela *“Form1”*. A partir desse momento passa a ter uma caixa de texto com o nome *“TextBox1”*. Este controlo tem o aspecto de uma caixa de texto, permite ao utilizador e ao programador escrever ou ler mensagens de texto nela.

Escreva a mensagem <i>“Caixa de texto”</i> na caixa de texto	TextBox1.text = "Caixa de texto"
Altere a cor de fundo da janela para amarelo	TextBox1.BackColor = Color.Yellow
Adicione barras de deslocamento Horiz,Vert <i>“ScrollBar”</i>	TextBox1.ScrollBars = ScrollBars.Both
Janela com várias linhas	TextBox1.Multiline = True
Altere a localização da textbox	TextBox1.Location = New Point(50,40)
Altere a altura da caixa de texto para 20 pixels	TextBox1.height = 50
Altere a largura da caixa de texto para 20 pixels	TextBox1.Width = 100
Alinhe ao centro o texto que será escrito na caixa	TextBox1.TextAlign= HorizontalAlignment.Center
Altere o nome deste objeto para <i>“txtJanela”</i>	TextBox1.Name = "txtJanela"

Objeto “Button”

Arraste um controlo do tipo *“Button”* para a janela *“Form1”*. A partir desse momento passa a ter um botão com o nome *“Button1”*. Este controlo permite ao utilizador *“clicar”* sobre o botão fazendo com que o código associado a este botão seja executado.

Escreva a mensagem <i>“Soma”</i> na propriedade	Button1.text = "Soma "
Altere a cor de fundo do botão para amarelo	Button1.BackColor = Color.Yellow
Altere a cor da letra para azul	Button1.ForeColor = Color.Blue

Altere a altura da caixa de texto para 20 pixels	Button1.height = 50
Altere a largura da caixa de texto para 20 pixels	Button1.Width = 100
Altere o nome deste objeto para "BtSoma"	TextBox1.Name = "BtSoma"

3.3.2. Exemplo “Seleção de imagens”

Neste exemplo, o utilizador pode selecionar uma de várias imagens na interface **Form1** do programa. O utilizador selecciona um de três **Radiobutton**, fazendo aparecer a imagem respectiva na **Picturebox** existente na janela.



Public Class Form1

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    RadioButton1.Location = New Point(50, 50)
    RadioButton1.Width = 50
    RadioButton1.Height = 25
    RadioButton1.Text = "Bolhas"           ' Bolhas é o nome que o utilizador vê

    With RadioButton2
        .Location = New Point(50, 70)
        .Width = 50
        .Height = 25
        .Text = "Pedra"
        .Select()                      ' Este radiobutton fica selecionado
        .BackColor = Color.Aquamarine   ' Esta é a cor de fundo do Radiobutton2
    End With

    With RadioButton3
        .Location = New Point(50, 90)
        .Width = 50
        .Height = 25
        .Text = "Salmao"
    End With

    PictureBox1.Location = New Point(200, 20)
    PictureBox1.Anchor = AnchorStyles.Left
    PictureBox1.BackColor = Color.AliceBlue
    PictureBox1.BorderStyle = BorderStyle.Fixed3D

End Sub
```

' Este subprocedimento é executado quando o utilizador selecionar o Radiobutton1

```
Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ...
    ... RadioButton1.BackColor = Color.Aquamarine          ' A cor de fundo do radiobutton1 passa a ser Aquamarine
```

```

PictureBox1.BackgroundImage = Image.FromFile("c:\windows\Sop Bubbles.bmp") ' Na picturebox1 aparece a
imagem Soap Bubbles
End Sub

Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
...
    RadioButton2.BackColor = Color.Aquamarine
    PictureBox1.BackgroundImage = Image.FromFile("c:\windows\Greenstone.bmp")
End Sub

Private Sub RadioButton3_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
...
    RadioButton3.BackColor = Color.Aquamarine
    PictureBox1.BackgroundImage = Image.FromFile("c:\windows\santa fe stucco.bmp")
End Sub

End Class

```

Objeto RadioButton

Arraste um controlo do tipo “RadioButton” para a janela “Form1”. A partir desse momento passa a ter um botão com o nome “RadioButton1”. Este controlo permite ao utilizador “clicar” sobre o botão fazendo com que o código associado a este botão seja executado.

Escreva a mensagem “Soma” na propriedade	RadioButton1.text = “Bolhas ”
Altere a cor de fundo do botão para amarelo	RadioButton1.BackColor = Color.Yellow
Altere a cor da letra para azul	RadioButton1.ForeColor = Color.Blue
Altere a altura da caixa de texto para 20 pixels	RadioButton1.height = 50
Altere a largura da caixa de texto para 20 pixels	RadioButton1.Width = 100
Altere a largura da caixa de texto para 20 pixels	RadioButton1.Position = new Point(200,20)
Altere a largura da caixa de texto para 20 pixels	RadioButton1.Width = 100
Altere o nome deste objeto para “BtSoma”	RadioButton1.Name = "RbBolhas"

Objeto PictureBox

Arraste um controlo do tipo “PictureBox” para a janela “Form1”. A partir desse momento passa a ter um botão com o nome “PictureBox1”. Este controlo permite ao utilizador “clicar” sobre o botão fazendo com que o código associado a este botão seja executado.

Altere a cor de fundo do botão para amarelo	PictureBox1.BackColor = Color.Yellow
Altere a altura da caixa de texto para 20 pixels	PictureBox1.height = 50
Altere a largura da caixa de texto para 20 pixels	PictureBox1.Width = 100
Altere a posição do canto superior esquerdo	PictureBox1.Location = new Point(200,20)

A picturebox acompanha o lado esquerdo PictureBox1.Anchor = AnchorStyles.Left do form

Altere a moldura exterior da picture box PictureBox1.BorderStyle = BorderStyle.Fixed3D

Altere a imagem presente na picturebox PictureBox1.BackGroundImage=Image.FromFile("c:\windows\Gre.bmp")

Altere a imagem presente na picturebox PictureBox1.BackGroundImageLayout=ImageLayout.Stretch

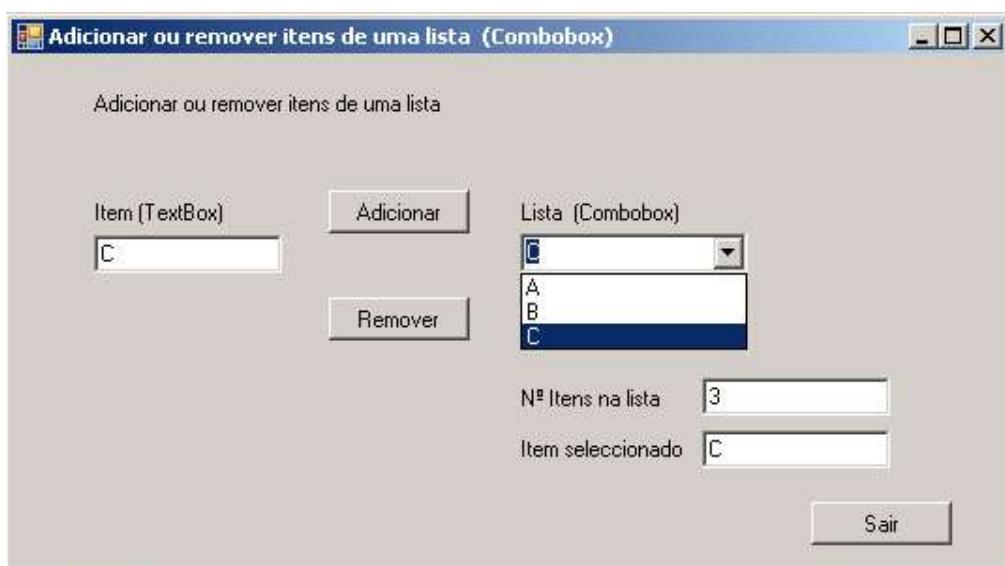
Altere o nome deste objeto para PictureBox1.Name = "pBBolhas"
"pBBolhas"

3.3.3. Exemplo “Lista de String”

Neste exemplo pretende-se apresentar o controlo **ComboBox**.

O texto que o utilizador introduzir na **TextBox** da esquerda será adicionado ou removido da **ComboBox** da direita quando o utilizador premir o botão adicionar ou remover, respectivamente.

Nesta interface **Form1** o programador utiliza cinco controlos do tipo **Label**, três controlos do tipo **Textbox**, três controlos do tipo **Button** e um controlo do tipo **ComboBox**.



Public Class Form1

' Quando o utilizador clicar no botao Adicionar esta subrotina é chamada

```
Private Sub BtAdicionar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtAdicionar.Click
    ComboBox1.Items.Add(txtItem.Text)
    txtNItens.Text = ComboBox1.Items.Count
End Sub
```

' adiciona à combobox o texto presente na textbox
' visualiza na textbox o nº de linhas/itens presentes na combobox

' Quando o utilizador clicar no botao Remover esta subrotina é chamada

```
Private Sub btRemover_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btRemover.Click
    ComboBox1.Items.Remove(txtItem.Text)
    txtNItens.Text = ComboBox1.Items.Count
End Sub
```

' remove da combobox o item/linha/texto presente na textbox

' Quando o utilizador selecionar na combobox uma nova linha esta subrotina é chamada

```
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles ...
    txtItemSeleccionado.Text = ComboBox1.SelectedItem ' Visualiza a linha/item selecionado pelo utilizador
End Sub
```

```

' Esta subrotina é chamada quando o utilizador selecionar o botão Sair
Private Sub BtSair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtSair.Click
    End
End Sub

End Class

```

Objeto ComboBox

Arraste um controlo do tipo “*ComboBox*” para a janela “Form1”. A partir desse momento passa a ter um objeto com o nome “*ComboBox1*”. Este controlo permite ao programador criar uma lista de itens/texto e ao utilizador ler os itens e selecionar um deles.

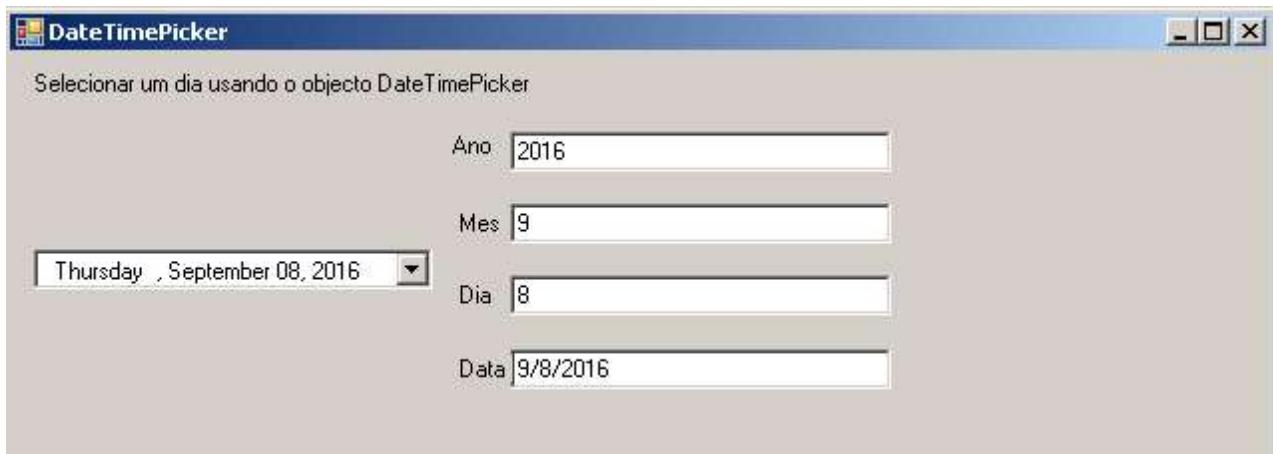
Escreva a mensagem “ <i>ListaItems</i> ” na propriedade text	ComboBox1. text = "Lista Items "
Altere a cor de fundo do botão para amarelo	ComboBox1. BackColor = Color.Yellow
Altere a cor da letra para azul	ComboBox1. ForeColor = Color.Blue
Altere a altura da caixa de texto para 20 pixeis	ComboBox1. height = 50
Altere a largura da caixa de texto para 20 pixeis	ComboBox1. Width = 100
Devolve o número de itens presentes na combobox	ComboBox1. Items.Count
Adiciona um novo Item/linha à combobox	ComboBox1. Items.Add (“A”)
Remove um item à combobox	ComboBox1. Items.remove (“A”)
Remove todos os itens da combobox	ComboBox1. Clear ()
Devolve o item que o utilizador selecionou	ComboBox1. SelectedItem
Altere o nome deste objeto para “Cb_Letras”	ComboBox1. Name = "CB_Letras"

3.3.4. Exemplo “Calendário”

Neste exemplo pretende-se apresentar o controlo **DateTimePicker**.

Quando o utilizador selecionar uma nova data, usando o controlo **DateTimePicker** (à esquerda na figura), o sub procedimento **DateTimePicker1_ValueChanged** será executado e nas textbox da direita aparecerá o ano, o mês , o dia e a data que o utilizador selecionou.

Nesta interface o programador utiliza um controlo do tipo **DateTimePicker**, quatro controlos tipo **Label**, e quatro controlos do tipo **Textbox** .



Public Class Form1

' Quando o utilizador selecionar na DateTimePicker uma nova data esta subrotina é chamada

Private Sub DateTimePicker1_ValueChanged(ByVal sender

```
txtAno.Text = DateTimePicker1.Value.Year  
txtMes.Text = DateTimePicker1.Value.Month  
txtDia.Text = DateTimePicker1.Value.Day  
txtData.Text = DateTimePicker1.Value.Date
```

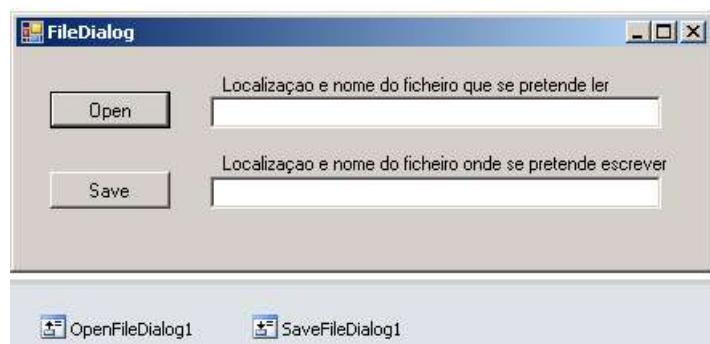
```
End Sub  
End Class
```

3.3.5. Exemplo “Selecionar um ficheiro no disco”

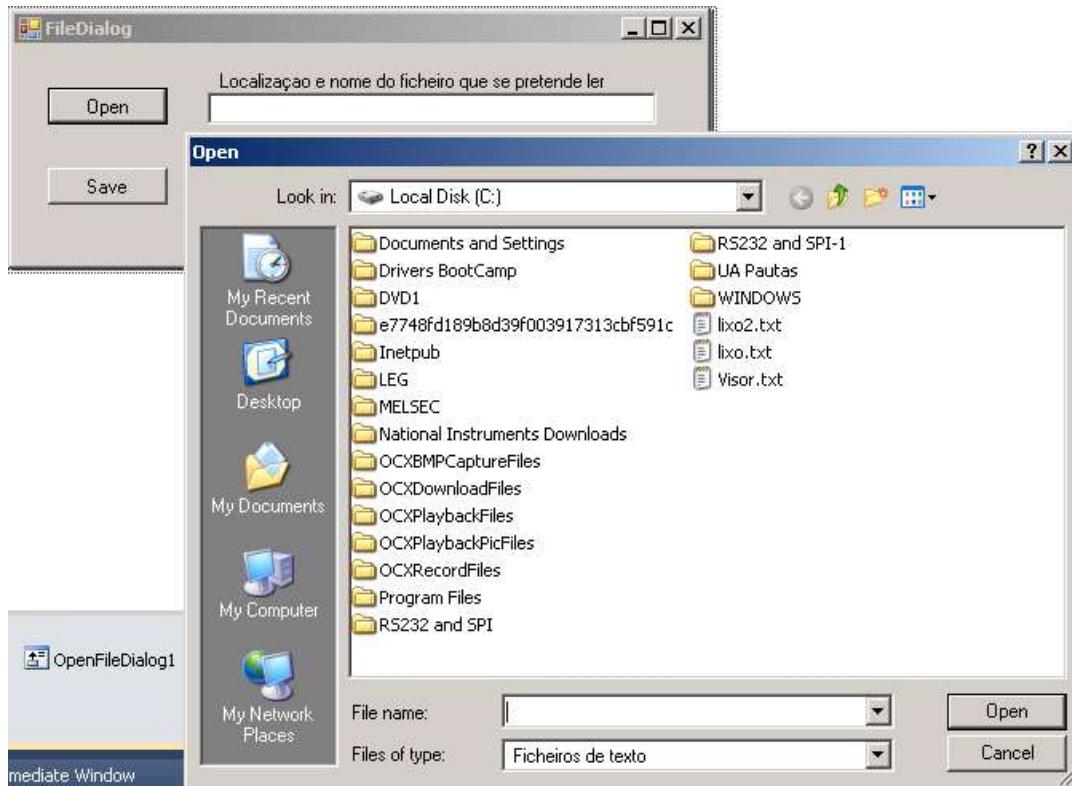
Neste exemplo pretende-se apresentar os controlos **SaveFileDialog** e **OpenFileDialog**.

Ao contrário do que o nome deste tipo de objetos daria a entender, eles não permitem gravar ou abrir ficheiros em disco, permitem apenas que o utilizador veja o conteúdo do disco e selecione o diretório e o nome do ficheiro que prenda posteriormente abrir ou onde pretende gravar dados.

Neste exemplo, o programador utiliza dois controlos do tipo **Label**, dois controlos do tipo **Textbox**, dois controlos do tipo **Button**, um controlo do tipo **SaveFileDialog** e outro do tipo **OpenFileDialog**.



Quando o utilizador pressionar o botão “Open”, o método ShowDialog() do objeto OpenFileDialog é executado e surge no ecrã uma janela de diálogo que permite ao utilizador navegar no disco, visualizar os diretórios e ficheiros existentes.



Quando o utilizador selecionar um dos ficheiros existentes, a sua localização e nome ficam guardados na propriedade **FileName** do objeto **OpenFileDialog1** e essa localização e nome podem ser por exemplo visualizadas numa textbox (TextBox1.Text = OpenFileDialog1.FileName).

Se o utilizador sair da janela de diálogo sem selecionar um ficheiro “Cancel”, a função OpenFileDialog1.ShowDialog() retorna o número dois, o que poderá ser visualizado numa textbox (TextBox1.Text = OpenFileDialog1.ShowDialog()).

Public Class Form1

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
' Filter: Quando a Janela de diálogo aparecer ao utilizador
'serão mostrados apenas os ficheiros com a extensão *.bmp
OpenFileDialog1.Filter = "Ficheiros de imagem|*.bmp|*.jpg|*.jpeg"
OpenFileDialog1.FileName = ""
```

End Sub

```
Private Sub BtOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtOpen.Click
Dim ii As Integer
```

```
' Mostra ao utilizador a janela de diálogo
' Na janela de diálogo o utilizador pode retornar "Cancel"
'sem selecionar nenhum ficheiro.
'Se o ficheiro foi seleccionado a função retorna 1 caso contrário retorna 2
ii = OpenFileDialog1.ShowDialog() ' ii=2 - cancel ii = 1 - open
```

```
' Quando o utilizador sai da janela de diálogo,
'o nome e a localização do ficheiro seleccionado
'está memorizado na propriedade "FileName"
TextBox1.Text = OpenFileDialog1.FileName
```

```
' Se um ficheiro imagem foi seleccionado
If ii <> 2 Then
    PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)
```

End If

End Sub

Objeto OpenFileDialog1

Arraste um controlo do tipo “*OpenFileDialog*” para a janela “Form1”. A partir desse momento passa a ter um objeto com o nome “*OpenFileDialog1*”. Este controlo permite ao programador navegar no disco e visualizar os diretórios e ficheiros existentes. Depois da Janela de diálogo estar aberta *OpenFileDialog1.ShowDialog()* o utilizador pode selecionar um ficheiro ou cancelar e sair da janela. Se sair da janela depois de ter selecionado um ficheiro esta função retorna o número 1 e o nome e a localização do ficheiro é guardada na propriedade *FileName*, se o utilizador sair da janela de diálogo fazendo “cancel” esta função retorna o número 2.

Quando a janela de diálogo ficar visível apenas mostra os *OpenFileDialog1.Filter = "bmp|*.bmp | jpg|*.jpg"* ficheiros com a(s) extensões indicadas

Mostra a janela de diálogo ao utilizador e retorna o número *TextBox1.Text = OpenFileDialog1.ShowDialog()* 1 ou o número 2 quando o utilizar sair da janela

Depois do utilizador ter selecionado um ficheiro e saído da *textBox1.Text = OpenFileDialog1.FileName* janela, contém o nome e localização do ficheiro selecionado.

Crie o botão **Save** na interface (**Form**) e o código respectivo **PrivateSub BtSave....**

3.3.6. Exemplo “Ler e gravar ficheiros de texto”

Neste exemplo pretende-se abrir ficheiros de texto e visualizar o seu conteúdo numa caixa de texto. Pretende-se também gravar uma string num ficheiro texto.

StreamWriter e StreamReader

Os objetos do tipo **StreamReader** e **StreamWriter** permitem, respectivamente, ler ou escrever em ficheiros de texto.

Para poder usar objetos do tipo *StreamReader* e *StreamWriter* deve importar o “*System.IO*”, acrescentando no seu código:

Imports System.IO

Para escrever o texto “Ola” no ficheiro “c:\exemplo.txt”, deve declarar um objeto do tipo *StreamWriter*, por exemplo com o nome “**ficheiroguardar**”, fazendo:

Dim ficheiroguardar as StreamWriter

Para guardar o texto “Ola” no ficheiro “c:\exemplo.txt”, apagando o que lá estava antes, deve fazer:

```
ficheiroguardar = new StreamWriter ("c:\exemplo.txt", False )  
ficheiroguardar.write("Ola")
```

Para acrescentar o texto “Ola”, **mantendo** o que estava antes, deve fazer:

```
ficheiroguardar = new StreamWriter ("c:\exemplo.txt", True )  
ficheiroguardar.write("Ola")
```

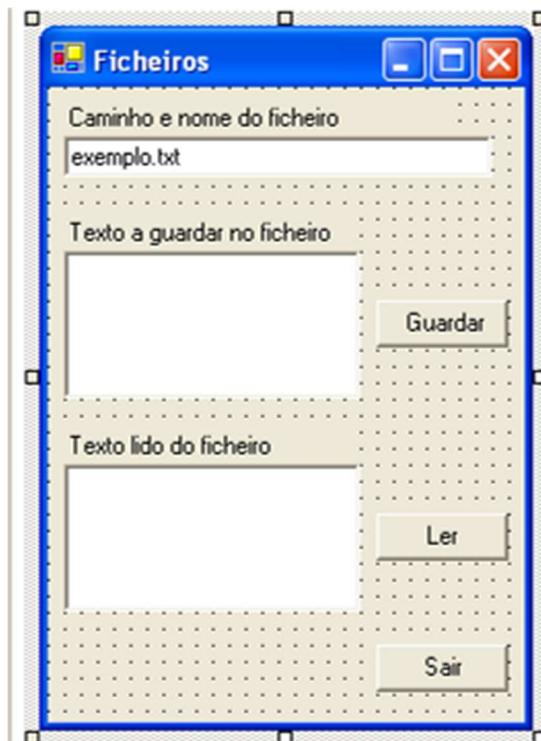
Finalmente, para que o texto seja de facto escrito em disco deve fazer:

```
ficheiroguardar.close()
```

Para ler o conteúdo de um ficheiro de texto, por exemplo, o conteúdo do ficheiro “c:\exemplo.txt”, deve declarar um objeto do tipo StreamReader, por exemplo com o nome “**ficheiroler**”, fazendo:

```
Dim ficheiroler as StreamReader  
ficheiroler = new StreamReader("c:\exemplo.txt")  
Textbox1.text = ficheiroler.readtoend()  
ficheiroler.close()
```

Após esta apresentação dos objetos do tipo StreamReader e StreamWriter será fácil perceber o exemplo seguinte:



```
□ 'O namespace System.IO fornece as classes
└ necessárias para a manipulação de ficheiros
  Imports System.IO

□ Public Class Form1
  Inherits System.Windows.Forms.Form

  □ Windows Form Designer generated code

  □
    Private Sub btnGuardar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGuardar.Click
      Dim ficheiroGuardar As StreamWriter = New StreamWriter(txtPath.Text, False)
      ficheiroGuardar.WriteLine(txtDadosGuardar.Text)
      ficheiroGuardar.Close()
    End Sub

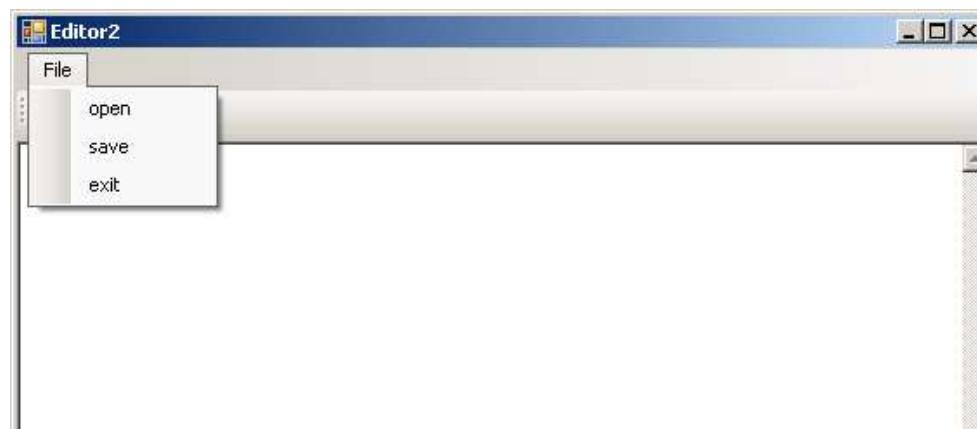
  □
    Private Sub btnAbrir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAbrir.Click
      Dim ficheiroLer As StreamReader = New StreamReader(txtPath.Text)
      txtDadosLer.Text = ficheiroLer.ReadToEnd()
      ficheiroLer.Close()
    End Sub

  □
    Private Sub btnSair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSair.Click
      End
    End Sub

  End Class
```

3.4. Trabalho nº 1 - “Editor de texto”

Pretende-se desenvolver um pequeno editor de texto que permita abrir e gravar ficheiros utilizando os objetos do tipo **MenuStrip**, **StatusStrip**, **OpenFileDialog**, **SaveFileDialog**, **StreamReader** e **StreamWriter**.



```
Imports System.IO
Public Class Editor2
    Dim le As StreamReader
    Dim escreve As StreamWriter

    Private Sub OpenToolStripMenuItem_Click(ByVal sender As System.Object,
        Dim s As String
        s = OpenFileDialog1.ShowDialog()

        If s = 1 Then
            le = New StreamReader(OpenFileDialog1.FileName)
            TextBox1.Text = le.ReadToEnd
            le.Close()
        End If

    End Sub

    Private Sub SaveToolStripMenuItem_Click(ByVal sender As System.Object,
        Dim s As String

        s = SaveFileDialog1.ShowDialog()
        If s = 1 Then
            escreve = New StreamWriter(SaveFileDialog1.FileName)
            escreve.Write(TextBox1.Text)
            escreve.Close()
        End If

    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object,
        End
    End Sub
```

3.5. Linguagem de programação

Mesmo depois da interface gráfica de um novo programa estar concebida e implementada, o nosso programa tem de ser capaz de processar os dados recebidos através dessa interface.

Tal como todas as linguagens de programação o VBasic permite ao programador criar variáveis, usar operações matemáticas, operações booleanas, instruções de controlo de fluxo e também agrupar o seu código em funções ou procedimentos.

3.5.1. Bases numéricas, prefixos

Em VBasic, o programador tem a liberdade de escrever os números em várias bases numéricicas: Decimal, Octal, Hexadecimal mas tem de indicar ao compilador em qual das bases escreve um determinado número. Para isso tem de colocar um prefixo **&H** ou **&O** antes do número

Hexadecimal	&H9f	' O número hexadecimal 9f corresponde ao número decimal 159
Octal	&O65	' O número octal 65 corresponde ao número decimal 53

Por defeito, se nada for dito ao compilador, ele assume que o número foi escrito na base decimal. Mas, mesmo na base decimal um número pode ser positivo ou negativo, pode ser pequeno ou grande e necessitar de vários bytes em memória para o armazenar, pode também ser inteiro ou real. Em VBasic, o programador pode indicar ao compilador de que forma um número deve ser considerado escrevendo a seguir ao número as letras correspondentes ao tipo de número. Por exemplo, o programador pode querer que o número 100, escrito na base decimal, seja tratado como um número inteiro ou como um número real (com vírgula):

Números inteiros		
<i>Inteiro sem sinal</i>	100UI	'Unsigned Int
<i>Inteiro curto</i>	100S	'Short
Números reais		
<i>Single</i>	100F	'Float
<i>Double</i>	100R	'Real

3.5.2. Variáveis

Para o programa poder guardar na memória RAM do computador: números, texto e dados em geral o programador pode declarar diferentes tipos de variáveis. Estas variáveis podem ser escalares, matriciais ou estruturas complexas.

Byte, Integer, Decimal, Long, Boolean, String, Variant

Em VBasic, o programador tem vários tipos de variáveis à disposição, dependendo do tipo de valores que pretender guardar pode usar variáveis do tipo “char”, “integer” e outros tipos.

Uma variável do tipo “*Byte*” ocupa 8 bits na memória do computador. Por isso, estas variáveis podem conter números positivos de 0 a 255 ou, números negativos e positivos de -128 a 127 “*Signed Byte - SByte*”.

O bit mais significativo das variáveis pode ser usado para representar o sinal (-). Quando este bit vale “1” o número é negativo. Quando as variáveis são do tipo “*unsigned*”, sem sinal, todos os bits, mesmo o mais significativo é usado para representar um número positivo.

Uma variável do tipo “*char*” ocupa 32 bits na memória do computador. Mas só pode conter valores correspondentes a caracteres da tabela ascii.

Uma variável do tipo “*Short*” ocupa 16 bits na memória do computador. Por isso, estas variáveis podem conter números negativos e positivos de -32768 a 32767 ou, números positivos de 0 a 65535 “*Unsigned Short - UShort*”.

Uma variável do tipo “*integer*” ocupa 32 bits na memória do computador. Por isso pode conter números positivos de 0 a 2^{32} “*unsigned integer - UInteger*” ou, números negativos e positivos “*Integer*”.

As variáveis do tipo “vírgula flutuante”: *Single* e *Double* ocupam 4 e 8 bytes respectivamente na RAM.

As variáveis, mais não são que posições da memória RAM do computador, onde podemos escrever ou ler valores. Esses valores podem ser numéricos ou alfanuméricos. Se forem alfanuméricos é necessário codificá-los em números de acordo, por exemplo, com a norma American Standard Coded Information Interchange – ASCII.

Grandezas escalares

Para declarar uma variável em VBasic chamada *Contador* do tipo *Integer* deve fazer:

```
Dim Contador as Integer
```

Para atribuir um valor a esta variável, por exemplo o valor 100, pode fazê-lo logo no momento da sua declaração:

```
Dim Contador as Integer = 100
```

' Numeros inteiros com sinal

Dim s As Short = &H7FFF	' 2 bytes (-32 768 a +32 767)
Dim i16 As Int16 = &H7FFF	' 2 bytes (-32 768 a +32 767)
Dim i32 As Int32 = &H7FFFFFFF	' 4 bytes (-2 147 483 647 a + 2 147 483 647)
Dim i As Integer = &H7FFFFFFF	' 4 bytes (-2 147 483 647 a + 2 147 483 647)
Dim i64 As Int64 = &H7FFFFFFFFFFFFFFF	' 8 bytes
Dim il As Long = &H7FFFFFFFFFFFFFFF	' 8 bytes

' Numeros inteiros sem sinal

Dim b As Byte = &HFF	' 1 byte (0 a 255)
Dim us As UShort = &HFFFF	' 2 bytes (0 a 65 535)
Dim ui16 As UInt16 = &HFFFF	' 2 bytes (0 a 65 535)
Dim ui32 As UInt32 = &H7FFFFFFF	' 4 bytes (0 a 2 147 483 647)
Dim ui As UInteger = &H7FFFFFFF	' 4 bytes (0 a 2 147 483 647)
Dim ui64 As UInt64 = &H7FFFFFFFFFFFFFFF	' 8 bytes
Dim ul As ULong = &H7FFFFFFFFFFFFFFF	' 8 bytes

' Numeros reais

Dim de As Decimal = 3.123456789
Dim si As Single = 3.123456789
Dim d As Double = 3.123456789

```
'Texto
```

```
Dim st As String = "Ola"
```

Arrays e Strings

Uma matriz não é mais que um conjunto de posições da RAM do computador. As matrizes podem ser unidimensionais "vectores" ou multidimensionais.

Uma "string" é um vector que ocupa várias posições de memória, onde cada caracter é guardado numa posição de memória do vector e o último elemento do vector, o último byte, tem todos os bits a zero. Cada caracter da "string" é codificado em binário, de acordo com a tabela ASCII, e só depois é guardado na memória do computador.

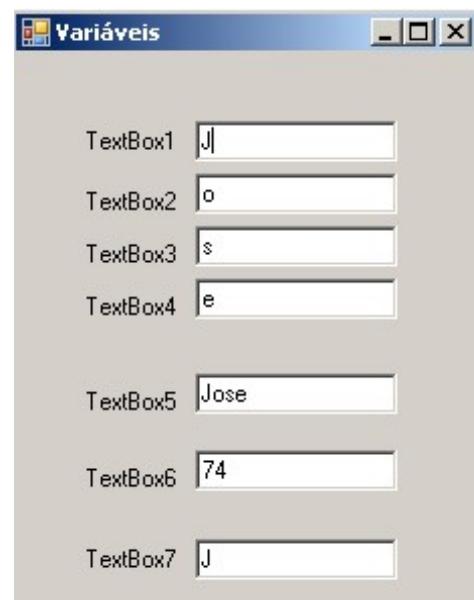
```
Dim letra(9) as char          ' Vector com 10 elementos: letra(0) a letra(9)
Letra(0) = "J"                ' J      = 74dec = &H4A      = 0b 01001010
Letra(1) = "o"                ' o      = 111dec = &H6F      = 0b 01101111
Letra(2) = "s"                ' s      = 115dec = &H73      = 0b 01110011
Letra(3) = "e"                ' e      = 101dec = &H65      = 0b 01100101
Letra(4) = chr(0)             ' valor nulo = 0 = &H00      = 0b 00000000
Textbox1.text = Letra          ' na textbox aparece a palavra Jose
```

```
Dim nome(10)(50) as char      ' Matriz de duas dimensões
```

' Variáveis globais

```
Dim cc(,) as char = {{"J","o"}, {"s","e"}}
Dim nome() as char = {"j","o","s",chr(101)}
Dim aluno() as char = "Jose"
Dim I as short = &H4A
```

```
Private sub Form1_Load( .....
    TextBox1.Text = cc(0,0)  ' J
    TextBox2.Text = cc(0,1)  ' o
    TextBox3.Text = cc(1,0)  ' s
    TextBox4.Text = cc(1,1)  ' e
    TextBox5.Text = nome    ' Jose
    TextBox6.Text = I       ' 74
    TextBox7.Text = chr(I)  ' J
end sub
```



3.5.3. Declaração de Funções e Procedimentos

As funções e os procedimentos permitem agrupar linhas de código numa única entidade que pode ser chamada a partir de outros locais do programa, evitando repetir as mesmas linhas de código ao longo do programa.

O exemplo seguinte descreve como se pode declarar e usar um procedimento. Neste exemplo o procedimento tem o nome *Mensagem()*.

Apesar do procedimento *Mensagem()* ser o primeiro a ser declarado neste exemplo, ele é chamado depois e a partir do sub procedimento *Form_Load(...)*. Neste exemplo o procedimento não recebe nem retorna valores.

```
Sub Mensagem()  
    TextBox1.Text = "Mensagem"          ' Texto  
End Sub  
  
Sub Form_load(...)  
    Mensagem()  
End Sub
```

Existe uma diferença entre *funções* e *procedimentos*. Os dois permitem “agrupar” código, tornando mais fácil a sua organização e reutilização ao longo do programa, mas só as funções podem retornar valores.

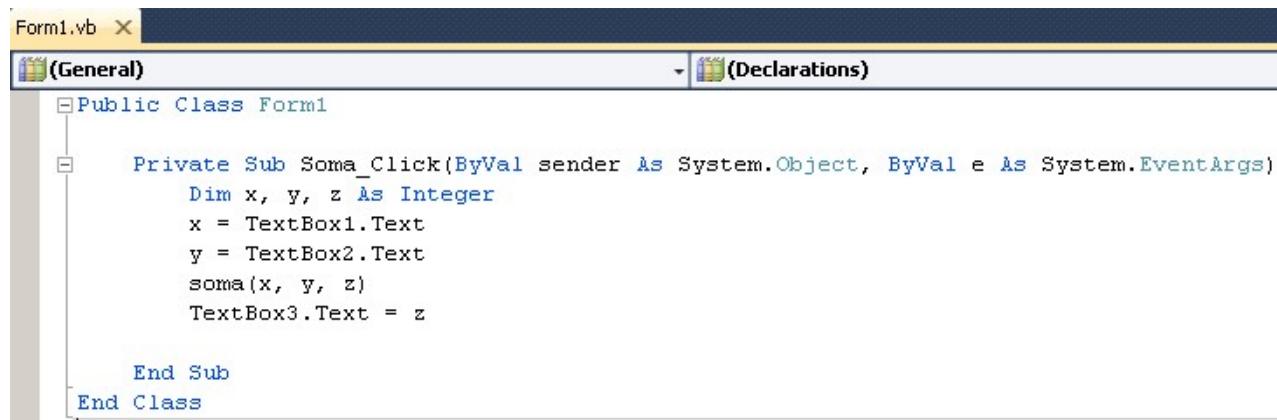
No exemplo seguinte é declarada uma função, intitulada *Pi()*. Esta função retorna o número 3,14 quando é chamada no sub procedimento *Form_Load(...)*.

```
Function Pi() Double  
    Return 3.14  
End Function  
  
Private sub Form_load(...)  
    TextBox1.Text = Pi()                ' Na caixa de texto aparece o número 3.14  
End Sub
```

Funções - passagem de parâmetros por valor ou por referência

Tanto as funções como os procedimentos podem receber parâmetros.

Neste exemplo, a função **soma(X,Y,Z)** recebe os parâmetros **X** e **Y** por valor. Ou seja, o valor/conteúdo de **X** e **Y** são passados para os parâmetros internos “**a**” e “**b**” da função **soma()**. O parâmetro **a** recebe o valor de **X** e o parâmetro **b** recebe o valor de **Y**. Mas as variáveis **X**, **Y** e os parâmetros **a** e **b** correspondem a zonas de memória diferentes.



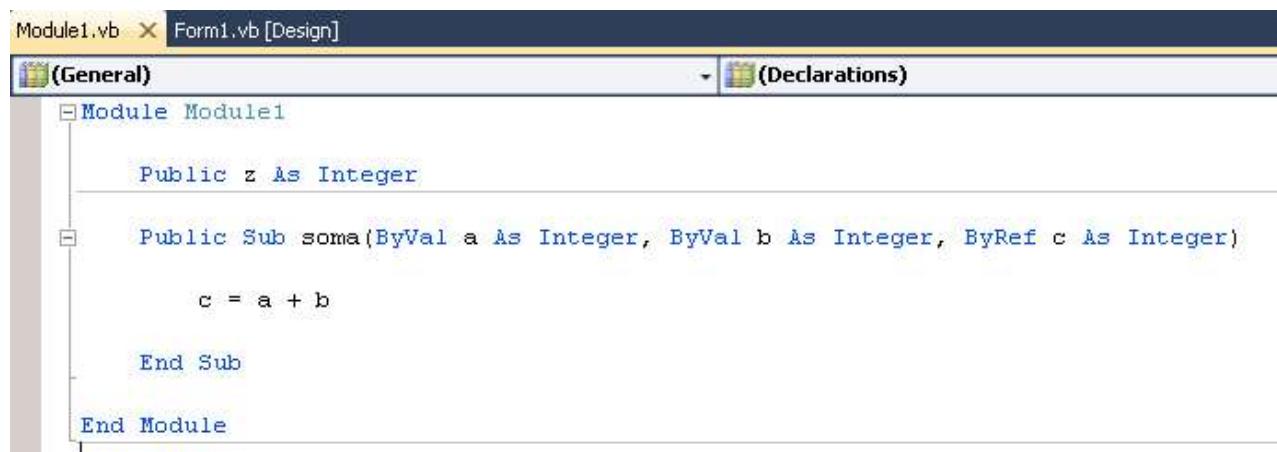
```
Form1.vb X Form1.vb [Design]
(General) (Declarations)

Public Class Form1

    Private Sub Soma_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim x, y, z As Integer
        x = TextBox1.Text
        y = TextBox2.Text
        soma(x, y, z)
        TextBox3.Text = z
    End Sub
End Class
```

Pode observar-se na figura seguinte que a função **soma** recebe por valor **ByVal** dois parâmetros do tipo inteiro **a** e **b**. Depois de efectuada a soma esse valor é atribuído ao parâmetro **c** que é passado por referência **ByRef**.

Como o parâmetro **c** é passado por referência, tanto o parâmetro **c** como a variável **Z** correspondem à mesma zona de memória. Ou seja, atribuir valores a **c** dentro da função **soma** é o mesmo que atribuir esses valores à variável **Z** fora da função **soma()**.



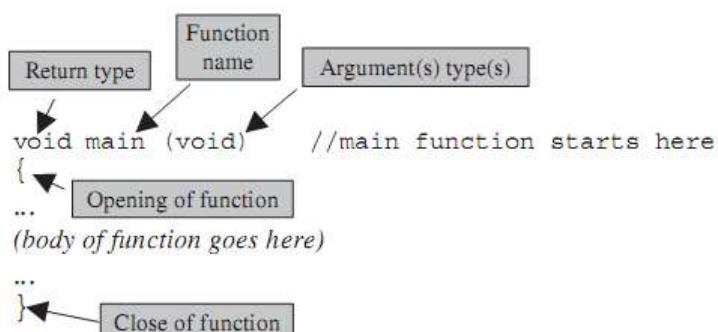
```
Module1.vb X Form1.vb [Design]
(General) (Declarations)

Module Module1

    Public z As Integer

    Public Sub soma(ByVal a As Integer, ByVal b As Integer, ByRef c As Integer)
        c = a + b
    End Sub

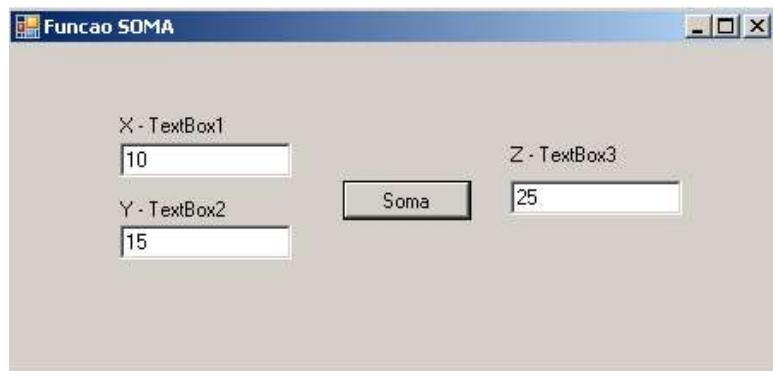
End Module
```



Funções externas e Variáveis externas

Se o seu programa, em “VBBasic”, estiver organizado em vários ficheiros de texto, pode declarar as variáveis ou as funções num deles e utilizá-las noutras ficheiros, mas tem de declarar as variáveis como sendo do tipo “*public*” e devem ser declaradas fora de qualquer função.

No exemplo seguinte pretende-se somar os dois valores. O utilizador introduz dois valores nas caixas de texto da esquerda (TextBox1 e TextBox2), e o resultado da soma é apresentado na caixa de texto da direita (TextBox3).



Foram declaradas três variáveis inteiros **X,Y,Z** e uma função **soma()** no ficheiro module1.vb que podem ser usadas noutras ficheiros, nomeadamente no ficheiro form1.vb

Quando o utilizador do programa selecionar o botão “Soma”, o sub procedimento **Soma_Click** é executado. Nessa altura, de acordo com o código escrito neste exemplo, o conteúdo das caixas de texto da esquerda (TextBox1 e TextBox2) é atribuído às variáveis globais **X** e **Y** e é chamada a função global **soma()** do module1.vb. Na função **soma()**, é atribuída à variável global **Z** o resultado da soma. Quando a execução do programa retorna ao sub procedimento **Soma_Click(...)** é atribuído à caixa de texto da direita (TextBox3) o valor da variável global **Z**.

The screenshot shows the Visual Studio IDE interface with three main windows:

- Module1.vb:** Contains the declaration of a module named **Module1** with a public variable **x, y, z As Integer** and a public sub procedure **soma()**. Inside **soma()**, the code is **z = x + y**.
- Form1.vb:** Contains the declaration of a class named **Form1** with a private sub procedure **Soma_Click(ByVal sender As Object, ByVal e As EventArgs)**. Inside **Soma_Click()**, the code is:

```
x = TextBox1.Text  
y = TextBox2.Text  
soma()  
TextBox3.Text = z
```
- Solution Explorer:** Shows the project structure for "WindowsApplication1" with files "My Project", "Form1.vb", and "Module1.vb".

3.5.4. Tipos de operadores

Operadores elementares (= + - * /)

Em “VBBasic” os operadores de atribuição “=”, soma “+”, subtração “-”, multiplicação “x” e divisão “/” têm a sintaxe que o exemplo seguinte apresenta:

```
Private Sub main()
    Dim X,Y,Z as Char
    X= 3
    Y= 2
    Z= X+Y           // Depois desta instrução a variável Z contém o valor 5
    Z= Z * 2 + 1     // Depois desta instrução a variável Z contém o valor 11
    Z= Z / 2          // Depois desta instrução a variável Z contém o valor 5
    TextBox1.Text =Z  // Depois desta instrução, a caixa de texto apresenta o valor 5
End Sub
```

Operadores relacionais

Os operadores booleanos comparam dois valores e com base na comparação realizada retornam um valor booleano “Verdadeiro” ou “Falso”.

Maior que	>
Maior ou igual	\geq
Menor que	<
Menor ou igual	\leq
Igual	=
Diferente	\neq

```
Public sub main()
    Dim A,B,C as Integer
    A=10
    B=15
    C=20

    If A < B Then   TextBox1.Text = "O resultado da comparação é TRUE, 10 é menor que 15"
    If B > A Then   TextBox1.Text = "O resultado da comparação é TRUE, 15 é maior que 10"
    If A <> B Then  TextBox1.Text = "O resultado da comparação é TRUE, 10 é diferente de 10"
    If A = A Then    TextBox1.Text = "O resultado da comparação é TRUE, 10 é igual a 10"
EndSub
```

Operadores lógicos

Intercepção de conjuntos	And
Reunião de conjuntos	Or
Negação	Not
Ou exclusivo	Xor

A	B	A And B	A Or B	Not A	A Xor B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

```
Private Sub main()
```

```
    Dim A,B as Boolean
```

```
    A=True
```

```
    B=False
```

```
    If B Or A Then
```

```
        TextBox1.Text = "True ou False é True"
```

```
    If A Xor B Then
```

```
        TextBox1.Text = "True xor False é True"
```

```
    If Not B Then
```

```
        TextBox1.Text = "Negação de False é True "
```

```
    If Not (A And B) Then
```

```
        TextBox1.Text = "A negação de (True and False) = True"
```

```
EndSub
```

3.5.5. Estruturas de controlo

A execução de um programa é sequencial, cada instrução é gravada na memória do computador em posições consecutivas e é executada pela ordem com que foi gravada, a menos que o programador tenha inserido no próprio código instruções que permitam alterar essa sequência de execução.

O ciclo While ... End While

A instrução WHILE(*condição*) ... EndWhile permite repetir a execução de algumas linhas código, mais exatamente das linhas de código que tenham sido escritas entre a palavra While e a palavra EndWhile. Estas linhas de código são repetidas enquanto a *condição* booleana do WHILE for verdadeira.

A *condição* do ciclo WHILE pode conter vários operandos e operadores mas o resultado final das operações tem de ser um valor booleano, verdadeiro ou falso. Enquanto a condição for verdadeira o ciclo repete-se.

While (*condição*)

instruções

instruções

EndWhile

```
i = 0  
while (i < 3)  
i=i+1;  
EndWhile  
TextBox1.Text = i
```

Depois de 3 ciclos, TextBox1.Text = 3

O ciclo For ... Next

Este ciclo permite repetir a execução das linhas de código que estiverem escritas entre o **For** e o **Next** do próprio ciclo. Este ciclo é semelhante ao ciclo WHILE, contudo a “condição” do ciclo WHILE pode assumir um valor falso em função da execução das instruções do próprio ciclo, fazendo com que o ciclo possa terminar mais cedo ou tarde, ao passo que o ciclo FOR tem um número de repetições predefinido.

for i=0 **to** n **step** k

instruções

instruções

Next

for i=0 to 10 step 1

...

Next

TextBox1.Text= i

‘Depois de 10 ciclos, TextBox1.Text = 11

Exemplo 1

Neste exemplo todos os 16 elementos da matriz Galo(3,3) ficam com os valores indicados na tabela. Os dois ciclos **For** do exemplo repetem-se 4 vezes. Contudo, como o segundo ciclo For foi declarado dentro do primeiro, por cada ciclo do primeiro (**For i**) o segundo ciclo (**For j**) repete-se 4 vezes. Ou seja, a instrução $\text{Galo}(i,j) = i * j$ repete-se 16 vezes.

```
Public Galo(3,3) as integer  
Public i, j as integer
```

```
For i= 0 to 3 Step 1  
    For j= 0 to 3 Step 1  
        Galo( i , j ) = i * j  
        TextBox1.Text = Csrt(i) + Cstr(j)  
    Next
```

```
Next
```

Na Matriz Galo(i,j) o índice “j” corresponde às colunas e o índice “i” às linhas.

$\text{Galo}(i,j)$	$j=0$	1	2	3
$i=0$	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9

If... End If

O programador pode usar a instrução IF quando se pretende executar algumas linhas de código se, e apenas se “**IF**”, uma determinada condição for verdadeira, ou pelo contrário “**Else**”, executar outras linhas de código se a condição for falsa.

```
if ( condição )
```

```
'se condição verdadeira  
instruções;  
else  
'se condição falsa  
instruções;  
End If
```

```
i= 5;
```

```
if ( i < 10 )
```

```
'Se verdadeira  
TextBox1.Text = 3  
else  
'Se falsa  
TextBox1.Text = 2  
End If
```

Como i = 5, a condição (i<10) é verdadeira

TextBox1.Text = 3

Se i fosse maior que 10 a condição (i<10) era falsa

TextBox1.Text = 2

Em suma, esta instrução permite que apenas um de dois conjuntos de instruções seja executado, em função de uma *condição* ser verdadeira ou falsa.

Select Case End Select

A instrução **Select case** permite que apenas um de muitos conjuntos de instruções seja executado, Consoante o valor numérico ou alfanumérico de uma variável será executado apenas o caso correspondente (**case**).

```
Dim i as Integer
```

```
i = 1
```

```
Select Case i
```

```
Case 1: instruções
```

```
Case 2: instruções
```

```
Case Else: instruções
```

```
End Select
```

```
Dim s as String
```

```
s = "2"
```

```
Select Case ( s )
```

```
Case "1": TextBox1.Text = "Um"
```

```
Case "2": TextBox1.Text = "Dois"
```

```
Case Else: TextBox1.Text = "Outro"
```

```
End Select
```

Como s = “2,”

TextBox1.Text = “Dois”

Se s = “1”

TextBox1.Text = “Um”

3.5.6. Biblioteca de funções para manipulação de texto

A função “Mid”

Permite extrair uma *substring* a partir de uma *string* original.

Esta função recebe três parâmetros: uma variável do tipo *string*, contendo um conjunto de caracteres; um número com a posição do primeiro caracter a ler; e o número de caracteres consecutivos a ler.

Mid-Exemplo:

```
Dim strOriginal As String = "0123456789"  
MessageBox.Show(Mid(strOriginal, 1, 2))  'devolve o valor "01"  
MessageBox.Show(Mid(strOriginal, 5, 3))  'devolve o valor "456"
```

A função “Split”

Divide uma *string* em *substrings*.

Esta função recebe uma *string*, um carácter delimitador e devolve um array de *strings*. Em cada posição do array devolvido vem uma parte da *string* original. Por exemplo, se a *string* “String_original” possuir um espaço em branco entre cada letra e o utilizador quiser separar as letras, obtendo em cada posição do array uma só letra, pode indicar à função “split” para usar o carácter “espaço“ como divisor da *string* original. A este carácter, usado para dividir a “String_original”, chama-se delimitador.

Split-Exemplo:

```
Dim strOriginal As String = "o l a"  
Dim arrayDeStrings() As String  
arrayDeStrings = Split(strOriginal, " ")  
MessageBox.Show(arrayDeStrings(0))  
MessageBox.Show(arrayDeStrings(1))  
MessageBox.Show(arrayDeStrings(2))
```

A função “InStr”

Permite localizar a posição de uma *substring* numa *string* maior.

Esta função recebe duas *strings*: a *string* maior e a *substring* e devolve um inteiro com posição da *substring* na *string* maior. O valor devolvido pode ser (0) se a *substring* não for encontrada na *string* maior, ou (ε 1)

InStr-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
Dim strChave As String = "cd"  
MessageBox.Show(InStr(strOriginal, strChave)) 'devolve o valor 3
```

As funções “Right” e “Left”

Devolve uma *substring* que contém um determinado número de caracteres, a contar da direita ou da esquerda, da *string* original.

Por existir uma propriedade do form com o mesmo nome é necessário utilizar o caminho completo *Microsoft.VisualBasic.Left* e *Microsoft.VisualBasic.Right*.

Right e Left-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
MessageBox.Show(Microsoft.VisualBasic.Left(strOriginal, 3))  'devolve o valor "abc"  
MessageBox.Show(Microsoft.VisualBasic.Right(strOriginal, 2))  'devolve o valor "hi"
```

A função “Len”

Devolve um inteiro com o número de caracteres que existem numa *string*.

Esta função recebe uma *string*, da qual queremos saber o seu comprimento, ou seja o número de caracteres que a constituem e devolve um número.

Len-Exemplo:

```
Dim strOriginal As String = "abcdefghijklm"  
MessageBox.Show(Len(strOriginal)) 'devolve o valor 9
```

A função “StrComp”

Compara duas *strings*.

Esta função recebe duas *strings* e devolve um valor inteiro, igual a zero se as duas *strings* forem iguais.

StrComp-Exemplo:

```
Dim palavra1 As String = "123"  
Dim palavra2 As String = "123"  
MessageBox.Show(StrComp(palavra1, palavra2)) 'devolve o valor 0, ou -1 se for diferente
```

A função “CStr”

Permite converter um número numa *string*. A *string* obtida pode então ser atribuída a variáveis do tipo *string* ou usada em funções de manipulação de *strings*.

CStr-Exemplo:

```
Dim texto As String  
texto = CStr(123)
```

A função “Chr”

Converte um valor numérico no seu carácter correspondente, de acordo com a tabela ASCII

Chr-Exemplo:

```
MessageBox.Show(Chr(65)) 'devolve o valor "A"
```

A função “Asc”

Converte um carácter no seu valor decimal de acordo com a tabela ASCII.

Esta função recebe uma string e devolve um valor inteiro.

Asc-Exemplo:

```
MsgBox(Asc("A")) 'devolve o valor 65
```

A função “Trim”

Rerira os espaços em branco no inicio e no fim de uma string, mas deixa ficar os espaços em branco no meio da string.

Esta função recebe uma string e devolve uma string sem espaços em branco no inicio e no fim.

Trim-Exemplo:

```
Dim strOriginal As String = " ola bom dia "
MsgBox(Trim(strOriginal)) 'devolve a string "ola bom dia"
```

Veja também as funções:

InStrRev; UCASE(Text1); LCASE(Text1); LTrim, RTrim.

3.5.7. Funções matemáticas

$^$	- Expoente
SqrT	- Raiz quadrada
Mod	- Resto da divisão inteira (Res = I Mod 10)
\	- Resultado inteiro da divisão
Abs(n)	- Valor absoluto de n;
Atan(n)	- Arco Tangente de n, n em radianos;
Cos(n)	- Coseno de n, n em radianos;
Sin(n)	- Seno de n, n em radianos;
Tan(n)	- Tangente de n, n em radianos;
Exp(n)	- Expoente de n;
Sign(n)	- Devolve o sinal de um numero, -1 se <0, 1 se > 0;

3.6. Projeto com vários Forms e Módulos

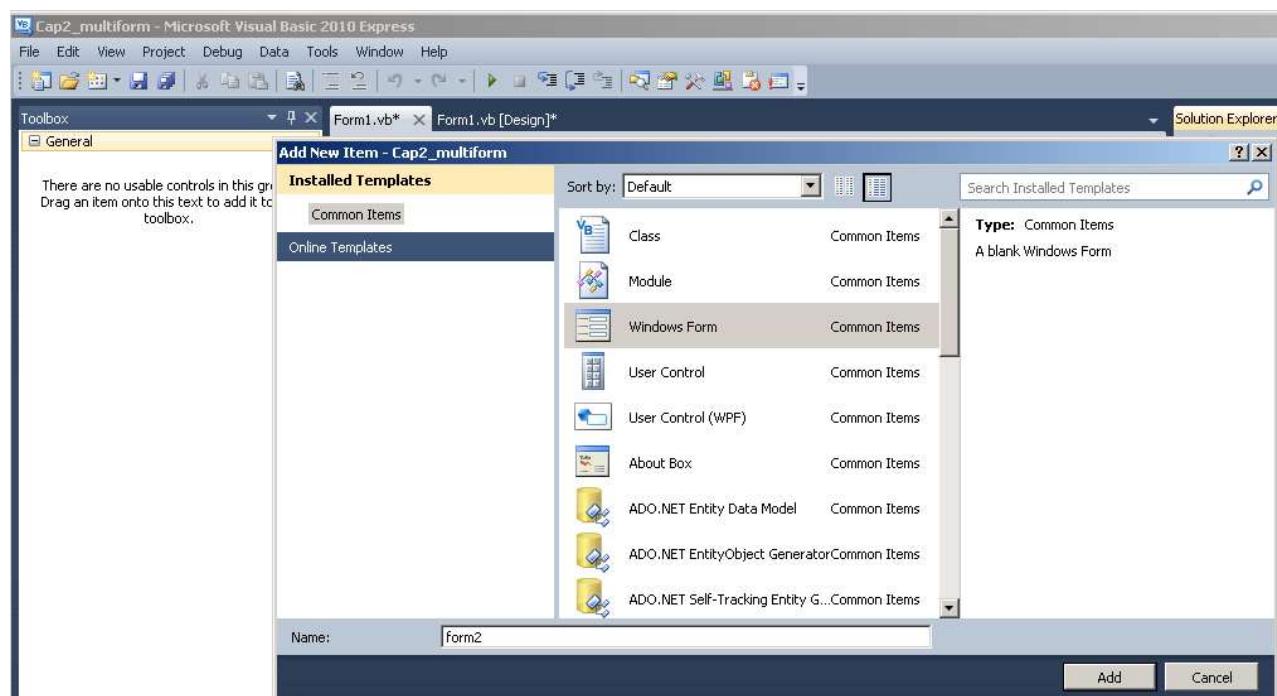
Nesta secção vamos abordar projetos em VBasic com várias interfaces (forms), módulos, variáveis e funções globais.

Como até agora apenas usámos um Form nos projetos, sempre que o programa era executado era esse “form” que aparecia no ecrã.

3.6.1. Criar um segundo Form

Vamos criar um segundo “Form” com o nome “Form2” onde o utilizador poderá introduzir texto numa TextBox. Pretende –se que ao voltar ao “Form” inicial apareça numa TextBox o nome que o utilizador introduziu na TextBox do segundo Form.

Para criar um segundo “Form” selecione a opção “**Add Windows form**” no menu “Project”. Crie um Form com o nome **Form2**.



Nessa altura poderá observar na janela “**Solution Explorer**” que o seu projeto passou a ter dois forms: o *Form1* e o *Form2*.

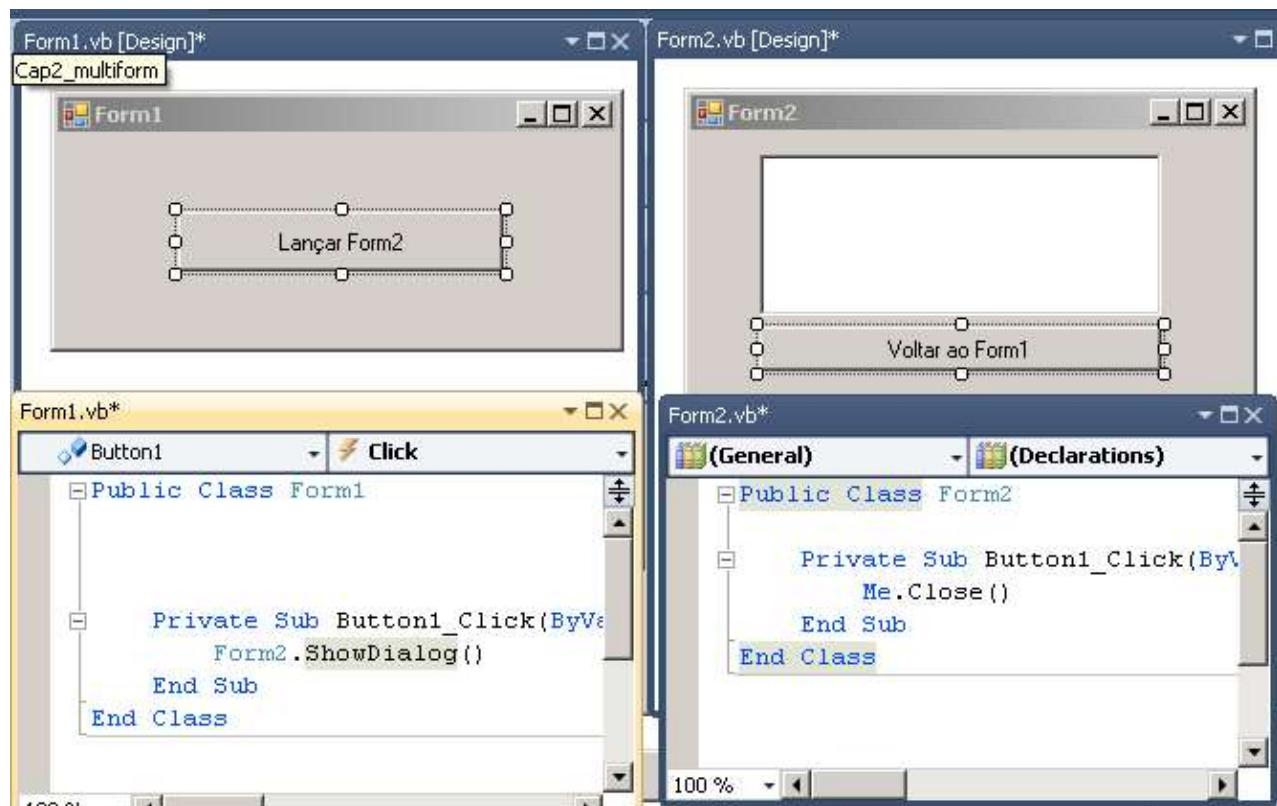
Cada um deles possui uma janela de interface (Form1.vb design e Form2.vb design) e um ficheiro de código (Form1.vb e Form2.vb).



3.6.2. Alternar entre Forms

Se no *Form1* usar a função **Form2.ShowDialog()** ativa e visualiza o *Form2*. Para fechar o *Form2*, num dos subprocedimentos do *Form2* deve fazer **me.Close()**.

O exemplo seguinte possui dois *Forms*, cada um com um botão, que quando for pressionado muda para o outro *Form*, o código é auto explicativo.



3.6.3. Passar dados entre Forms

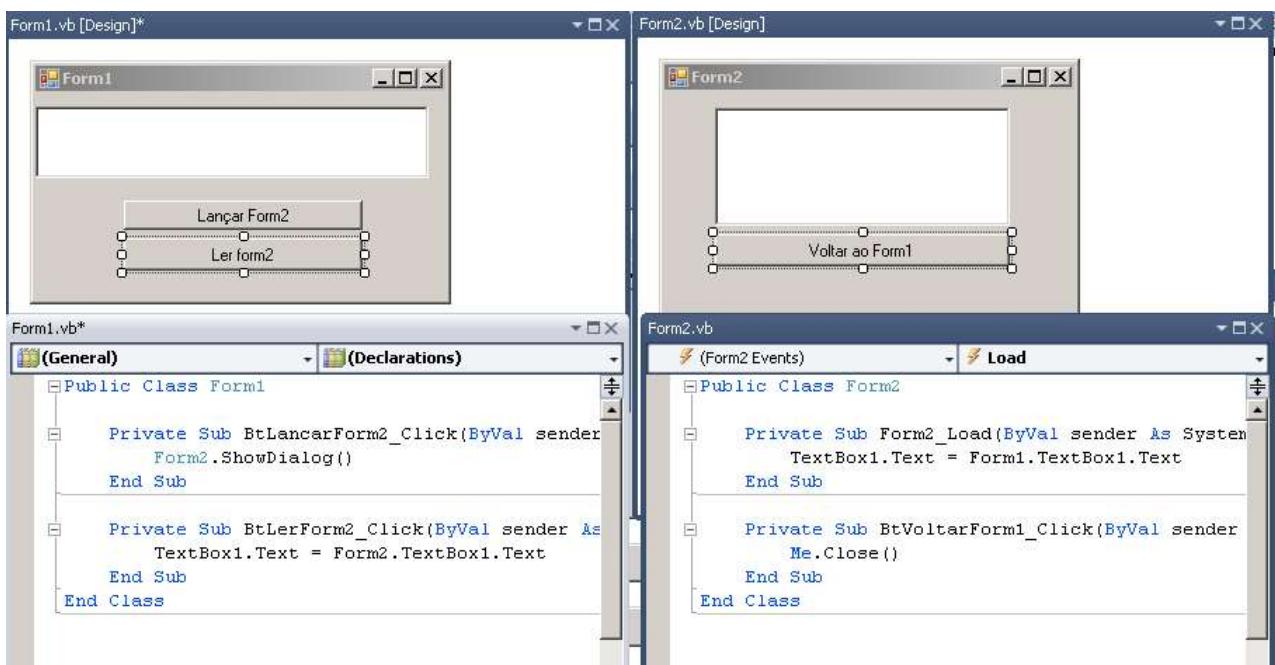
Em cada *Form* o utilizador pode introduzir dados, mas pode ser necessário que os dados introduzidos num *Form* tenham de ser tratados noutro *Form*. Poem-se por isso a necessidade de passar dados entre *Form*.

Neste exemplo, o utilizador introduziu texto na “Textbox1” do *Form2* e quer ver esse texto no objeto “TextBox1” do *Form1*. De facto, existem dois objetos “TextBox1”, um pertence ao *Form1* e o outro ao *Form2*.

O *Form1* tem dois botões, um permite ativar o *Form2* e o outro botão permite visualizar na Textbox1 do *Form1* o conteúdo do segundo “**Form2.TextBox1**”.

Para identificarmos objetos ou variáveis de um *Form*, a partir de outro *Form*, devemos referir o nome do *Form* onde essas variáveis ou objetos foram declarados, por exemplo:

TextBox1.text = Form2.TextBox1.text

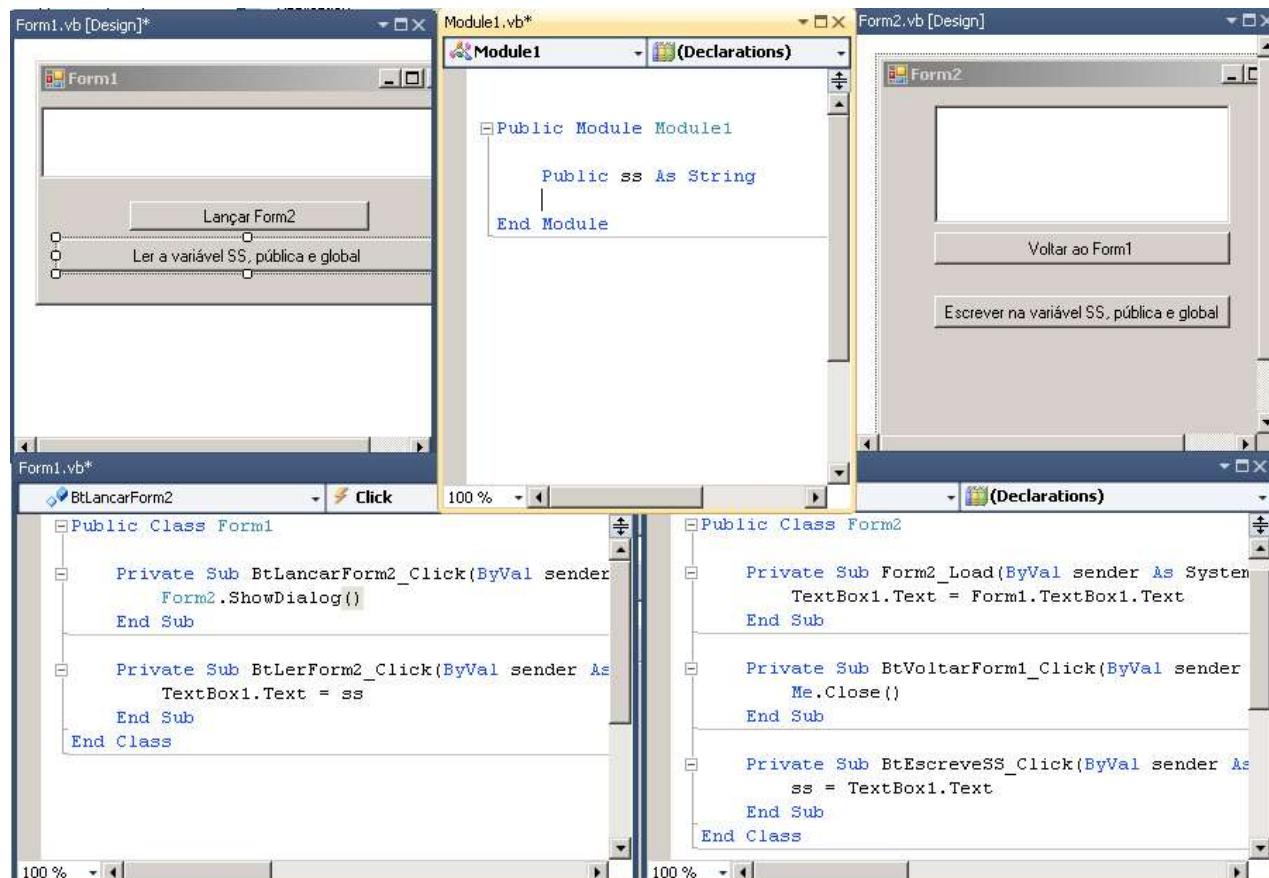


3.6.4. Módulos com código e dados

Pode também adicionar ao seu projeto um “**module**”. Ao contrário dos Forms, os módulos apenas permitem definir código e dados (variáveis), mas não têm uma interface gráfica associada.

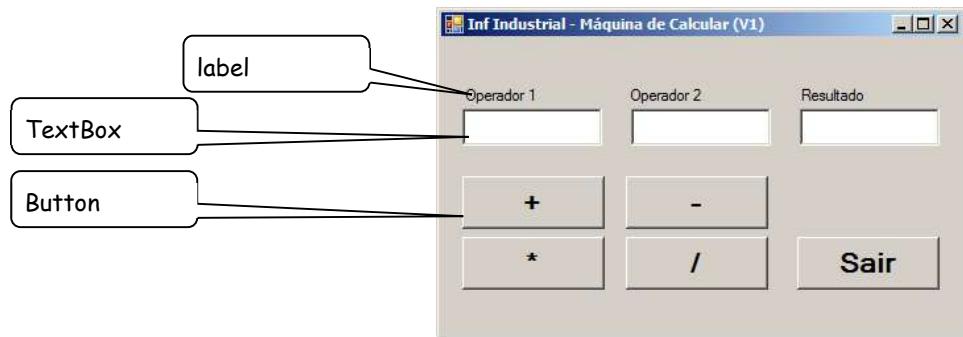
O exemplo seguinte apresenta dois forms, o Form2 tem um botão que ao ser premido grava na variável global “SS” o texto que estiver na Form2.TextBox1.

O Form1 tem outro botão que lê a variável global e visualiza o texto na Form1.TextBox1.



3.6.5. Exemplo 1 : Máquina de calcular com 4 operações matemáticas fundamentais

Máquina de calcular com as 4 operações matemáticas principais: adição, subtração, divisão e multiplicação.



Inserir os objetos indicados na figura:

- Label, TextBox e Button, como indicado na figura.

Para cada um dos objetos, pode-se alterar as propriedades: Nome, texto, tipo de letra, etc. e associar a cada objeto um conjunto de instruções a executar, quando o respetivo objeto é selecionado.

Para cada **button** associar uma das operações matemáticas e, escrever o código indicado na listagem, clicando 2 vezes em cada um dos objetos. A cada objeto está associado uma rotina: **Private Sub**, onde o cabeçalho é inserido automaticamente:

```
Private Sub Btn_Sair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btn_Terminar.Click
    'Terminar a aplicação
    End
End Sub
```

Instruções:

- End - termina a execução do programa;
- Cint - converte um texto numa variável numérica inteira;
- CDbl - converte uma expressão em double;

Notas:

- É importante a organização dos programas;
- Os programas devem estar devidamente identificados:
 - Data, Identificação do aluno, Número da aula prática;
- Os programas devem ser comentados;
- Os objetos e as rotinas devem ser identificados utilizando critérios bem definidos. Os objetos podem ser identificados completando o seu tipo com um prefixo adequado, por exemplo:
 - TextBox - TextBoxLeitura
 - Button - ButtonSair
 - Label - LabelFicheiro

Listagem do programa:

```
Public Class Form1

    Private Sub Btn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt.Click
        End      ' Terminar programa
    End Sub

    Private Sub Btn_Soma_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Soma.Click
        ' Adição
        Txt_Res.Text = CInt(Txt_Op1.Text) + CInt(Txt_Op2.Text)
    End Sub

    Private Sub Btn_Sub_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Sub.Click
        ' Subtração
        Txt_Res.Text = CInt(Txt_Op1.Text) - CInt(Txt_Op2.Text)
    End Sub

    Private Sub Btn_Mul_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Mul.Click
        ' Multiplicação
        Txt_Res.Text = CInt(Txt_Op1.Text) * CInt(Txt_Op2.Text)
    End Sub

    Private Sub Btn_Div_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Div.Click
        ' Divisão
        Txt_Res.Text = CInt(Txt_Op1.Text) / CInt(Txt_Op2.Text)
    End Sub

End Class
```

3.6.6. Exemplo 2 : Máquina de calcular com registo das operações realizadas em ficheiro

Realizar a leitura e escrita em ficheiro de texto, dos resultados das operações matemáticas realizadas na máquina de calcular. Na janela lateral de leitura poderá visualizar o conteúdo desse ficheiro.

Escrita em ficheiro de texto, é necessário definir o nome e local de armazenamento do ficheiro de texto e **abrir o ficheiro** através da instrução:

```
'Declaração e abertura do ficheiro de escrita em disco  
FileOpen(1, "E:\resultados.Txt", OpenMode.Output)
```

Esta instrução deve ser executada sempre que se quer escrever no ficheiro.

Escrita de dados no ficheiro:

```
Print(1, " + " + Txt_Op1.Text)
```

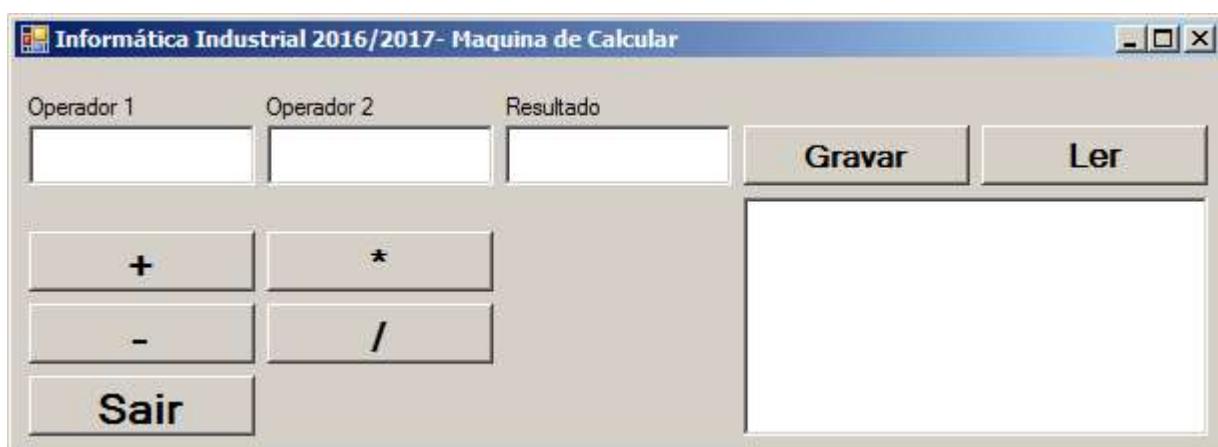
Para fechar e “gravar” em disco, executar a instrução:

```
' Fecha ficheiro de escrita em disco  
FileClose(1)
```

Leitura do ficheiro:

```
' Cada vez que esta função é executada devolve uma linha de texto existente no ficheiro  
Txt_Ficheiro.Text = Txt_Ficheiro.Text + LineInput(1) + vbCrLf
```

Ao exemplo anterior acrescentar dois botões com os nomes “Btn_Gravar” e “Btn_Ler”, acrescentar uma janela de texto com o nome “Txt_Ficheiro”.



```
Private Sub Btn_Gravar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Gravar.Click  
    FileOpen(1, "E:\resultados.txt", OpenMode.Append)  
    PrintLine(1, "Resultado= " + Txt_Res.Text)  
    FileClose()  
End Sub  
  
Private Sub Btn_Ler_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Bt_Ler.Click  
    FileOpen(1, "E:\resultados.txt", OpenMode.Input)  
  
    While (EOF(1) = False)  
        Txt_Ficheiro.Text = Txt_Ficheiro.Text + LineInput(1) + vbCrLf  
    End While  
  
    FileClose()  
End Sub
```

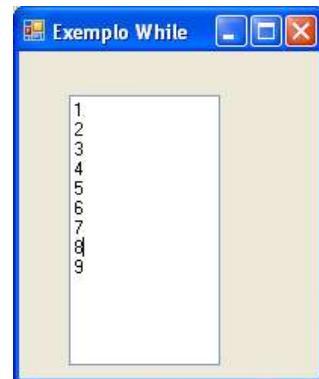
A função **EOF(1)** (*End Of File*) indica o fim do ficheiro, ou seja, já foram lidas todas as linhas do ficheiro (1), devolvendo o valor “True”

O ciclo **While ... End While** repete automaticamente as instruções existentes no seu interior, ou seja enquanto a função **EOF(1)** devolver o valor falso o ciclo é repetido, realizando a leitura da próxima linha do ficheiro.

Outro exemplo para o **ciclo While**, pode ser realizado para incrementar uma variável inteira, com o nome “i”. Durante a execução do programa, a variável “i” assume inicialmente o valor 1, sendo incrementada sequencialmente até 10. O valor 10 já não é visualizado na caixa de texto “Txt_Res”.

Dim i As Integer

```
i = 1
While (i < 10)
    Txt_Res.Text = Txt_Res.Text + CStr(i) + vbCrLf
    i = i + 1
End While
```



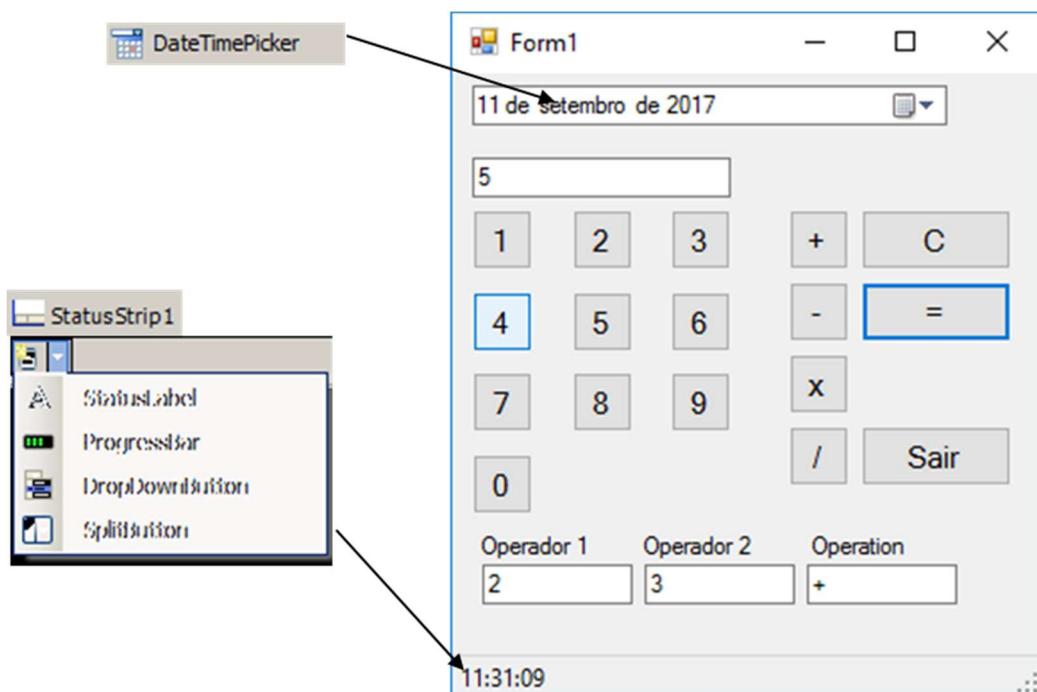
3.7. Trabalho nº 2 – “Máquina de Calcular”

José Paulo Santos, Hugo Ribeiro, Abílio Borges, Miguel Oliveira

3.7.1. Introdução

Pretende-se desenvolver uma máquina de calcular elementar, com a visualização da data e hora do computador que permita a escrita e a leitura de dados em ficheiros de texto, para familiarização e aquisição de conhecimentos de algumas instruções e objetos do Visual Basic.

Os objetos Timer e StatusStrip, não são visualizados na “form” do programa, mas na parte inferior do form1.vb(design) e são executados numa tarefa paralela à execução do programa.



O objeto StatusStrip1 -StatusLabel adicionado, é utilizado para visualizar a hora atual do computador.

O Objeto timer (com o nome TimerDate) é executado ciclicamente, sendo necessário ativá-lo, nas propriedades gerais do objeto ou através de instruções de código, como é feito mais adiante.

As instruções escritas no procedimento associado ao Form, são executadas no início do programa e utilizadas para atribuir valores iniciais às variáveis do programa (predefinições).

As instruções associadas aos temporizadores são executadas ciclicamente, quando ativo. O botão “C”, limpa o conteúdo do campo de entrada de dados.

3.7.2. Objectivos

O trabalho prático pretende familiarizar o aluno com o ambiente de programação do visual studio.

Familiarização com os objectos de programação:

- Textbox, Button, Label, Checkbox, Picture, Panel, RadioButton, Form, Modules, etc.

Familiarização com as instruções de programação:

- End - termina a execução do programa,
- Operadores matemáticos: +, -, *, /
- Cint – Converte texto num número inteiro,
- Escrita e leitura em ficheiros de texto,
- While ... End While : execução em ciclo,
- If Then Else – Instrução de condição,
- Escrita e Leitura em ficheiro de texto;

```
...box\AulasUA\II_2017-2018\Pratica\Aula2\Ex3\Ex3\Form1.vb  
1  Public Class Form1  
2  
3      'Global program variables  
4      Dim operator1 As String  
5      Dim operator2 As String  
6      Dim operation As String  
7  
8      '-----  
9      'Initialization routine2  
10     '-----  
11     Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load  
12         'Initialize the Timer  
13         TimerDate.Interval = 500  
14         TimerDate.Enabled = True  
15     End Sub  
16  
17     '-----  
18     'Mathematical operation Button callbacks  
19     '-----  
20     Private Sub ButtonAdd_Click(sender As Object, e As EventArgs) Handles ButtonAdd.Click  
21         SetOperationAndOperator1("+", TextBoxOpResult.Text)  
22     End Sub  
23  
24     Private Sub ButtonSubtract_Click(sender As Object, e As EventArgs) Handles ButtonSubtract.Click  
25         SetOperationAndOperator1("-", TextBoxOpResult.Text)  
26     End Sub  
27  
28     Private Sub ButtonMultiply_Click(sender As Object, e As EventArgs) Handles ButtonMultiply.Click  
29         SetOperationAndOperator1("*", TextBoxOpResult.Text)  
30     End Sub  
31  
32     Private Sub ButtonDivide_Click(sender As Object, e As EventArgs) Handles ButtonDivide.Click  
33         SetOperationAndOperator1("/", TextBoxOpResult.Text)  
34     End Sub  
35  
36     'Called when +, -, * or / are pressed  
37     Private Sub SetOperationAndOperator1(operation_arg As String,  
38         operator1_arg As String)  
39         'Set operation  
40         operation = operation_arg  
41  
42         'Set operator1  
43         operator1 = operator1_arg  
44  
45         'Delete from TextBoxOpResult  
46         TextBoxOpResult.Text = ""  
47  
48         'for debug, show operator1 and operation  
49         TextBoxOp1.Text = operator1  
50         TextBoxOperation.Text = operation
```

```
51    End Sub
52
53    Private Sub ButtonClearResult_Click(sender As Object, e As EventArgs) Handles ButtonClearResult.Click
54        TextBoxOpResult.Text = ""
55        TextBoxOp1.Text = ""
56    End Sub
57
58    Private Sub ButtonEqual_Click(sender As Object, e As EventArgs) Handles ButtonEqual.Click
59        operator2 = TextBoxOpResult.Text
60
61        'debug: show operator2 in textboxOp2
62        TextBoxOp2.Text = operator2
63
64        If IsNumeric(operator1) And IsNumeric(operator2) Then
65            If operation = "+" Then
66                TextBoxOpResult.Text = CDbl(operator1) + CDbl(operator2)
67            ElseIf operation = "-" Then
68                TextBoxOpResult.Text = CDbl(operator1) - CDbl(operator2)
69            ElseIf operation = "*" Then
70                TextBoxOpResult.Text = CDbl(operator1) * CDbl(operator2)
71            ElseIf operation = "/" Then
72                TextBoxOpResult.Text = CDbl(operator1) / CDbl(operator2)
73            End If
74
75        Else
76            TextBoxOpResult.Text = "Error"
77        End If
78    End Sub
79
80    '-----
81    'Numeric Button callbacks
82    '-----
83    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
84        TextBoxOpResult.Text = TextBoxOpResult.Text + "1"
85    End Sub
86
87    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
88        TextBoxOpResult.Text = TextBoxOpResult.Text + "2"
89    End Sub
90
91    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
92        TextBoxOpResult.Text = TextBoxOpResult.Text + "3"
93    End Sub
94
95    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
96        TextBoxOpResult.Text = TextBoxOpResult.Text + "4"
97    End Sub
98
99    Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
```

...box\AulasUA\II_2017-2018\Pratica\Aula2\Ex3\Ex3\Form1.vb

```
100     TextBoxOpResult.Text = TextBoxOpResult.Text + "5"
101 End Sub
102
103 Private Sub Button6_Click(sender As Object, e As EventArgs) Handles
104     Button6.Click
105     TextBoxOpResult.Text = TextBoxOpResult.Text + "6"
106 End Sub
107
108 Private Sub Button7_Click(sender As Object, e As EventArgs) Handles
109     Button7.Click
110     TextBoxOpResult.Text = TextBoxOpResult.Text + "7"
111 End Sub
112
113 Private Sub Button8_Click(sender As Object, e As EventArgs) Handles
114     Button8.Click
115     TextBoxOpResult.Text = TextBoxOpResult.Text + "8"
116 End Sub
117
118 Private Sub Button9_Click(sender As Object, e As EventArgs) Handles
119     Button9.Click
120     TextBoxOpResult.Text = TextBoxOpResult.Text + "9"
121 End Sub
122
123 '-----
124 'Other methods
125 '-----
126 Private Sub ButtonExit_Click(sender As Object, e As EventArgs) Handles
127     ButtonExit.Click
128 End Sub
129
130 'Periodic timer callback to update time on ToolStripStatusLabel
131 Private Sub TimerDate_Tick(sender As Object, e As EventArgs) Handles
132     TimerDate.Tick
133     ToolStripStatusLabel1.Text = Format(Now, "HH:mm:ss")
134 End Sub
135 End Class
```



CAPÍTULO

4

EIA232

JPSantos
2023

4. EIA232 - PROTOCOLO DE COMUNICAÇÃO SÉRIE

4.1. Protocolo de comunicação EIA232

Para os equipamentos poderem comunicar entre si têm de respeitar um conjunto de regras comuns. A esse conjunto de regras de comunicação chama-se protocolo de comunicação.

A *Recomendação para Standard 232 – RS232* tinha por objectivo permitir a ligação de equipamentos digitais a redes públicas analógicas, por exemplo, a ligação de terminais e computadores à rede telefónica, usando para o efeito modems. Convém lembrar que na época a única rede de comunicação mundial era a rede telefónica, analógica, concebida para transmitir a voz humana.

Este protocolo foi aprovado pela *Electronic Industries Association* - EIA, sob a designação EIA232C. A norma EIA-232 tem tido várias evoluções, desde a primeira versão elaborada em 1960, até à versão EIA-232F.

Os “*modulator/demodulator – modems*” convertem sinais digitais em sinais analógicos, o que é fundamental quando se pretende transmitir informação a grandes distâncias usando as linhas analógicas da rede telefónica. Os sinais analógicos, com frequências adequadas, sofrem menor atenuação e menor distorção ao longo de condutores de cobre, sendo por isso capazes de percorrer maiores distâncias que os sinais digitais.

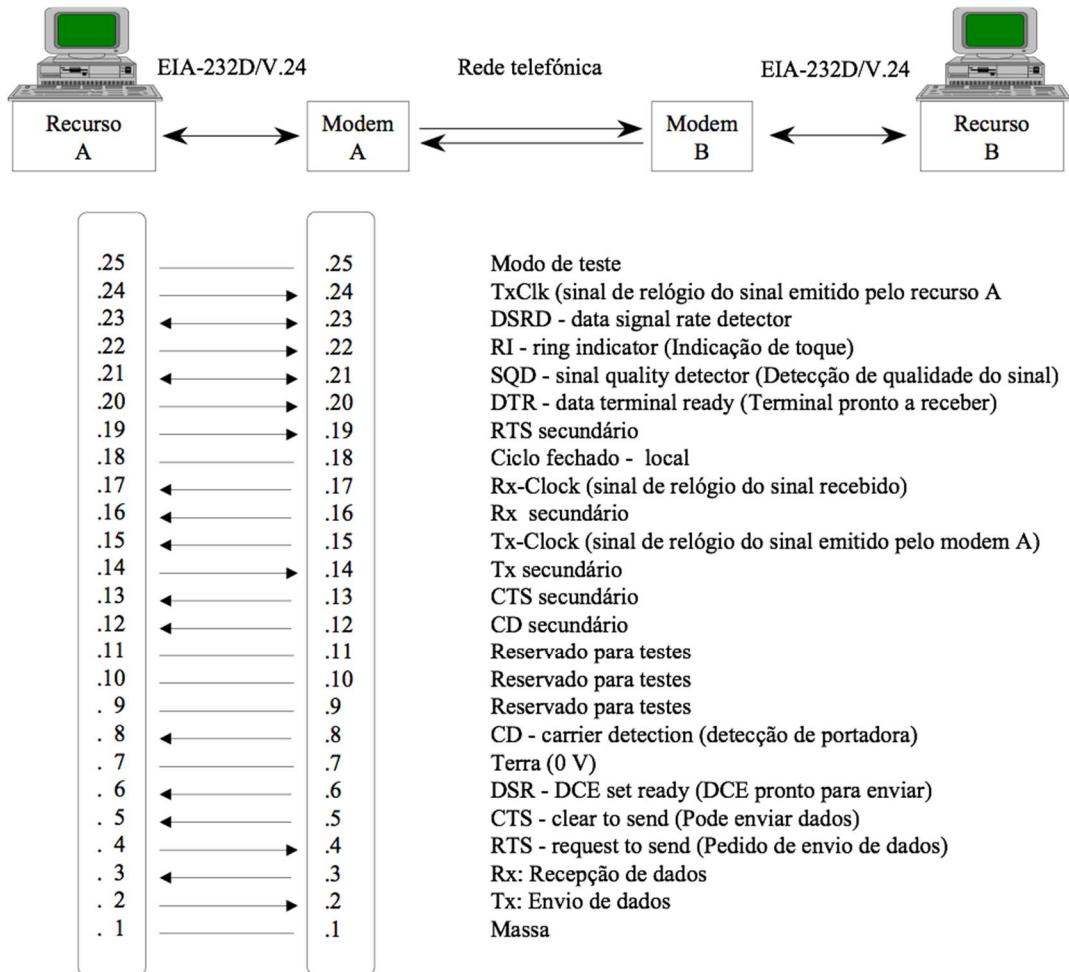


Figura 4.1: A gênese do protocolo Rs232

Note que a norma EIA-232D/V.24 apenas define a interface entre cada recurso e o seu modem respectivo, definindo por exemplo o significado e utilidade de cada pino das fichas DB25 (na Figura 4.1). Para interligar fisicamente recursos que se encontram próximos um do outro pode-se ligar as suas portas série diretamente, sem recorrer a modems.

Os pinos Tx e Rx são utilizados pelo recurso (A) para enviar e receber dados. Os outros pinos são utilizados para controlar: o início e fim da ligação telefónica, o fluxo de dados e ainda se for caso disso controlar o sincronismo do recurso (A) com o modem (A). Os pinos secundários da ficha DB25 permitem que uma segunda transferência de informação possa ocorrer em simultâneo usando a mesma ficha (a mesma interface).

A versão EIA-232D prevê dois tipos comunicação: síncrona e assíncrona. Numa comunicação assíncrona o sinal gerado pelo emissor, mais exatamente, o instante da transição do sinal (de 0 para 1 e de 1 para 0) depende apenas do seu relógio interno. Pelo contrário, numa comunicação síncrona tanto o emissor como o receptor dependem do mesmo sinal de relógio para determinar os instantes da transição do sinal de dados e dessa forma saberem em que instante de tempo amostrar o sinal. No modo de comunicação síncrona, além de ser necessário utilizar os pinos Tx e Rx para enviar e receber dados, é necessário também utilizar os pinos *TxClock* e *RxClock* para sincronizar o emissor e o receptor.

A revisão D normalizou a utilização das fichas DB25 e permitiu o aumento da tensão nos condutores que passou de ± 15 para ± 25 V. A distância máxima de 15 metros também foi alterada, atualmente a distância máxima pode ser superior, mas a capacidade parasita da linha não pode exceder os 2,5 nF. A revisão E aprovou a utilização de uma nova ficha de 26 pinos, a ficha ALT-A.

A comunicação série RS232 continua a ser um dos protocolos de comunicação de referência e muito utilizado industrialmente. Inúmeros equipamentos utilizam a comunicação série para troca de dados, por exemplo pistolas para leitura de códigos de barras, impressoras de etiquetas de códigos de barras e códigos 2D (Datamatrix) conectados a PLC's e computadores. As comunicações série referidas são utilizadas também para implementar sistemas de rastreabilidade, para comunicar com PLC's e Comandos Numéricos para programação e diagnóstico.



A comunicação RS232 é uma comunicação ponto a ponto, ou seja, é realizada apenas entre dois equipamentos. Para que ambos os equipamentos comuniquem entre si, é necessário que ambos tenham os mesmos parâmetros de comunicação:

- **Velocidade de transmissão** de dados, definido em bits por segundo (bps), os valores mais utilizados são: 9600 bps, 19200 bps, 38400 bps;

- **Controlo de paridade**, permite o controlo de erros na transmissão: ODD, EVEN e NONE.

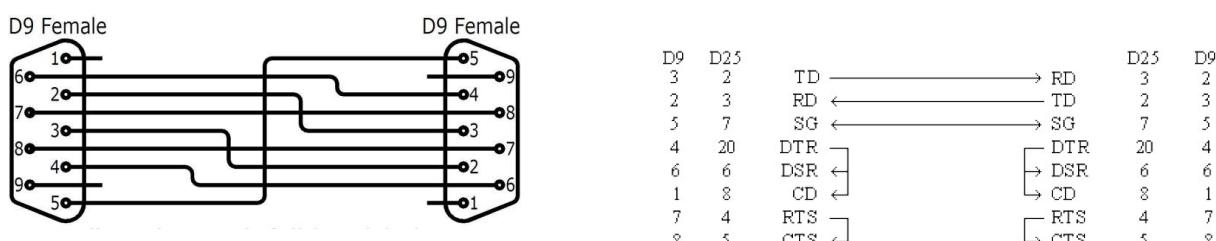
É adicionado um bit pelo dispositivo emissor ao carácter a enviar. O dispositivo receptor após leitura do carácter calcula o bit de paridade e, compara-o com o bit de paridade recebido do dispositivo emissor.

Quando é selecionado o controlo de paridade par: o somatório de todos os bits a nível lógico “1” incluindo o bit de paridade é par. O controlo de paridade ímpar é semelhante.

- **Controlo de fluxo**: RTS, Xon/Xoff, Sem controlo de fluxo;

- **Stop Bits**, número de bits após envio de um caractere: 1 ou 2 stop bits; para definir o fim de transmissão do carácter.

Para os equipamentos comunicarem é necessário interligar os dois equipamentos através de um cabo (conjunto de condutores), como o indicado na figura:



Os computadores mais recentes não dispõem de porta série RS232, mas sim de portas USB, sendo por isso necessário utilizar um adaptador USB/RS232 e instalar o driver apropriado.

4.1.1. Topologia

Neste contexto, por topologia entende-se a forma geométrica como os equipamentos se encontram fisicamente ligados entre si. Existem várias topologias possíveis, a saber: barramento, anel, árvore, estrela e ainda uma topologia mista/híbrida

No caso do protocolo Rs232, está prevista a ligação entre **dois equipamentos** apenas, não se trata por isso de uma verdadeira rede, mas sim uma ligação ponto a ponto, entre dois equipamentos.

4.1.2. Meio comunicação

Os fios de cobre, os cabos coaxiais, a fibra óptica e mesmo o ar, são alguns de meios de comunicação que podem ser usados para transmitir dados entre equipamentos.

No caso vertente, o protocolo Rs232 define a utilização do **fio de cobre** para interligar os dois equipamentos.

4.1.3. Sinais eléctricos

Dependendo do meio de comunicação utilizado, alguns protocolos de comunicação podem optar pela utilização de sinais eléctricos, ópticos, ou de rádio frequência para transmitir “bits” entre equipamentos.

No caso do protocolo Rs232 é aplicada uma tensão positiva, **entre 5 e 25 volts**, para enviar um bit com o valor lógico “0”, e uma tensão negativa, **entre -5 e - 25 volt**, para enviar um bit com o valor lógico “1”.

A Figura 4.2 descreve como os sinais Rs232 são gerados a partir de tensões TTL. As tensões *Transistor Transistor Logic* - TTL só podem assumir os valores 0 ou 5 volt, impõem-se por isso converter estas tensões nas tensões definidas no protocolo Rs232.

O tipo de sinais eléctricos aplicados a cada um dos pinos Tx e Rx destas fichas encontram-se regulados pela norma V.28.

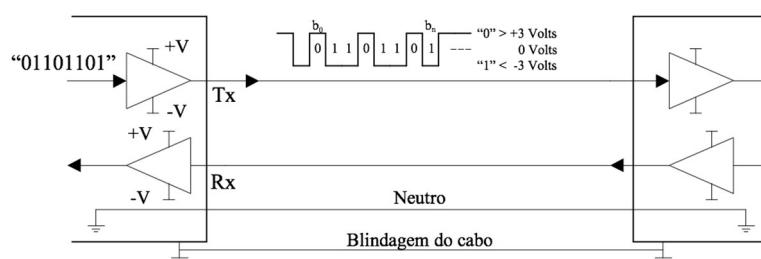


Figura 4.2: Transmissor, Receptor Rs232, conversão TTL

Este protocolo define também o tempo que o emissor deve manter a tensão constante, positiva ou negativa, por cada bit enviado (*bit time*). Quanto menor for este tempo, maior é o número de bits que o emissor pode enviar por segundo. Várias **taxas de transferência** (*baudrate*) estão previstas neste protocolo: 150, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bits/s.

A Figura 4.3 ilustra as tensões e a frequência do sinal eléctrico aplicado num fio de cobre durante uma transmissão, com uma taxa de 9600 bits/s. Pode observar-se que é aplicada uma tensão negativa de -5 volt, durante 1/9600 seg (*bit time*) para transmitir um bit com o valor lógico “1”.

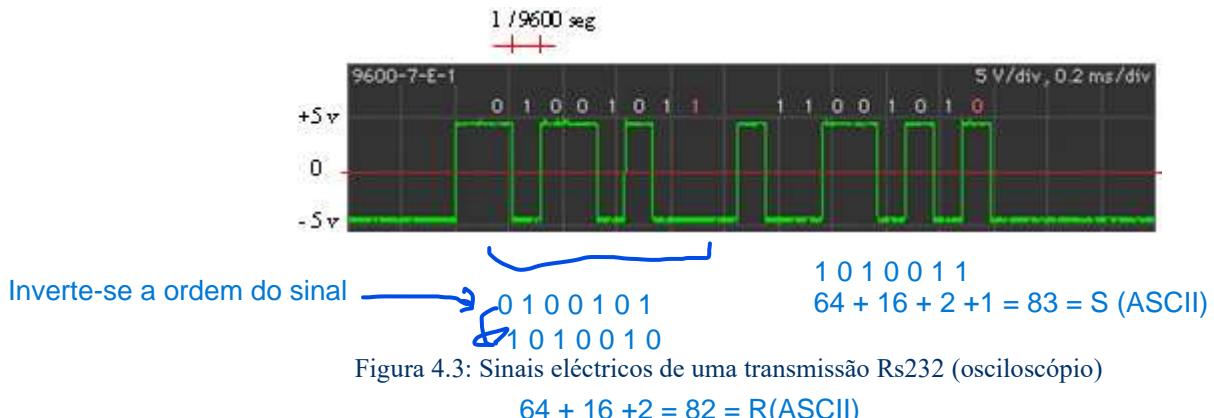


Figura 4.3: Sinais eléctricos de uma transmissão RS232 (osciloscópio)

$$64 + 16 + 2 = 82 = \text{R(ASCII)}$$

4.1.4. Formato da palavra série

Cada palavra série é constituída por um “*start bit*”, seguido de 5, 6, 7 ou 8 *bits de dados*, um *bit de paridade* e no final o “*stop bit*”:

- O “*start bit*” corresponde a uma tensão positiva aplicada durante um “*bit time*”;
- Podem ser enviados 7 ou 8 *bits de dados* em cada palavra série;
- O *bit de paridade* é opcional, pode ou não ser utilizado. No caso de ser utilizada a paridade par, este bit assume o valor “1” ou “0” de forma a que o número de bits a “1” enviados na palavra série seja sempre um número par.
- O “*stop bit*” corresponde a uma tensão negativa aplicada no fio de cobre durante 1, 1½, ou 2 “*bit time*”.

A Figura 4.4 ilustra o envio de uma palavra série com uma taxa de transferência de 9600 bit/s, 7 bits de dados, paridade par - “Even” e 1 *stop bit*.

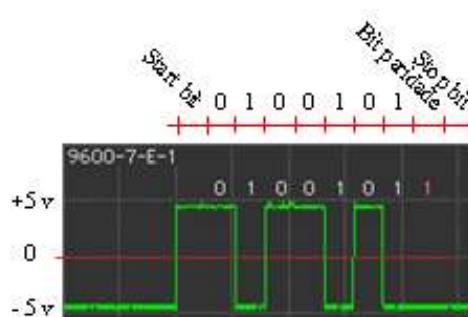


Figura 4.4: Palavra série (osciloscópio)

4.1.5. Detecção de erros de transmissão

O protocolo RS232 prevê o envio de um bit de paridade para permitir que o equipamento receptor possa detectar eventuais erros de transmissão. É esse o papel do bit de paridade representado a vermelho na Figura 4.4. Este bit é adicionado automaticamente pelo emissor e é analisado pelo receptor.

Se os dois equipamentos forem previamente configurados para utilizarem uma comunicação com paridade par, e se dos 7 ou 8 bits de dados enviados numa palavra série, apenas 3 tiverem o valor lógico “1”, o número de “Uns” a enviar nessa palavra seria ímpar. Neste caso, o bit de paridade da palavra assume automaticamente o valor “1”, de forma que o número total de bits a “1” numa palavra série passe a ser sempre um número par (numa comunicação com paridade par).

Desta forma, o receptor sabe que receberá sempre um número par de “1” em cada palavra série, caso contrário houve um erro de transmissão.

4.1.6. Tipo de diálogo

Quando apenas um dos equipamentos pode enviar dados e outro se limita a receber, diz-se que têm um diálogo do tipo “*simplex*”.

Quando os dois equipamentos podem receber e enviar dados, mas não em simultâneo, diz-se que têm um diálogo do tipo “*half duplex*”.

Quando os dois equipamentos podem enviar e receber dados em simultâneo, têm um diálogo do tipo “*full duplex*”.

Numa comunicação Rs232 existem dois fios de cobre distintos para a transmissão de dados, um para o envio e outro para a recepção, por essa razão os dois equipamentos podem receber e enviar dados em simultâneo. Diz-se por isso que estamos na presença de um diálogo do tipo “*Full duplex*”.

4.1.7. Controlo do fluxo de dados

O protocolo Rs232 prevê duas formas do equipamento receptor pedir ao emissor para suspender por momentos o envio de dados. Na terminologia inglesa, controlo de fluxo designa-se por “*Handshake*”.

Uma das formas de controlar o fluxo de informação entre os equipamentos consiste no envio, do receptor para o emissor, de uma palavra série especial (Xoff- 0x13 = 00010011 bin = 19 dec).

Quando emissor recebe a palavra Xoff suspende o envio de novos dados até que o receptor lhe envie a palavra Xon (Controlo de fluxo por Xon-Xoff).

A segunda forma de controlar o fluxo de informação entre o emissor e o receptor, consiste na utilização de fios de cobre adicionais a ligar as fichas Rs232 dos dois equipamentos. Mais exactamente, fios de cobre adicionais a ligar o pino *Request to Send - RTS* do emissor ao *Clear To Send – CTS* do receptor. Quando emissor pretender enviar dados ativa o pino *RTS* e o receptor pode ativar o seu pino *CTS* para indicar que aceita receber dados (Controlo de fluxo por hardware).

4.1.8. Fichas EIA232

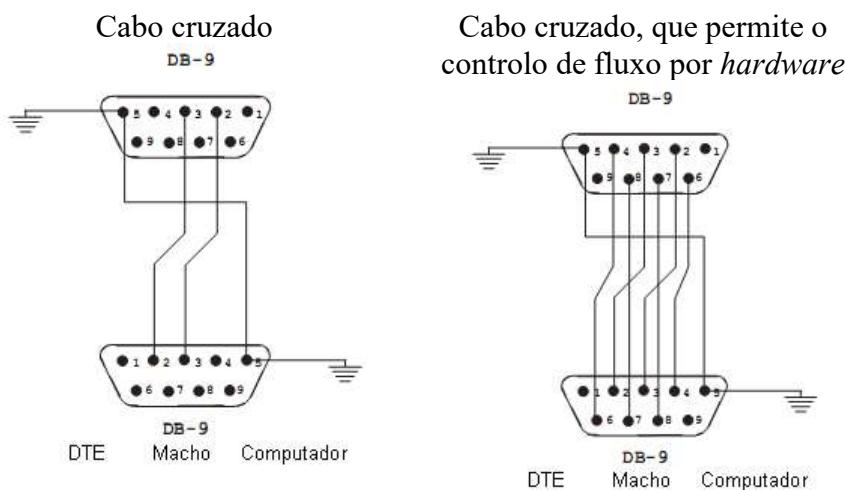
As fichas DB25 e DB9 descritas na Figura , são fichas macho, disponíveis nos computadores e outros equipamentos do tipo DTE - *Data Terminating Equipment*. A ficha RJ45 presentes na figura é fêmea e as fichas DB-9 e DB-25 são macho.

DB-25	CONNECTOR DB-9	RJ-45	FUNCTION	CODE NAME	DIRECTION
1	3	4	Ground	G	
2	6	Transmit data	TXD	Output	
3	2	Receive data	RXD	Input	
4	7	Request to send	RTS	Output	
5	8	Clear to send	CTS	Input	
6	6	Data set ready	DSR	Input	
7	5	Chassis ground	G		
8	1	Carrier detect	CD		
20	4	Data terminal ready	DTR	Output	
22	9	Ring indicator	RI	Input	

Figura 4.5: Fichas EIA232 [Sanchez' 2007]

4.1.9. Cabos Rs232

Em algumas fichas, o pino Tx corresponde ao pino número 2, noutros casos ao número 3. Mas, independentemente de a ficha ser macho ou fêmea, independentemente de o equipamento ser um computador (DTE – *Data Terminal Equipment*) ou um modem (DCE – *Data Communication Equipment*), o pino de transmissão *Tx* de um equipamento liga sempre ao pino *Rx* do outro equipamento. Nalgumas fichas o pino “terra” corresponde ao pino 5, noutras corresponde ao pino 7. Em todo o caso, os pinos “terra” das duas fichas devem ser ligados entre si.



Cabos EIA232 [Sanchez' 2007]

O cabo descrito no lado esquerdo da Figura permite uma comunicação Rs232, contudo este cabo não permite um controlo de fluxo por hardware. Já o cabo da direita permite o controlo de fluxo entre equipamentos através dos pinos *Clear To Send* - CTS e *Request To Send* – RTS.

4.1.10. Conversão dos níveis TTL para os níveis Rs232

Ao contrário do computador, a interface USART/Rs232 de alguns dispositivos pode gerar apenas tensões TTL, tensões de 0 e 5 volt, ao passo que os sinais eléctricos gerados durante uma comunicação Rs232 atingem tensões positivas entre 5 e 25 volt e tensões negativas entre -5 e -25 volt. Por esta razão pode ser necessário utilizar um conversor de tensões. O circuito integrado MAX232 desempenha essa função e converte uma tensão de + 5 v numa tensão negativa de **-5 v**, e uma tensão de 0 volt numa tensão positiva de **+5 v**.

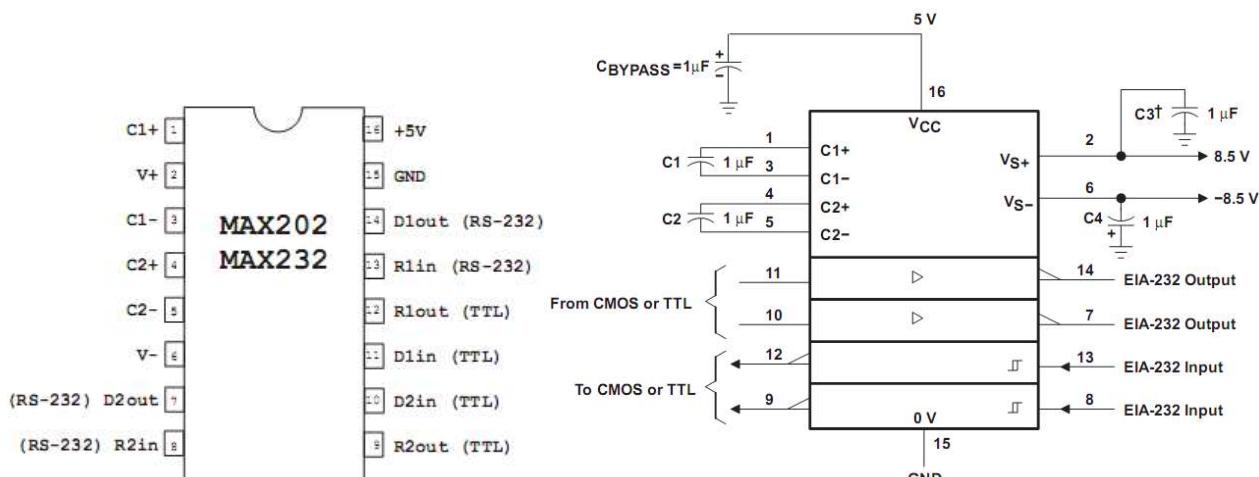
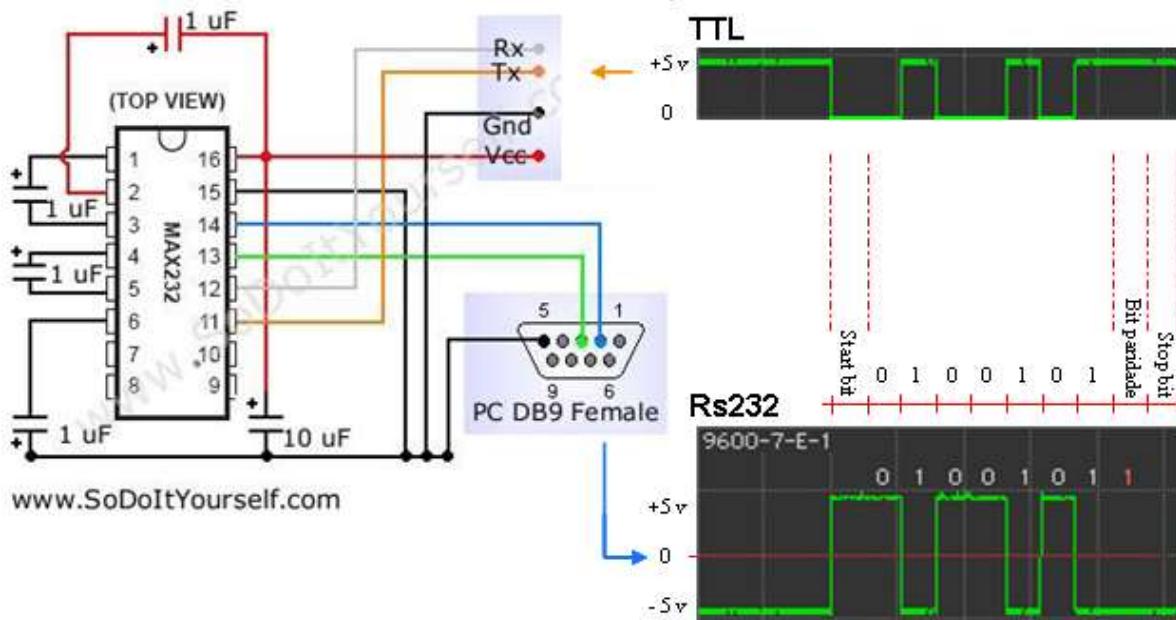


Figura: MAX232 - Conversão de níveis TTL para níveis Rs232

Tabela ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0 0	000	000	NULL	32 20	040	 	Space		64 40	100	@	@	96 60	140	`	`		
1 1	001	001	Start of Header	33 21	041	!	!		65 41	101	A	A	97 61	141	a	a		
2 2	002	002	Start of Text	34 22	042	"	"		66 42	102	B	B	98 62	142	b	b		
3 3	003	003	End of Text	35 23	043	#	#		67 43	103	C	C	99 63	143	c	c		
4 4	004	004	End of Transmission	36 24	044	$	\$		68 44	104	D	D	100 64	144	d	d		
5 5	005	005	Enquiry	37 25	045	%	%		69 45	105	E	E	101 65	145	e	e		
6 6	006	006	Acknowledgment	38 26	046	&	&		70 46	106	F	F	102 66	146	f	f		
7 7	007	007	Bell	39 27	047	'	'		71 47	107	G	G	103 67	147	g	g		
8 8	010	010	Backspace	40 28	050	((72 48	110	H	H	104 68	150	h	h		
9 9	011	011	Horizontal Tab	41 29	051))		73 49	111	I	I	105 69	151	i	i		
10 A	012	012	Line feed	42 2A	052	*	*		74 4A	112	J	J	106 6A	152	j	j		
11 B	013	013	Vertical Tab	43 2B	053	+	+		75 4B	113	K	K	107 6B	153	k	k		
12 C	014	014	Form feed	44 2C	054	,	,		76 4C	114	L	L	108 6C	154	l	l		
13 D	015	015	Carriage return	45 2D	055	-	-		77 4D	115	M	M	109 6D	155	m	m		
14 E	016	016	Shift Out	46 2E	056	.	.		78 4E	116	N	N	110 6E	156	n	n		
15 F	017	017	Shift In	47 2F	057	/	/		79 4F	117	O	O	111 6F	157	o	o		
16 10	020	020	Data Link Escape	48 30	060	0	0		80 50	120	P	P	112 70	160	p	p		
17 11	021	021	Device Control 1	49 31	061	1	1		81 51	121	Q	Q	113 71	161	q	q		
18 12	022	022	Device Control 2	50 32	062	2	2		82 52	122	R	R	114 72	162	r	r		
19 13	023	023	Device Control 3	51 33	063	3	3		83 53	123	S	S	115 73	163	s	s		
20 14	024	024	Device Control 4	52 34	064	4	4		84 54	124	T	T	116 74	164	t	t		
21 15	025	025	Negative Ack.	53 35	065	5	5		85 55	125	U	U	117 75	165	u	u		
22 16	026	026	Synchronous idle	54 36	066	6	6		86 56	126	V	V	118 76	166	v	v		
23 17	027	027	End of Trans. Block	55 37	067	7	7		87 57	127	W	W	119 77	167	w	w		
24 18	030	030	Cancel	56 38	070	8	8		88 58	130	X	X	120 78	170	x	x		
25 19	031	031	End of Medium	57 39	071	9	9		89 59	131	Y	Y	121 79	171	y	y		
26 1A	032	032	Substitute	58 3A	072	:	:		90 5A	132	Z	Z	122 7A	172	z	z		
27 1B	033	033	Escape	59 3B	073	;	;		91 5B	133	[[123 7B	173	{	{		
28 1C	034	034	File Separator	60 3C	074	<	<		92 5C	134	\	\	124 7C	174	|			
29 1D	035	035	Group Separator	61 3D	075	=	=		93 5D	135]]	125 7D	175	}	}		
30 1E	036	036	Record Separator	62 3E	076	>	>		94 5E	136	^	^	126 7E	176	~	~		
31 1F	037	037	Unit Separator	63 3F	077	?	?		95 5F	137	_	_	127 7F	177		Del		

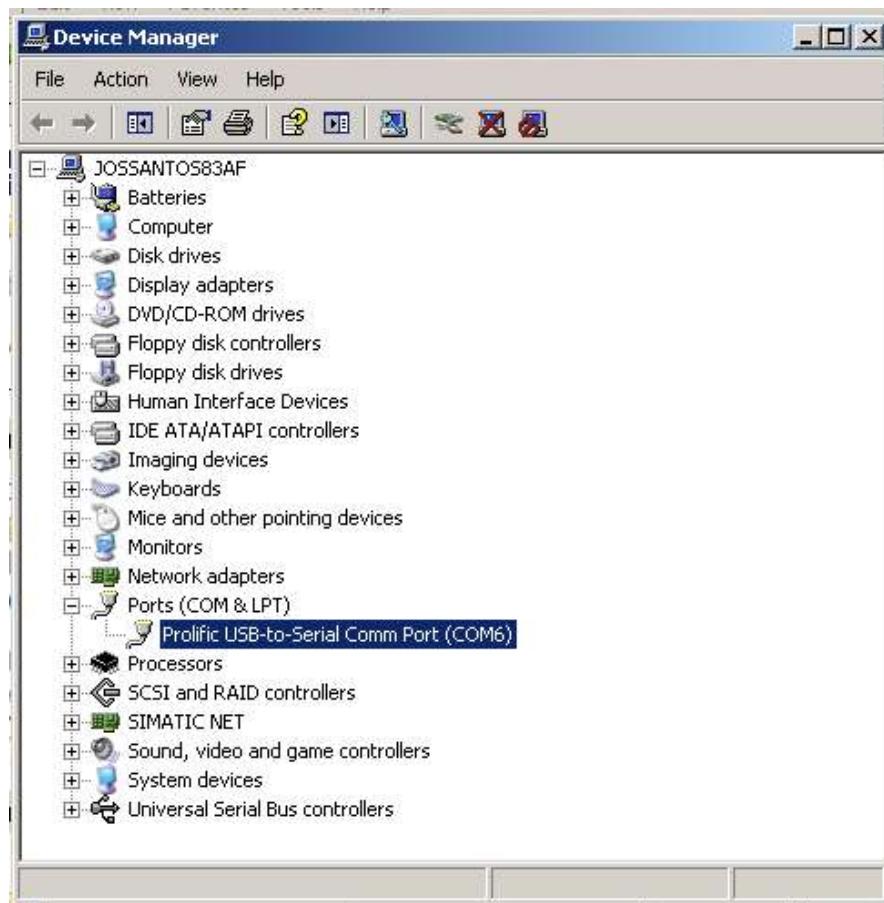
asciicharstable.com

4.2. VBasic - Como utilizar e configurar a interface EIA232 do Computador

Os programas desenvolvidos em VBasic podem configurar e utilizar a interface de comunicação EIA232 do seu computador através de um de dois tipos de objetos: do tipo *Mscomm* ou do tipo *SerialPort*.

O controlo “*SerialPort*” já está disponível na versão base do Vbasic 2012, mas se optarmos por utilizar o controlo “*Mscomm*” temos de o instalar manualmente no sistema operativo *Windows*.

A figura seguinte apresenta a janela do “*Device Manager*” do sistema operativo *Windows*. Neste exemplo, o computador tem uma porta série com a designação *COM6*.



4.2.1. Instalação dos drivers dos conversores USB-Rs232

As portas série RS232 já raramente equipam os computadores atuais. Estas foram substituídas por portas USB. Assim, é necessário transformar uma porta USB do computador numa porta série. Para isso utilizam-se os chamados conversores USB-RS232. Cada conversor tem um driver específico. Assim, o mais adequado será instalar os seguintes drivers windows para os três diferentes conversores:

<p>RS232 ATEN USB-RS232 Serial Converter http://www.aten.com/global/en/products/usb-&-thunderbolt/usb-converters/uc232a/#.WdIvjGhSyW9</p>	
<p>RS232 Prolific PL2303 ShowProduct.aspx?p_id=225&pcid=41">http://www.prolific.com.tw/US>ShowProduct.aspx?p_id=225&pcid=41</p>	
<p>RS485 CH340 https://sparks.gogo.co.nz/ch340.html</p>	

Para verificar se os drivers estão a funcionar corretamente devem-se verificar as portas que estão a ser utilizadas usando o “device manager”. Se, depois de ligar um equipamento, na secção das portas COM aparecer um símbolo amarelo quer dizer que não existe um driver para o equipamento que está ligado.

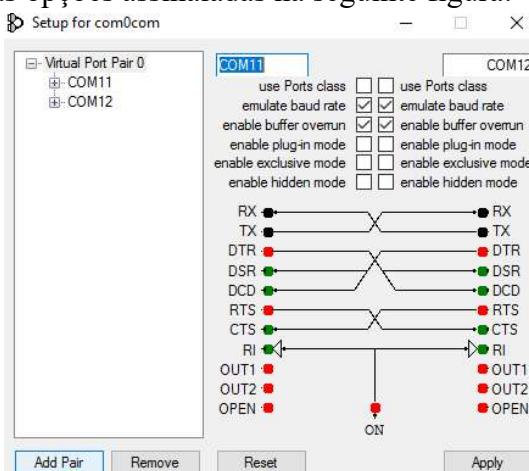
4.2.2. Instalação do com0com

Se não tiver um conversor USB/Rs232 é necessário instalar o software com0com (<http://com0com.sourceforge.net/>), open source e sem data de expiração. Para instalar aceda ao link <https://sourceforge.net/projects/com0com/files/latest/download>

Depois de instalar a aplicação, confirme no gestor de dispositivos se foi criado um par de portas COM. Se for este o caso em princípio não necessita de fazer os passos seguintes. Caso contrário pode fazer a configuração do programa executando o seguinte ficheiro

C:\Program Files (x86)\com0com\setupg.exe

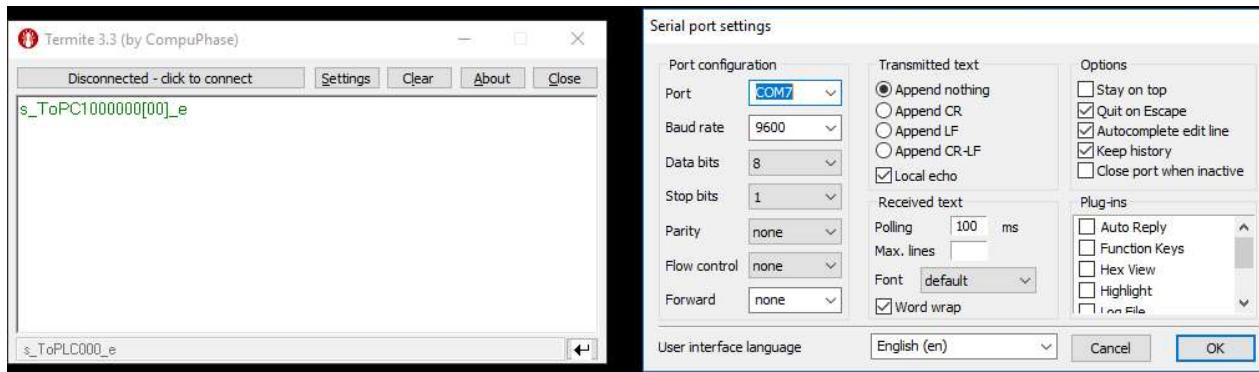
Na janela, devem selecionar as opções assinaladas na seguinte figura:



Caso não consiga criar corretamente as portas COM veja o seguinte manual:
<http://www.softwx.com/weather/virtualvp/VirtualSerialPorts64bitHelp.pdf>

4.2.3. Software de comunicação série Termite

O Termite https://www.compuphase.com/software_termite.htm é um software que permite a comunicação do computador com outros equipamentos por porta série. Para instalar, fazer download aqui <https://www.compuphase.com/software/termite-3.3.exe>. Na imagem em baixo por ver-se o software. Para utilizar o programa é preciso configurá-lo primeiro, indicando os parâmetros de comunicação adequados. Para fazer isso clicar em settings e depois escolher os parâmetros corretos.



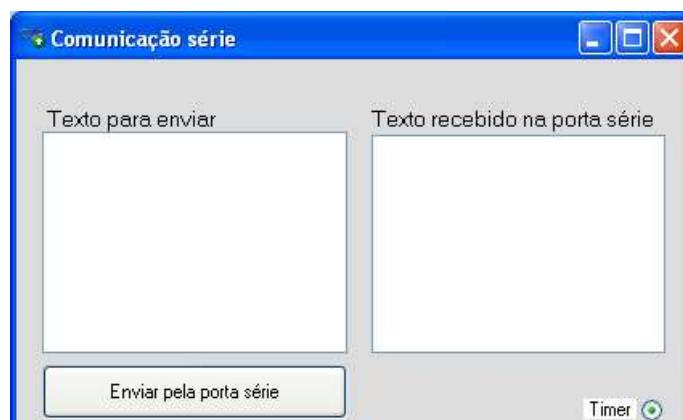
4.2.4. Exemplo - programa “SerialPortTxRx”

O programa da figura seguinte permite configurar a porta série do computador, mais exatamente a sua porta “*Com6*”, com uma taxa de transferência de 9600 bit/seg, 1 bit de paridade par (Even), 8 bits de dados e 2 stop bit.

Depois disto, sempre que o utilizador selecionar o botão “*Enviar*” serão enviados para a porta “*Com6*” todos os caracteres existentes na caixa de texto da esquerda “*txtEnviar*”.

Sempre que o computador receber uma palavra série na porta “*Com6*”, contendo por exemplo um carácter ASCII, o objeto “*SerialPort*” automaticamente e sem intervenção do nosso programa armazena esse (s) carácter (es) na sua memória interna.

Sempre que o *Timer1* gerar um evento, neste exemplo de 500 em 500 ms, os caracteres entretanto recebidos e armazenados nessa memória interna serão então visualizados na janela de texto da direita “*TxtReceber*”. Não se esqueça de “arrastar” um objeto do tipo *SerialPort* para a interface (form).



```

Public Class Form1
    Dim tx, rx As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 100
        Timer1.Start()
        SerialPort1.PortName = "com6"
        SerialPort1.BaudRate = 9600
        SerialPort1.Parity = IO.Ports.Parity.Even
        SerialPort1.DataBits = 8
        SerialPort1.StopBits = 2
        SerialPort1.Handshake = IO.Ports.Handshake.None
        SerialPort1.Encoding = System.Text.UTF8Encoding.UTF8
        SerialPort1.Open()
    End Sub

    Private Sub BtEnviar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtEnviar.Click
        tx = TxtEnviar.Text
        SerialPort1.WriteLine(tx)
        'Dim barray() As Byte = {48, 127, 230, 51, 52, 0}
        'SerialPort1.Write(barray, 0, 6)
    End Sub

    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As ...
        rx = rx & SerialPort1.ReadExisting()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        RadioButton1.Checked = Not (RadioButton1.Checked)
        txtReceber.Text = rx
    End Sub

    Private Sub AjudaToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim frmAjuda1 As New frmAjuda
        frmAjuda1.ShowDialog()
    End Sub

    Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As ...
        If SerialPort1.IsOpen() Then
            SerialPort1.Close()
        End If
    End Sub

End Class

```

4.2.5. Objetos do tipo SerialPort

O exemplo anterior permitiu dar a conhecer algumas propriedades dos objetos do tipo “*SerialPort*”. Contudo, estes objetos possuem muitos outros métodos e propriedades que serão aqui apresentados de uma forma mais detalhada e sistemática.

http://msdn.microsoft.com/en-us/library/system.io.ports.serialport_members.aspx

Nesta secção vamos usar um objeto do tipo *SerialPort* com o nome *SerialPort1*. Estes objetos usam a tabela ASCII para codificar cada carácter num conjunto de 7 ou 8 bits e depois enviam-nos pela linha Rs232 sequencialmente.

Fazem o processo inverso sempre que recebem um conjunto de 8 ou 7 bits na porta série, mas ignoram o bit mais significativo, o 8º bit, e convertem os 7 bits menos significativos num carácter ASCII. Por essa razão o valor mais alto que pode ser codificado é o valor $2^7 = 127$ que corresponde na tabela ASCII à tecla ‘DEL’. Mesmo se o comprimento da palavra série for de 8 bits, o bit mais significativo é transmitido mas não tem significado.

Para enviar ou receber palavras série, por exemplo caracteres ASCII, através da porta Rs232 do seu computador sem utilizar modems e sem controlo de fluxo por hardware, apenas necessita de utilizar as propriedades que a seguir se descrevem:

PortName

Compete ao programa indicar o número da porta série, de entre as que estiverem disponíveis no computador, que este objeto irá usar.

Ex: *SerialPort1.PortName* = “COM1”

Compete ao programa definir os parâmetros da comunicação série: taxa de transferência, bit paridade par/ímpar/nenhum, nº de bits de dados de cada palavra série, nº de stop bit.

SerialPort1.BaudRate = 9600

SerialPort1.DataBits = 8
 7

SerialPort1.Parity = *IO.Ports.Parity*.
 .Even
 .Odd
 .Space
 .None

SerialPort1.StopBits = *IO.Ports.StopBits*.
 .None
 .One
 .OnePointFive
 .Two

Open

Para que um objeto do tipo *SerialPort* configure e passe a monitorizar a porta série do computador é necessário que o programa contenha a instrução

SerialPort1.Open()

Write

O método *Write* de um objeto do tipo *SerialPort*, envia para a porta série do computador um conjunto de caracteres. Lembre-se que cada um dos três caracteres da string “ola” enviada neste exemplo é codificado em 7 bits de acordo com a tabela ASCII. Por isso serão enviadas três palavras série, cada uma contendo os 7 bits respectivos. Cada uma destas 3 palavras série começa por um start bit, seguido dos 7 bits e é terminada com o stop bit. Pode também ter um bit de paridade antes de cada stop bit.

```
SerialPort1.Write("ola")
```

Existe um segundo método chamado **Write com três parâmetros** que permite enviar um array de bytes, neste caso o método *write* não usa a tabela ASCII para converter caracteres/strings, mas sim, envia o valor binário de cada número.

O primeiro parâmetro contém o nome do array, o segundo parâmetro contém o índice do primeiro elemento do array que se quer enviar e o terceiro parâmetro contém o número de bytes que se quer enviar:

```
Write ( buffer As Byte(), offset As Integer, count As Integer)
```

Por exemplo, definimos um array de bytes chamado *barray* do tipo array de bytes.

```
Dim barray() As Byte = {48, 49, 50, 51, 52, 0}
```

O primeiro elemento do array contém o número 48 ‘*barray(0) tem o valor 48*

O segundo elemento do array contém o número 49 ‘*barray(1) tem o valor 49*

O último elemento do array contém o número 0

‘*barray(5) tem o valor 0, não confundir com o carácter ‘0’ que na tabela ASCII corresponde ao valor 48.*

Neste exemplo, todos os seis elementos do array serão enviados quando o método *Write* for executado:

```
SerialPort1.Write(barray, 1, 6)
```

Encoding

Para que os caracteres sejam codificados com 8 bits e não só com 7bits, passando a estar disponível um maior leque de caracteres para enviar ou receber. Tem de configurar a propriedade *Encoding* deste tipo de objetos.

```
SerialPort1.Encoding = System.Text.UTF8Encoding.UTF8
```

ReadExisting

O método *ReadExisting()* devolve ao programa o conjunto de caracteres entretanto recebidos na porta série do computador. Convém lembrar que o objeto *SerialPort* monitoriza de forma autónoma a porta série e armazena numa das suas variáveis internas os caracteres que forem chegando ao computador vindos de um modem externo, de um PLC, ou de outro computador.

```
TextBox1.text= ReadExisting()
```

4.2.6. Se o seu computador estiver ligado a um modem

Se quiser usar o controlo de fluxo por hardware “*handshaking por hardware*” necessita de utilizar os pinos *CTS*, *RTS* da ficha *Rs232* e pode interagir com eles usando as propriedades de um objeto do tipo *SerialPort* que a seguir se descrevem:

Handshake

Esta propriedade pode assumir os valores:

Sem handshaking

SerialPort1.Handshake = 0 ‘*IO.Ports.Handshake.None*

Handshaking com caracteres Xon/Xoff

SerialPort1.Handshake = 1 ‘*IO.Ports.Handshake.XOnXOff*

Handshaking através dos pinos CTS e RTS

SerialPort1.Handshake = 2 ‘*IO.Ports.Handshake.RequestToSend*

Handshaking por Xon/Xoff e pelos pinos CTS e RTS

SerialPort1.Handshake = 3 ‘*IO.Ports.Handshake.RequestToSendXOnXOff*

Se escolher o valor 2 ou 3, mesmo que o seu programa envie dados para um objeto do tipo *SerialPort* este só os enviará para o exterior se o pino *CTS* do computador for ativado pelo *modem* ou por outro dispositivo externo.

RtsEnable – Request To Send enable

Quando o programa escreve o valor *true* nesta propriedade o pino *RTS* da ficha *Rs232* do computador torna-se ativa e passa a ter uma tensão. Desta forma é possível perguntar ao modem se este pode receber mais caracteres vindos do computador para posteriormente os enviar pela linha telefónica.

CtsHolding – Clear To Send

Enquanto o *modem* aplicar uma tensão ao pino *CTS* da ficha *Rs232* do computador esta propriedade assume o valor *true*, e significa que o computador pode continuar a enviar caracteres para o modem.

DSR Holding – DCE Set Ready

Quando o modem está activo e pronto a funcionar aplica uma tensão no pino *DSR* do computador, nessa altura a propriedade *DSR Holding* assume o valor *true*.

Ex:

IF SerialPort1.DsrHolding then “O modem aplicou uma tensão ao pino DSR do computador ”

CD Holding – Carrier Detect

Compete ao *modem* aplicar uma tensão neste pino da ficha *Rs232* do computador se tiver uma ligação telefónica à espera de ser atendida ou já estabelecida.

Esta propriedade pode assumir o valor *true* ou *false*. Se o pino *CD* da ficha *Rs232* estiver activo (tiver tensão) a propriedade *CD Holding* assume o valor *true*.

Ex:

IF SerialPort1.CD Holding then “O modem informa o computador de ligação telefónica ativa ”

DTREnable – Data Terminal Ready enable

O computador ativa o pino *DTR* da sua ficha Rs232 para indicar ao seu *modem* que pode aceitar “ligações telefónicas”. Para isso o programa deve fazer com que esta propriedade assuma o valor *true*. Ex:

SerialPort1.DtrEnable = true

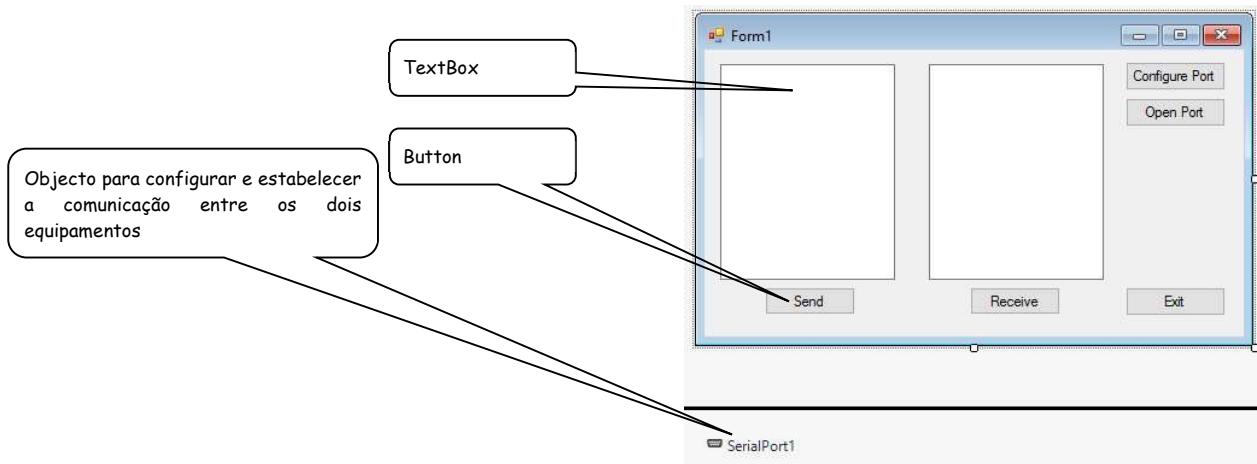
Se o programa fizer *SerialPort1.DtrREnable = false* o seu *modem* “desligará a ligação telefónica”.



Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

4.2.7. Exemplo: Programa simples de comunicação



Desenvolvimento de uma aplicação simples para envio e receção de mensagens entre dois computadores. O programa deverá conter:

- **Botão para configurar a porta de comunicação**, atribuindo os parâmetros de comunicação 9600 bps, sem paridade (None), 8 bits de dados e um stop bit;
- **Botão para iniciar/fechar** a comunicação entre computadores. Utilizar o mesmo botão alterando o texto;
- **Caixa de texto para escrever e enviar a mensagem;**
- **Caixa de texto para visualização da mensagem recebida;**

Utilizar o objeto **SerialPort**, para configurar e estabelecer a comunicação entre computadores, principais propriedades:

<code>SerialPort1.PortName="COM1"</code>	- Nome
<code>SerialPort1.ReadExisting</code>	- Leitura dos dados recebidos
<code>SerialPort1.Write (Txt_Envio.Text)</code>	- Envio dos dados
<code>SerialPort1.Isopen</code>	- Verifica se está aberta
<code>SerialPort1.Open</code>	- Abrir porta
<code>SerialPort1.Close</code>	- Fechar Porta

No windows, para consultar as propriedades da porta de comunicação (COM) associada ao conversor USB/RS232, consultar o “Painel de Controlo” -> “System and Security” -> “Device Manager”. Ver secção 2.

Listagem do programa:

```
...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex1\Ex1\Form1.vb
1  Public Class Form1
2
3      Dim data_received As String
4
5      '-----
6      'Initialization routines
7      '-----
8      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
9          TextBoxSend.Text = "My first message!"
10     End Sub
11
12     '-----
13     'Serial port configuration routines
14     '-----
15
16     'Configure serial communication parameters (hardcoded for now)
17     Private Sub ButtonConfigurePort_Click(sender As Object, e As EventArgs) Handles ButtonConfigurePort.Click
18         With SerialPort1
19             .PortName = "COM5" 'Check with device manager for the open port
20             .BaudRate = 9600
21             .Parity = IO.Ports.Parity.None
22             .DataBits = 8
23             .StopBits = IO.Ports.StopBits.One
24             .Encoding = System.Text.Encoding.UTF8
25         End With
26
27     End Sub
28
29     'Open and close serial port
30     Private Sub ButtonOpenClosePort_Click(sender As Object, e As EventArgs) Handles ButtonOpenClosePort.Click
31         If SerialPort1.IsOpen Then 'If opened, close. Change text on button
32             SerialPort1.Close()
33             ButtonOpenClosePort.Text = "Open Port"
34             ButtonOpenClosePort.BackColor = Color.LightGray
35         Else 'If closed, open. Change text on button
36             SerialPort1.Open()
37             ButtonOpenClosePort.Text = "Close Port"
38             ButtonOpenClosePort.BackColor = Color.Red
39         End If
40
41     End Sub
42
43     '-----
44     'Communication routines
45     '-----
46
47     'Send data in TextBoxSend
48     Private Sub ButtonSend_Click(sender As Object, e As EventArgs) Handles ButtonSend.Click
49         SerialPort1.WriteLine(TextBoxSend.Text)
50     End Sub
51
52     Private Sub ButtonReceive_Click(sender As Object, e As EventArgs) Handles
```

```
...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex1\Ex1\Form1.vb 2
        ButtonReceive.Click
53      data_received = data_received & SerialPort1.ReadExisting 'read from    ↵
          serial port
54      TextBoxReceive.Text = data_received 'copy from data_received
55      data_received = "" 'erase text in data_received
56  End Sub
57
58  '-----
59  'Other routines
60  '-----
61  Private Sub ButtonExit_Click(sender As Object, e As EventArgs) Handles    ↵
        ButtonExit.Click
62      End
63  End Sub
64
65
66 End Class
67
```



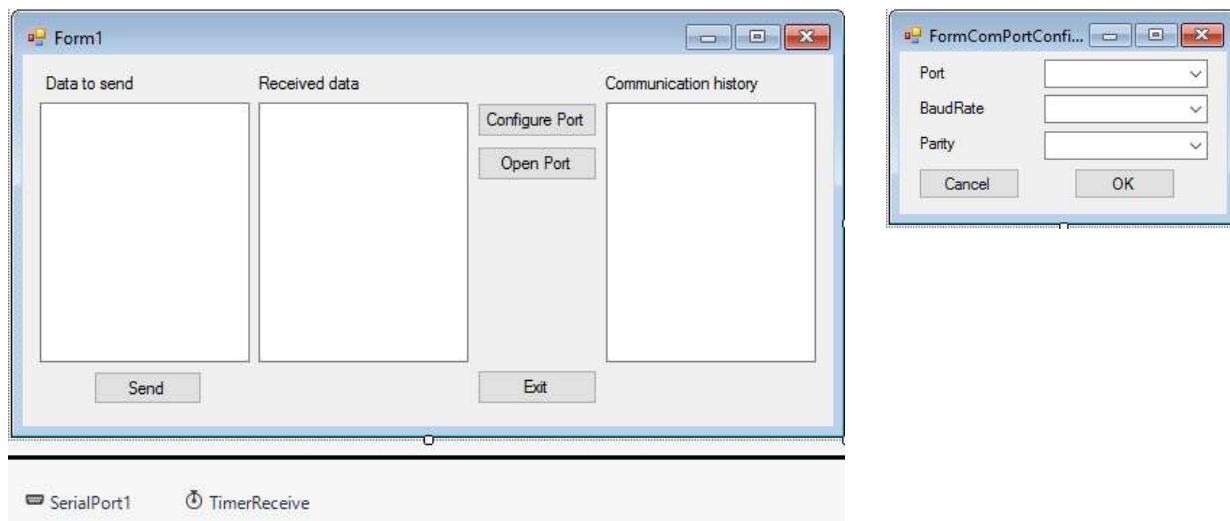
Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

4.3.Trabalho nº 3 - VBasic, Comunicação Rs232 entre computadores

Neste trabalho, pretende-se melhorar o programa do exemplo anterior, acrescentando as seguintes funcionalidades:

- Leitura automática (periódica) das mensagens usando um objeto Timer e removendo o botão
- Implementação de um registo do histórico de mensagens, com indicação da hora de receção da mensagem
- Utilização de um elemento *Module* para definição das variáveis globais dos parâmetros da porta. Para simplificar o programa, serão configurados apenas os parâmetros: nome da porta, baud rate e paridade. Todos os outros devem ficar no seu estado default.
- Utilização de um *Form* adicional para configuração dos parâmetros da porta
- Utilização da variável *is_configuration_valid* para gestão dos comportamentos do programa, i.e., só abre a porta COM se a configuração dos parâmetros for válida.
- Fazer o disable dos botões do programa de forma contextual, i.e. enquanto a porta COM estiver fechada, o botão de envio deve estar *disabled*.



Listagem do programa da Form1:

...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex2\Ex2\Form1.vb

```

1  Public Class Form1
2
3      Dim data_received As String
4
5      '-----
6      'Initialization routines
7      '-----
8      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
9          TextBoxSend.Text = "My first message!"
10
11         'Configure TimerReceive to tic every x millisecs
12         TimerReceive.Interval = 300
13         TimerReceive.Enabled = False 'Only tic when the Port is Open
14
15         'To make sure we do not open the Port without a valid configuration
16         ModuleComPortParameters.configuration_is_valid = False
17     End Sub
18
19     '-----
20     'Serial port configuration routines
21     '-----
22
23     'Configure serial communication parameters
24     Private Sub ButtonConfigurePort_Click(sender As Object, e As EventArgs) Handles ButtonConfigurePort.Click
25         FormComPortConfiguration.StartPosition = FormStartPosition.CenterParent
26         FormComPortConfiguration.ShowDialog()
27
28         'Now setup the object SerialPort1 using the variables from the
29         'ModuleComPortParameters
30         'Only use the values if configuration was declared valid
31         If ModuleComPortParameters.configuration_is_valid = True Then
32             With SerialPort1
33                 .PortName = ModuleComPortParameters.port
34                 .BaudRate = ModuleComPortParameters.baud_rate
35                 .Parity = ModuleComPortParameters.parity
36             End With
37         End If
38     End Sub
39
40     'Open and close serial port
41     Private Sub ButtonOpenClosePort_Click(sender As Object, e As EventArgs) Handles ButtonOpenClosePort.Click
42         If ModuleComPortParameters.configuration_is_valid = True Then
43             If SerialPort1.IsOpen Then 'If opened, close. Change text on button
44                 SerialPort1.Close()
45                 ButtonOpenClosePort.Text = "Open Port"
46                 ButtonOpenClosePort.BackColor = Color.LightGray
47                 TimerReceive.Enabled = False 'Stop receiving periodically
48             Else 'If closed, open. Change text on button
49                 SerialPort1.Open()
50                 ButtonOpenClosePort.Text = "Close Port"
51                 ButtonOpenClosePort.BackColor = Color.Red
52                 TimerReceive.Enabled = True 'Start receiving periodically

```

```
53         End If
54     Else
55         MsgBox("Cannot open port with an invalid configuration!")
56     End If
57 End Sub
58
59 '-----
60 'Communication routines
61 '-----
62
63 'Send data in TextBoxSend
64 Private Sub ButtonSend_Click(sender As Object, e As EventArgs) Handles ButtonSend.Click
65     SerialPort1.WriteLine(TextBoxSend.Text)
66 End Sub
67
68 'Receive data
69 Private Sub TimerReceive_Tick(sender As Object, e As EventArgs) Handles TimerReceive.Tick
70     data_received = data_received & SerialPort1.ReadExisting 'read from
71             serial port
72
73     If Len(data_received) > 0 Then 'Copy from reception buffer to TextBox
74         if there is something there
75         TextBoxReceive.Text = data_received 'copy from data_received
76         TextBoxHistory.Text = TimeOfDay.ToString + ":" +
77             data_received + vbNewLine +
78             TextBoxHistory.Text 'Append data to history
79         data_received = "" 'erase text in data_received
80     End If
81 End Sub
82
83 '-----
84 'Other routines
85 '-----
86
87
88 End Class
89
```

Listagem do programa da FormComPortConfiguration

```
...-2018\Pratica\Aula3\Ex2\Ex2\FormComPortConfiguration.vb 1
1 Public Class FormComPortConfiguration
2     '-----
3     'Initialization routines
4     '-----
5
6     Private Sub FormComPortConfiguration_Load(sender As Object, e As EventArgs) Handles MyBase.Load
7         'Configure the combo boxes with the appropriate options
8
9         'ComboBox Port Configuration
10        Dim available_ports As Array = IO.Ports.SerialPort.GetPortNames
11        Dim i As Integer 'cycle all available ports
12        For i = 0 To UBound(available_ports)
13            ComboBoxPort.Items.Add(available_ports(i))
14        Next
15        ComboBoxPort.SelectedIndex = 0 'Default value
16
17        'ComboBox Baud rate configuration
18        ComboBoxBaudRate.Items.Clear()
19        ComboBoxBaudRate.Items.Add("4800")
20        ComboBoxBaudRate.Items.Add("9600")
21        ComboBoxBaudRate.Items.Add("19200")
22        ComboBoxBaudRate.SelectedIndex = 2 'Default
23
24        'ComboBox parity configuration
25        ComboBoxParity.Items.Clear()
26        ComboBoxParity.Items.Add(IO.Ports.Parity.None)
27        ComboBoxParity.Items.Add(IO.Ports.Parity.Odd)
28        ComboBoxParity.Items.Add(IO.Ports.Parity.Even)
29        ComboBoxParity.SelectedIndex = 2
30
31    End Sub
32
33    '-----
34    'Button press routines
35    '-----
36
37    Private Sub ButtonCancel_Click(sender As Object, e As EventArgs) Handles ButtonCancel.Click
38        ModuleComPortParameters.configuration_is_valid = False
39        Me.Close()
40    End Sub
41
42    Private Sub ButtonOK_Click(sender As Object, e As EventArgs) Handles ButtonOK.Click
43        'Copy the values in the combo boxes to the variables in the
44        'ModuleComPortParameters
45        'In some cases, must use a switch case to map from string to IO.Ports
46        'type values
47
48        ModuleComPortParameters.port = ComboBoxPort.Text 'String to String
49        mapping, no sweat
50        ModuleComPortParameters.baud_rate = ComboBoxBaudRate.Text 'String to
51        Integer, automatic conversion
52
53        Select Case ComboBoxParity.Text 'String to IO.Ports.Parity, trickier
```

:

```
...-2018\Pratica\Aula3\Ex2\Ex2\FormComPortConfiguration.vb
50      Case "None"
51          ModuleComPortParameters.parity = IO.Ports.Parity.None
52      Case "Even"
53          ModuleComPortParameters.parity = IO.Ports.Parity.Even
54      Case "Odd"
55          ModuleComPortParameters.parity = IO.Ports.Parity.Odd
56  End Select
57
58  'Set configuration is valid flag
59  ModuleComPortParameters.configuration_is_valid = True
60
61  Me.Close()
62 End Sub
63 End Class
```

Listagem do programa da ModuleComPortParameters

```
...7-2018\Pratica\Aula3\Ex2\Ex2\ModuleComPortParameters.vb
1 Module ModuleComPortParameters
2     'Global variables for the communication parameters
3     Public port As String
4     Public baud_rate As Integer
5     Public parity As IO.Ports.Parity
6     Public configuration_is_valid As Boolean
7 End Module
8
```



CAPÍTULO

5

PLCs

JPSantos
2023

5. Relembrar a programação de PLCs (Fatek)

O que é um PLC e para que serve?

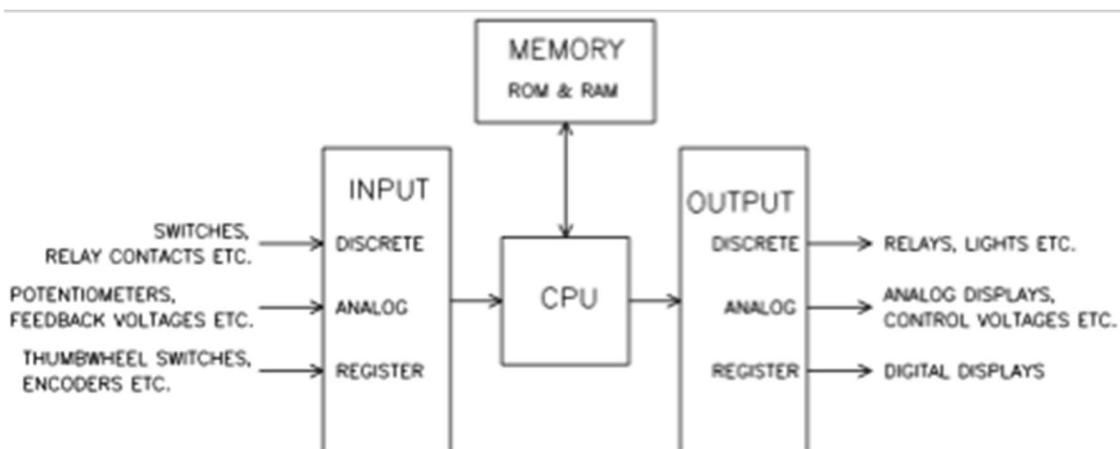
Qual a sua organização interna e os seus bornes externos ?

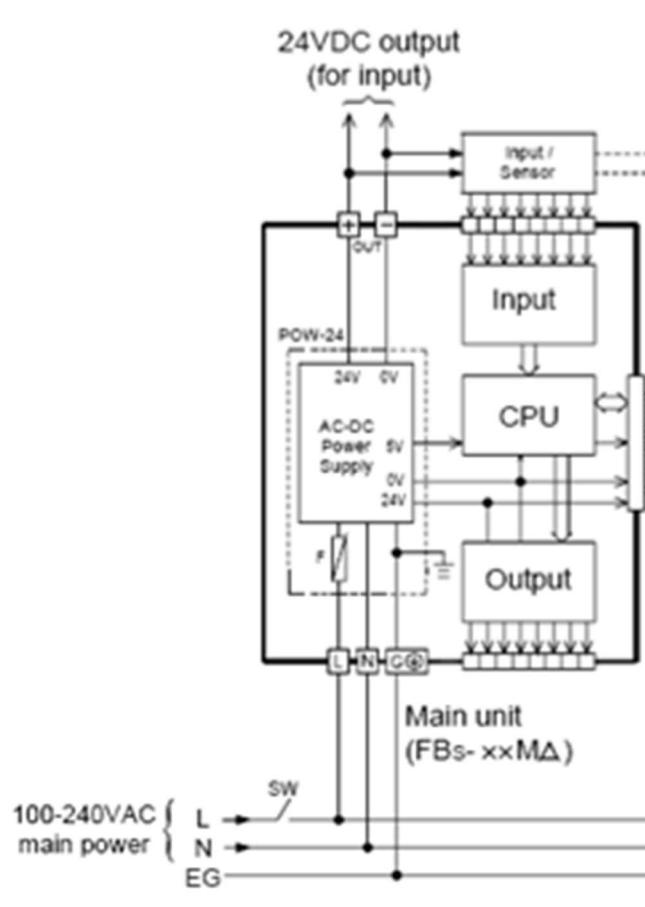
O que é a Linguagem Ladder ?

O que é um Ambiente de Desenvolvimento Integrado para o desenvolvimento dos programas (ex. Ladder) e a sua transferência para a memória interna do PLC ?

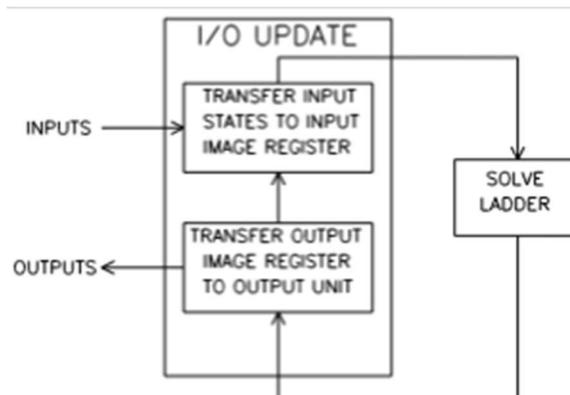
5.1. Arquitectura do PLC

- Entradas
- Memória
- CPU
- Saídas
- Alimentação





Ciclo automático



5.2. Bornes do PLC

O PLC Fatek disponível no Departamento, é o modelo FBs-20MC. Este modelo é alimentado a 230 volt a partir da rede elétrica, possui uma fonte de alimentação interna que gera 24Volt o que nos permite ativar as 12 entradas digitais (X -INPUT) e alimentar as 8 saídas digitais (Y-OUTPUT) que possui.

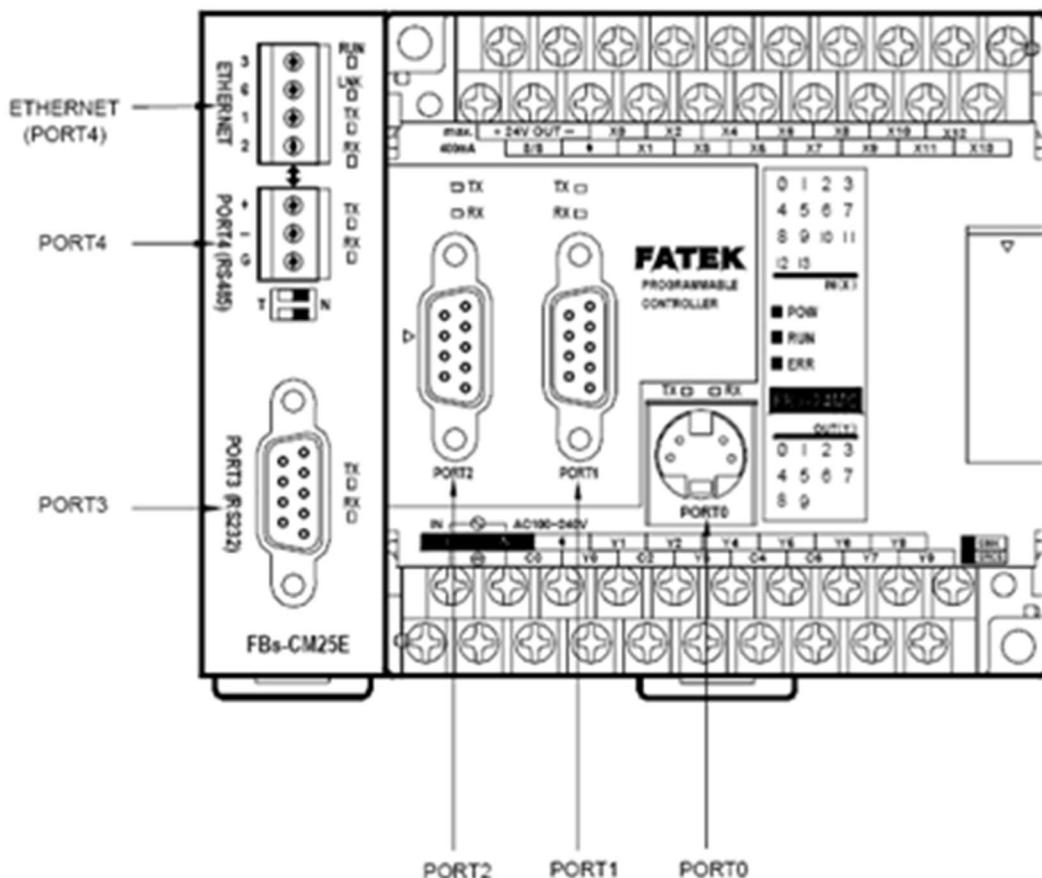


Figura: PLC – FATEK (<http://www.fatek.com>)

Manual do hardware:

Digital input http://www.esea.cz/support/fatek/FBe_Manual/Hardware/Chapter_5.pdf

Digital Output http://www.esea.cz/support/fatek/FBe_Manual/Hardware/Chapter_6.pdf

Excerto do Manual de Hardware da Fatek (pág. H1-2) Existe ainda um módulo adicional de comunicação com três interfaces de comunicação: Rs232, Rs488 e Ethernet (pág. H1-4)

Módulo Ethernet, Rs232, rs485:

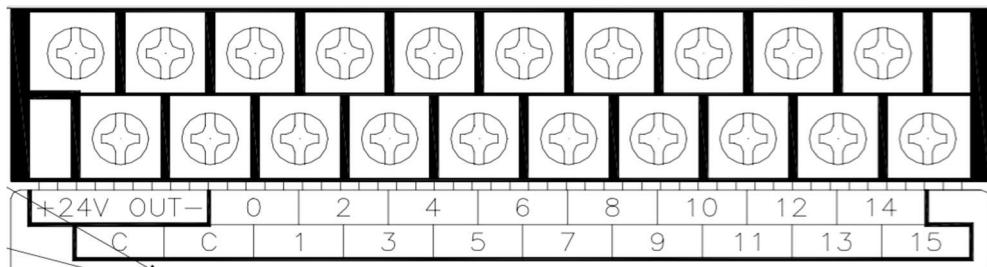
http://file.fatek.com/en/PLC/FBs/FBs_CM25E/FBs_CM25E_Datasheet_en.pdf

5.2.1. Entradas digitais

Em algumas marcas de autómatos, as entradas digitais têm a designação “X”.

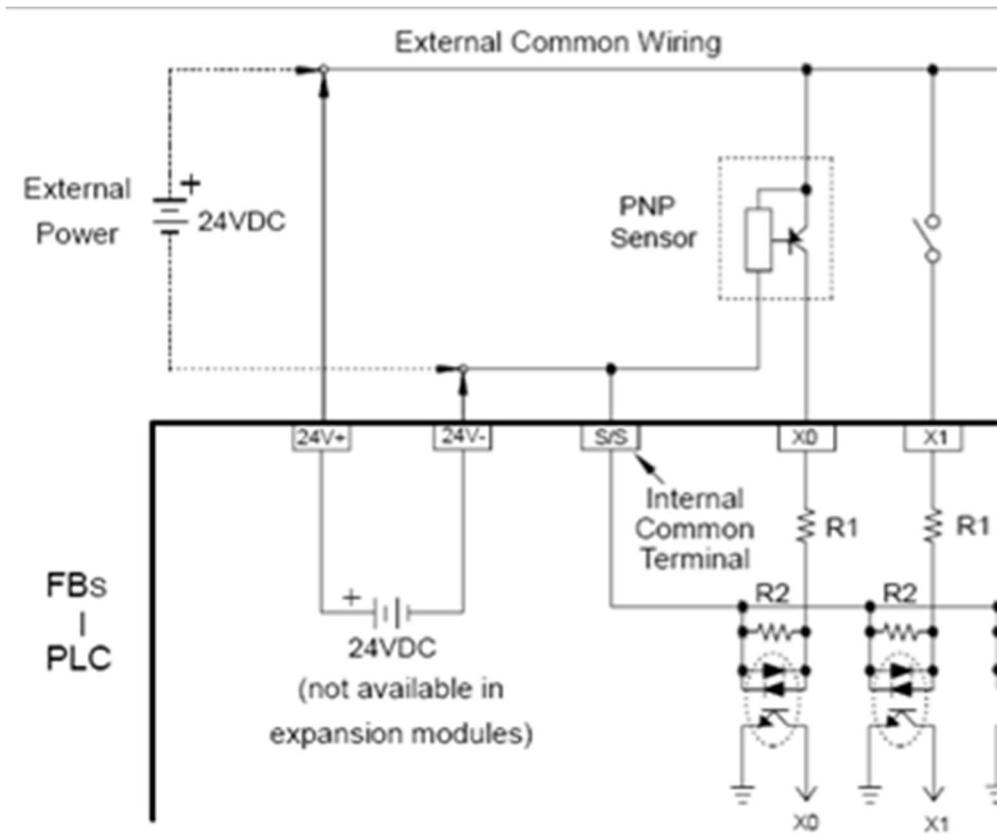
Por exemplo, 8 entradas digitais: X0 a X7, ou 16 entradas digitais: X0 a X15 (figura seguinte).

Na figura seguinte podemos também observar que a cada entrada digital deste modelo, corresponde um foto transístor interno que será ativado quando aplicarmos 24 v ao borne externo correspondente. Se esse foto transístor for ativado, o programa Ladder assume que essa entrada digital está ativa.



Analizando a figura seguinte, se aplicarmos 24v no borne X0 e 0v no borne S/S, cria-se uma diferença de potencial entre os bornes X0 e S/S, e uma intensidade de corrente através das resistências R1 e R2, e também através do foto diodo do foto transístor.

De facto, tanto podemos aplicar os +24v no borne X0 e 0 V no borne S/S, como o seu contrário, aplicar 24v no borne S/S e 0V no borne X0. Nos dois casos há uma intensidade de corrente num dos foto diodos do foto transístor, ativando essa entrada digital.



5.2.2. Saídas digitais

Em algumas marcas de autómatos, as saídas digitais têm a designação “Y”.

Por exemplo, 7 saídas digitais: Y0 a Y7, ou 12 saídas digitais: Y0 a Y11 (figura seguinte).

Na figura seguinte podemos também observar que a cada saída digital deste modelo corresponde a um Relé interno (R) que é controlado pelo programa Ladder. Quando o programa Ladder ativa uma saída digital, por exemplo Y2, corresponde a activar o relé interno Y2, estabelecendo o contacto elétrico entre o borne externo C2 e o borne externo Y2. Isto pode ser comprovado experimentalmente ligando um multímetro, no modo de “continuidade”, entre os bornes C2 e Y2. Ou seja, se alimentarmos o borne C2 com 24V, esses 24 volt “aparecerão” no borne Y2 quando o programa Ladder activar a saída Y2.

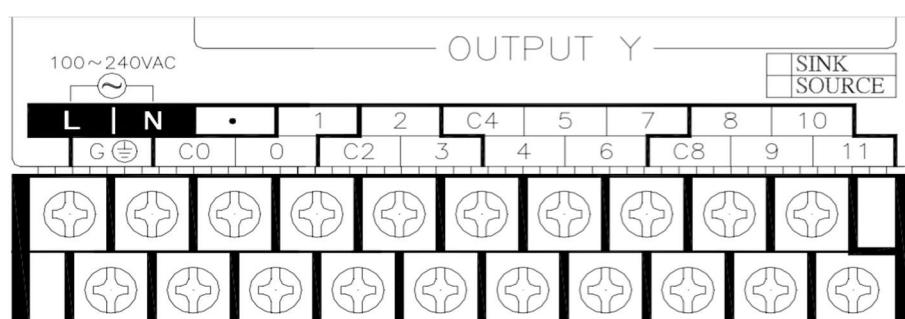
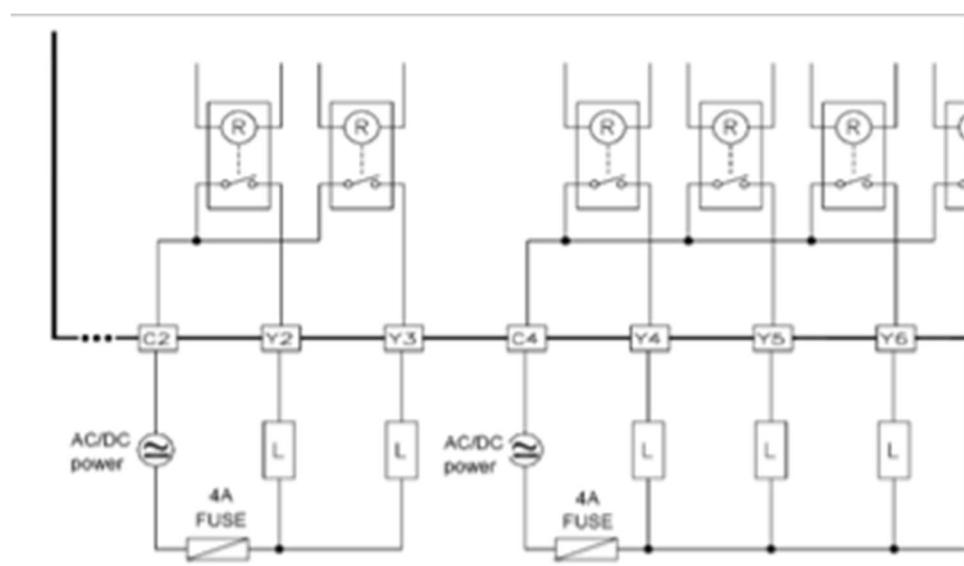
Podemos alimentar os bornes comuns C0, C2, C4,... com tensões até 230 v.

Na figura seguinte podemos observar que:

- o borne comum C2 pode alimentar as saídas Y2 e Y3,
- o borne comum C4 pode alimentar as saídas Y4, Y5, Y6 e Y7,
- o borne comum C8 pode alimentar as saídas Y8, Y9, Y10 e Y11,

Não se vê na figura, mas o borne C0 pode alimentar as saídas Y0 e Y1

Pergunta (“tipo questionário”): se ligar aparelhagem de som (o amplificador) ao borne C2, e as colunas de música ao borne Y2, será possível ouvir música?



5.2.3. Saídas e Entradas analógicas



FBs-B2A1D

Analog I/O Combo Expansion Board



Introduction

FBs-B2A1D is one of the analog I/O boards of FATEK FBs series PLC. For analog output it provides 1 channel of 12-bit (coded in 14 bits) output signal. For safety, the output signal will be automatically forced to zero(0V or 0mA) when the module is not serviced by CPU for 0.5 second. For analog input it provides 2 channels of A/D inputs with 12-bit (coded in 14 bits) resolution. The Input/Output signal types (voltage or current) can be selected by the field wiring.

Pin Assignment

Analogue Input	Analogue Output
VIO	VIO
IIO	VOO
VI1	IIO
II1	
Gnd	Gnd

Specification

Analog Input

Total Channels : 2 Channels

Resolution : 12 bits

Coding Format : 14 bits (0 ~ 16380)

Signal Resolution : 2.44mV(Voltage), 4.88uA(Current)

Registers Occupied : 2 Registers (D4072 · D4073)

Conversion Time : Updated each scan

Accuracy : ±1 %

Max. Absolute Input Rating :

±15V(Voltage), 30mA(Current)

Input Impedance : 100KΩ(Voltage), 125Ω(Current)

Measurement Range : 0 ~ 10V(Voltage)

0 ~ 20mA(Current)

Analog Output

Total Channels : 1 Channel

Resolution : 12 bits

Coding Format : 14 bits (0 ~ 16380)

Signal Resolution : 2.44mV(Voltage), 4.88uA(Current)

Register Occupied : 1 Register (D4076)

Conversion Time : Updated each scan

Accuracy : ±1 %

Max. and Min. output loading :

Voltage Output : 2K ~ 1MΩ

Current Output : 0 ~ 500Ω

Output Range : 0 ~ 10V (Voltage)

0 ~ +20mA (Current)

Common Specification

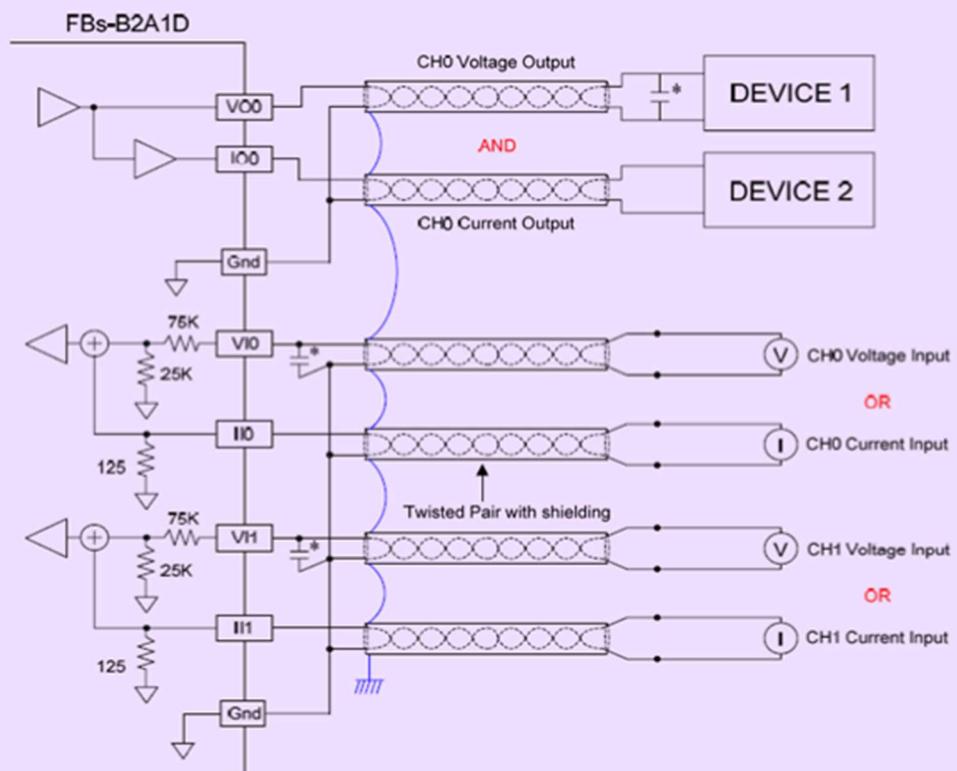
Indicator(s) : No

Internal Power Consumption : 5V, 100mA (Max. Load)

Operating Temperature : 0 ~ 60 °C

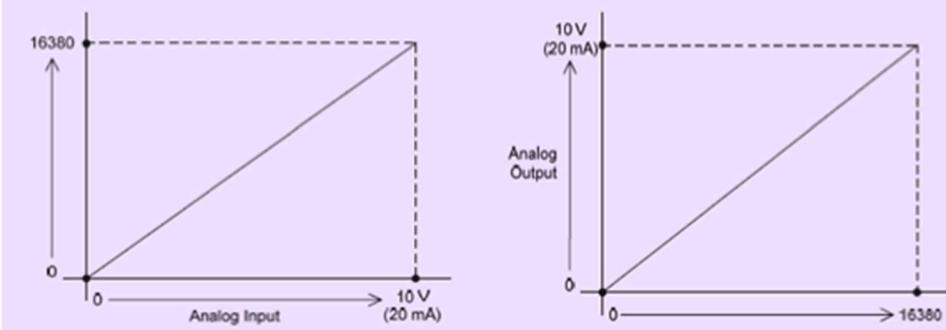
Storage Temperature : -20 ~ 80 °C

Wiring Diagram



* 0.1~0.47 uF capacitor (to filter noise)Advice to Install, but not necessary!!

Characteristics Charts



Registers allocation Map

Channel No.		Mapped Register
Analog Input	CH0	D4072 (0 ~ 16380)
	CH1	D4073 (0 ~ 16380)
Analog Output	CH0	D4076 (0 ~ 16380)

5.3. Memória Interna

http://www.esea.cz/support/fatek/FBs_Manual/Manual_1/instruction/Chapter_2.pdf

Typeee	Symbol	Item		Range	Remarks	
Digital < Bit Status >	X	Digital Input (DI)		X0~X255 (256)	Mapping to external digital I/O	
	Y	Digital Output (DO)		Y0~Y255 (256)		
	TR	Temporary Relay		TR0~TR39 (40)	For branched points	
	M	Internal Relays	Non-Retentive	M0~M799 (800)* M1400~M1911 (512)	M0~M1399 configurable as Non-retentive or Retentive, M1400~M1911 are fixed to Non-retentive	
			Retentive	M800~M1399 (600)*		
	Special Relay			M1912~M2001 (90)		
	S	Step Relays	Non-Retentive	S0~S499 (500)*	S20~S499 configurable as Retentive S500~S999 configurable as Non-retentive	
			Retentive	S500~S999 (500)*		
	T	Timer contact status		T0~T255 (256)		
	C	Counter contact status		C0~C255 (256)		
Register < Word Data >	TMR	CV of Timer Register	0.01STime Base	T0~T49 (50)*	The quantity of each time base can be configured	
			0.1S Time Base	T50~T199 (150)*		
			1S Time Base	T200~T255 (56)*		
	CTR	CV of Counter Register	16-bit Retentive	C0~C139 (140)*	Configurable as Non-retentive	
			16-bit Non-Retentive	C140~C199 (60)*	Configurable as Retentive	
			32-bit Retentive	C200~C239 (40)*	Configurable as Non-retentive	
			32-bit Non-Retentive	C240~C255 (16)	Configurable as Retentive	
	DR or HR	Data Registers	Retentive	R0~R2999 (3000)* D0~D3999 (4000)	R0~R3839 configurable as Non-retentive or Retentive, D0~D3999 are fixed to Retentive	
			Non-Retentive	R3000~R3839 (840)*		
	IR	Input Registers		R3840~R3903 (64)	Map to external AI Register input	
	OR	Output Registers		R3904~R3967 (64)	Map to external AO /Register output	
	Special Register	System Special Registers				
		High-Spped Timer Register				
		HSC Registers	Hardware (4sets)			
			Software(4sets)			
		Calendar Registers	Minute	Second		
			Day	Hour		
			Year	Month		
				Week		
	DR or ROR	Data Registers			As general purpose registers if ROR not been configured.	
		Read Only Registers			Configurable as ROR for recipe like application	
	FR	File Registers		F0~F8191(8192)	Need dedicated instruction to access	
	XR	Index Registers		V,Z (2) · P0~P9 (10)		

5.4. Editor Ladder (WinProladder)

O programa “WinProLadder” permite editar no computador programas para o PLC (em Ladder) e transferi-los para o PLC. Permite também monitorizar no computador, em tempo real, o funcionamento do programa autómato enquanto é executado pelo PLC.

Software (WinProladder):

https://elearning.ua.pt/pluginfile.php/1556974/mod_folder/content/0/WProlad328-22108-ENU.exe?forcedownload=1

Manual do WinProladder:

https://elearning.ua.pt/pluginfile.php/1556974/mod_folder/content/0/AulaT12_Winproladder_manual.pdf?forcedownload=1

Manual do Simulador

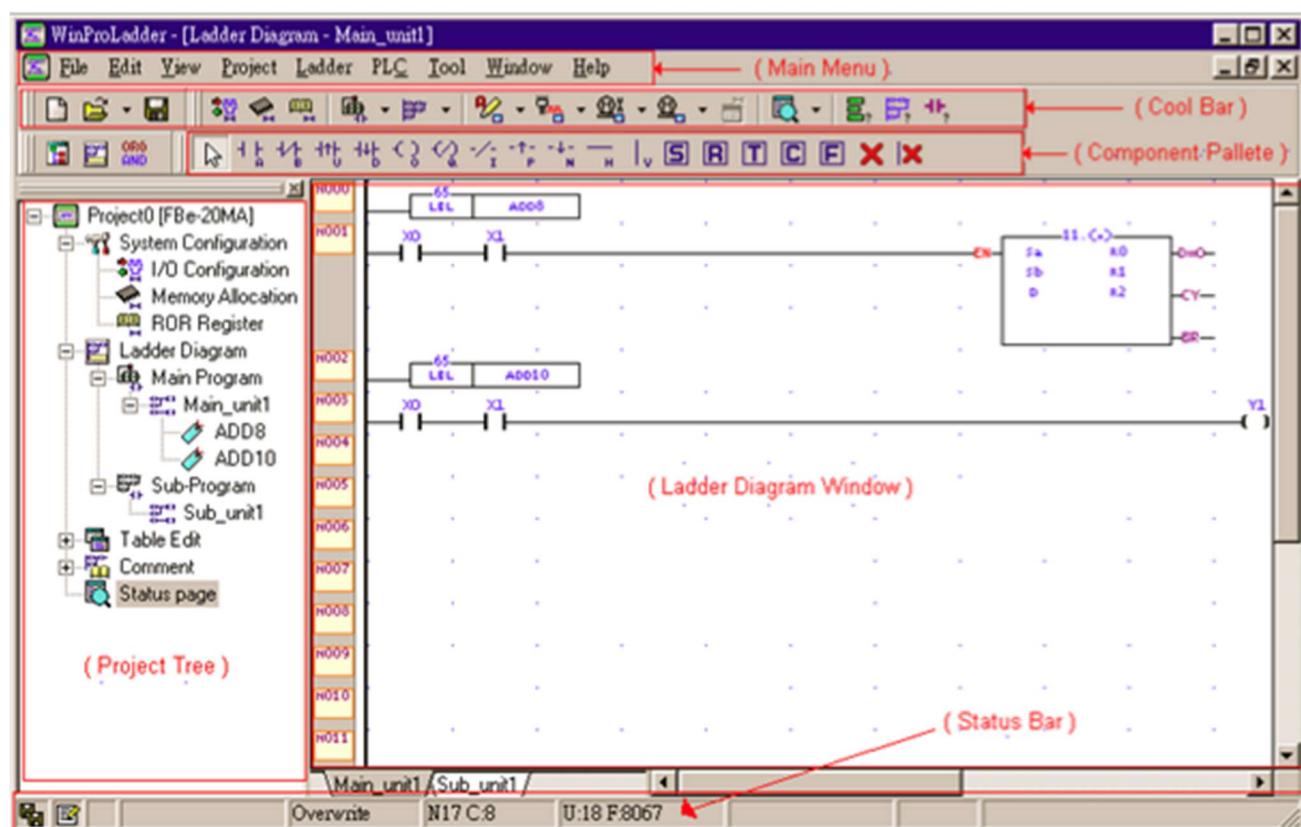
https://elearning.ua.pt/pluginfile.php/1556974/mod_folder/content/0/AulaT12_WinProladder_simulation_enu.pdf?forcedownload=1

A figura seguinte apresenta as várias janelas do ambiente de desenvolvimento dos programas Ladder para os PLCs da Fatek:

“Main Menu”, em cima, com atalhos para criar, abrir, ou gravar programas Ladder.

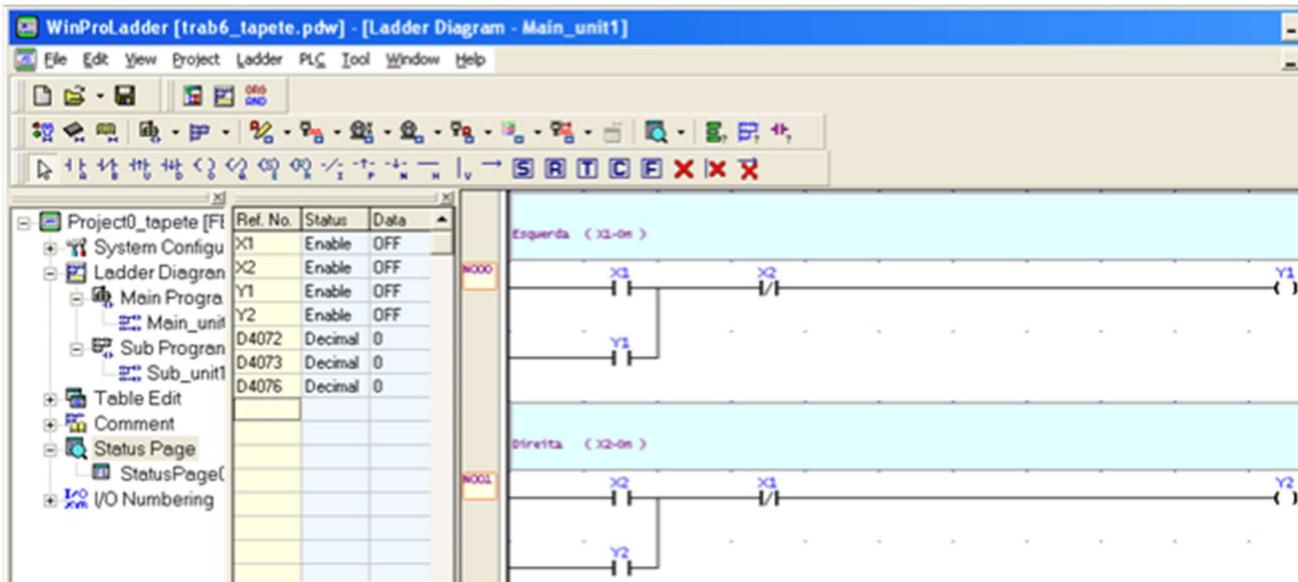
“Component Pallet”, com os símbolos da linguagem ladder para inserir/criar o programa ladder

“Project tree”, com os programas que compõem este projecto (Main Program, Sub-program), Status Page, .. “Ladder Diagram window”, para editar/criar o programa Ladder



5.4.1. Um exemplo de um programa autómato (Ladder)

A figura seguinte apresenta a interface do Winproladder, a janela de projeto à esquerda, a StatusPage ao centro, e o programa Ladder à direita.



1- Criar um novo projeto File- New Project - Edit - FBs-20-MC

A sequência apresentada a cima, significa que deve aceder ao menu “File”, selecionar a opção “New Projec”, deve selecionar o botão “Edit”, e depois escolher o modelo de PLC que temos no laboratório “FBs” com “20” input/output do tipo “MC”

2- Ativar a tabela “StatusPage” que permite monitorizar ou escrever: nas posições de memória do PLC, nas suas entradas X e saídas Y digitais do PLC, a partir do Computador , on-line.

Na janela de “ProjectTree” deve “clicar” com o rato na palavra “StatusPage”, depois voltar a clicar na palavra “StatusPage” com o botão direito do rato, dar um nome à nova tabela e fazer ok.

3- Usar uma linha “Network” para criar uma linha do programa Ladder, na janela da direita.

Cada “network” tem um número N000, N001, ...

Cada “network” tem várias colunas, repare no pontilhado, em cada célula/coluna pode inserir um símbolo Ladder.

4- Insira na “Network N0001” um “contacto normalmente aberto X1”, um “contacto normalmente fechado X2”, uma saída digital /coil Y1.

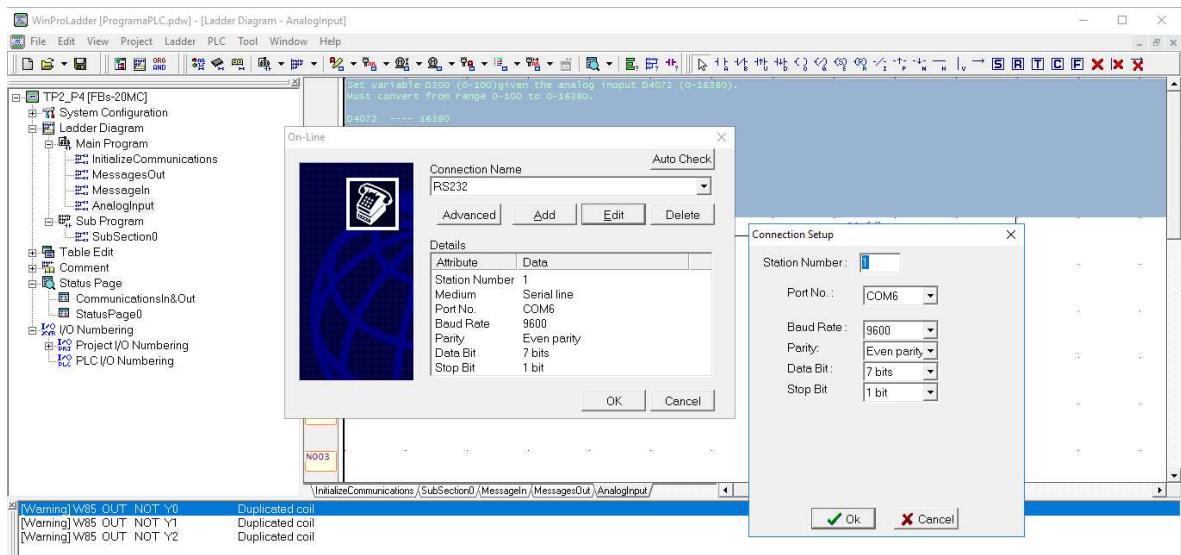
5- Insira na network N0001 um “contacto normalmente aberto Y1” por baixo do contacto X1. Para isso ser possível tem de inserir primeiro uma “linha vertical” por baixo do contacto “X1”.

6- Para apagar uma linha ou um componente deve selecionar o “X” vermelho adequado na “Component Pallet”, e depois selecionar no objecto que se pretende eliminar.

7- Inserir o comentário numa “Network”. Selecione uma network clicando com o rato no seu número, ex N0001. Depois com o botão direito do rato pode selecionar “Network Comment” ou “Program Comment”.

8- Para aumentar a largura de uma Network, o seu número de colunas, selecione com o rato uma das colunas dessa network, depois com o botão direito do rato pode, entre outras coisas, selecionar “Network Edit”, podendo expandir para 21 colunas ou comprimir para 11 colunas.

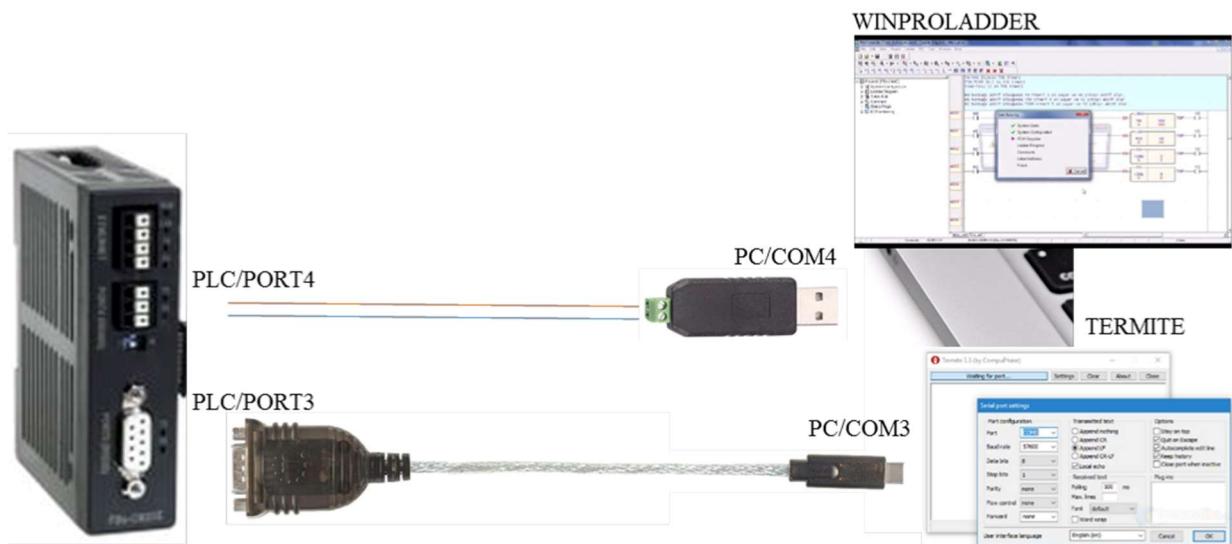
Depois deve-se dizer ao WinProLadder quais são estas portas de comunicação, quando se escolhe a opção para estar “online” com o PLC.



Uma das portas série do PLC vai ser utilizada pelo WinProLadder para comunicar, transferir o programa Ladder para o PLC, fazer debug ,etc. Esta pode ser a PORT4 do PLC (na figura de cima).

A outra porta, a PORT3 do PLC (na figura de cima), pode ser utilizada pelo programa criado por nós no PLC para enviar dados para um computador, onde o “Termite” ou outro programa de computador pode receber a informação enviada pelo PLC .

Neste caso, vamos usar dois conversores para ligar o computador/USB ao PLC/Rs232/Rs485



Após ligar os conversores anteriores a duas portas USB do computador, passamos a ter duas portas série no computador que podemos usar para ligar ao PLC, ou a outros equipamentos industriais.

Para verificar se os drivers estão a funcionar corretamente devem-se verificar as portas que estão a ser utilizadas usando o “[device manager](#)”.

Se, depois de ligar um conversor destes, na secção das “[Ports \(COM&LPT\)](#)” aparecer um símbolo amarelo quer dizer que o conversor não ficou bem instalado.

Se os drivers tiverem sido bem instalados, o “Device Manager” deve ter o aspecto da imagem da direita. Na imagem da direita, aparece :

“[ATEN USB to Serial Bridge \(COM9\)](#)”

“[Prolific USB-to-Serial Port \(COM3\)](#)”

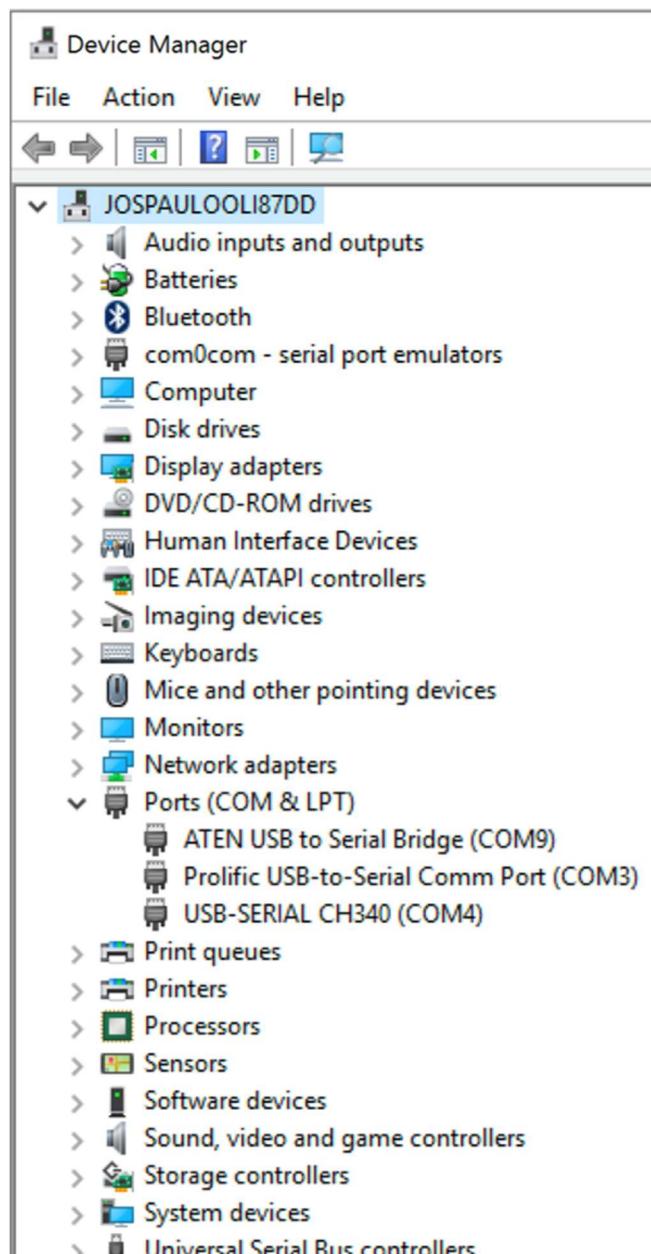
“[USB-SERIAL CH340 \(COM4\)](#)”

Isto significa que o nosso computador passou a dispor de:

-uma porta Rs232, com o nome “COM9”

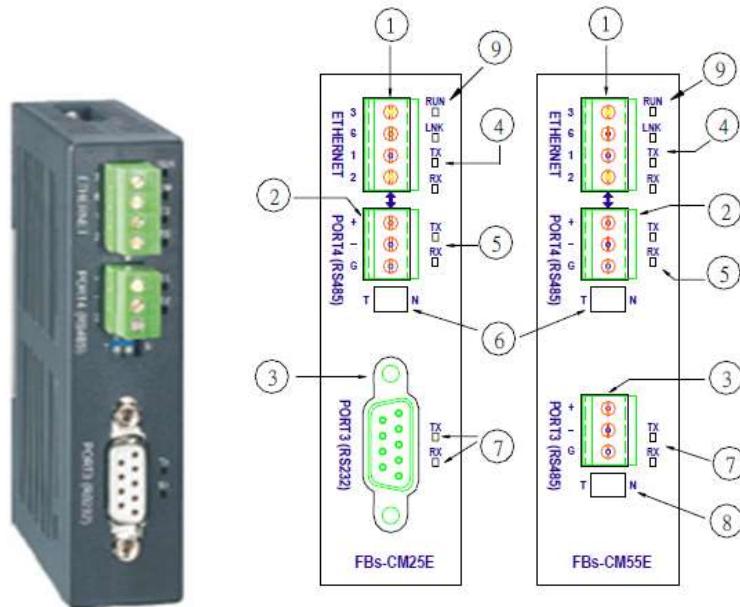
-outra porta série Rs232 com o nome “COM3”

-e uma terceira porta série , desta feita RS485, com o nome "COM4"



5.5. Módulo de comunicação Rs232, Rs485 e Ethernet da Fatek

O PLC tem uma placa/módulo adicional, com duas portas/fichas série e uma Ethernet.



5.5.1. Configuração do baudrate , bit da dados, bit paridade, stopbits .

Para configurar a porta série Rs232 ou Rs485 deste PLC, respetivamente as suas portas Port3 e Port4, é suficiente escrever o número certo nas suas posição de memória R4043 ou R4044 respetivamente. São posições de memória de 16 bits cada, em Hexadecimal 4 dígitos.

Para configurar a porta Rs232 , deve fazer
Para configurar a porta Rs485 , deve fazer

R4043 = 5622 hexadecimal = 22050 decimal
R4044 = 5622 hexadecimal = 22050 decimal

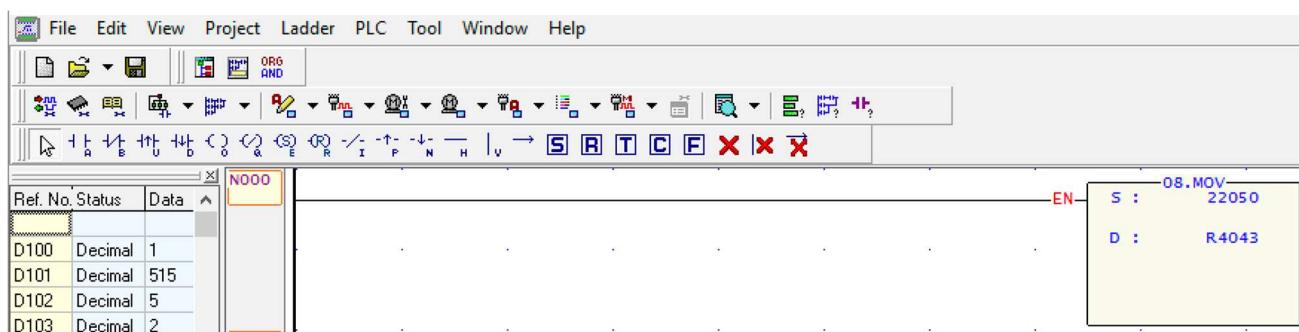
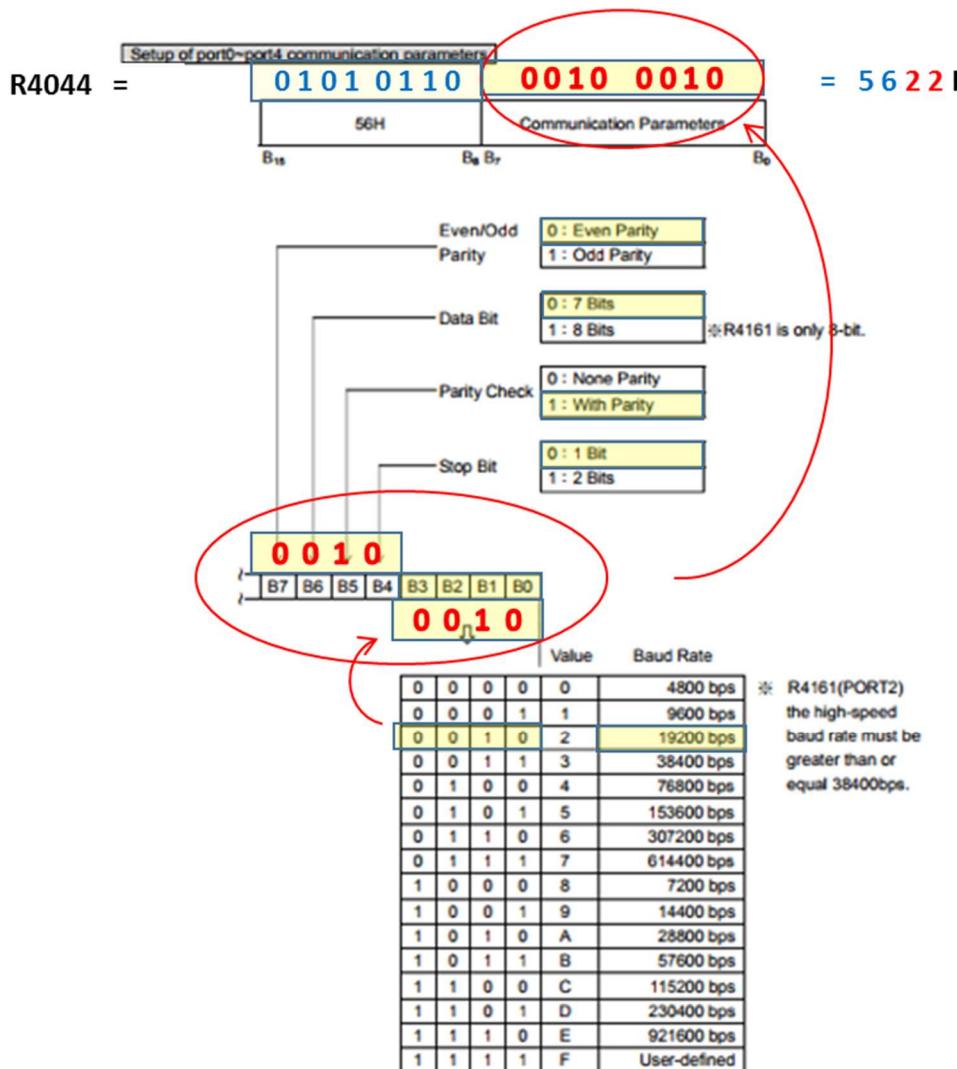


Fig. Configuração da porta Port3 (Rs232), em Ladder

Os dois dígitos mais significativos (da direita) são fixos e valem sempre 56H (o H significa Hexadecimal) 56H = 0101 0110 binário
Ou seja, R4043 e R4044 serão sempre igual a 56xx (em Hex) = 0101 0110 xxxx xxxx (em binário)

Os **xx** deste número hexadecimal, ou seja os 8 bits menos significativos (os “**x**” da direita), podem assumir vários valores, de acordo com o Baudrate, bits de dados, paridade, stop bits pretendidos.

Exemplo, para configurar a porta 4 do PLC com um baudrate de 19200 bit/s, 7 bits de dados, paridade par (even) e um stop bit, deve fazer R4044 = **5622 H**



Exercício: conf. a porta 3 do PLC
9600 7 bits de dados, 2 stop bits, odd

R4043 = 56xxH

B7 B6 B5 B4	B3 B2 B1 B0
1 0 1 1	0 0 0 1
B	1

R4043 = 56B1H

5.5.2. Criação da mensagem e envio, usando a função CLINK

Neste exemplo pretende-se enviar a mensagem “0123” para o computador. A mensagem a enviar é previamente guardada nas posições de memória D100 e posições consecutivas do PLC. Só depois a função CLINK é chamada com os parâmetros:

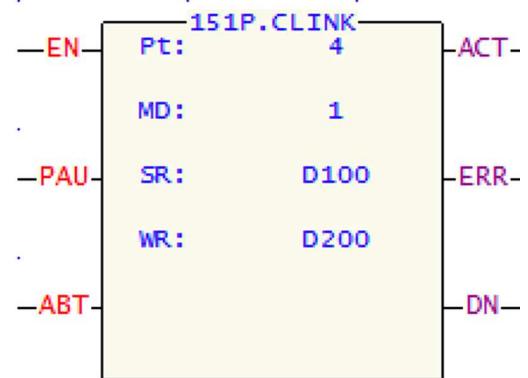
Pt: 4 controla a port4.
 MD: 1 modo de envio, Active ASCII data tx mode
 SR: D100 a mensagem a enviar começa em D100
 WR: D200 a msg enviada pelo PC será guardada em D200 e posições consecutivas.

Mensagem a enviar:

```
D100=1
D101=515 // 515decimal=0203hexadecimal
D102=6 // nº de bytes que CLINK envia
D102=02 // 02hexadecimal = 0000 0010 binário
D103='0' // o carácter '0' = 48 dec = 0011 0000
D104='1' // o carácter '1' = 49 dec = 0011 0001
D105='2' // o carácter '2' = 50 dec = 0011 0010
D106='3' // o carácter '3' = 51 dec = 0011 0011
D107=03 // 03hexadecimal = 0000 0011 binário
```

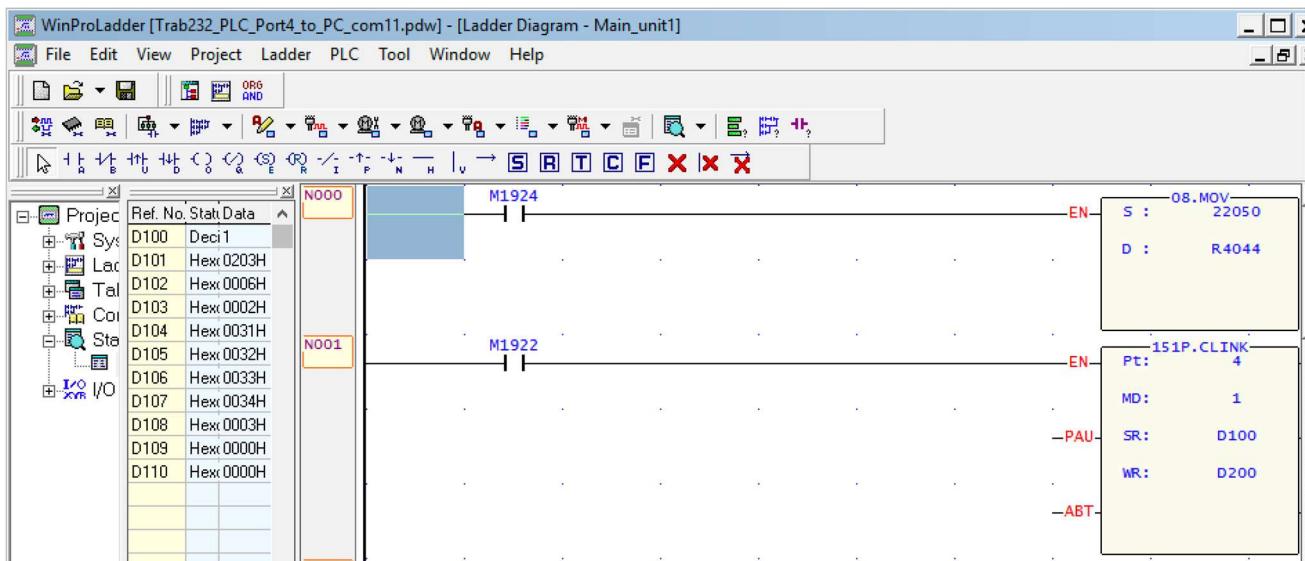
São enviados para o PC apenas os 6 bytes:

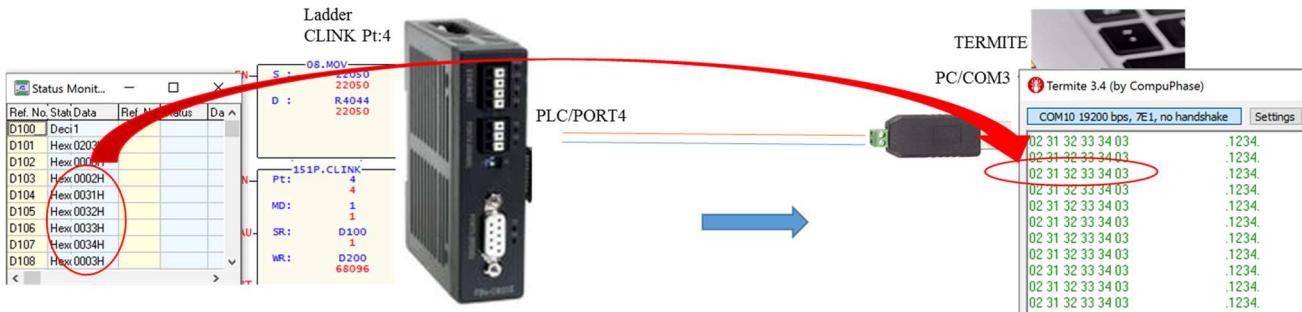
02 48 49 50 51 03



SR : Starting register of data transmission table

SR+0	Transmit only or Transmit then Receive	• Low byte is valid, 0: transmit only, no response from the slave 1: transmit then receive the responding message.
SR+1	Starting & Ending code for receiving	• High byte : Start of text for receiving Low byte : End of text for receiving
SR+2	Length of Transmission	• The maximum length of data to be transmitted is 511
SR+3	Data 1	• Low byte is valid
SR+4	Data 2	• Low byte is valid
SR+5	Data 3	• Low byte is valid
SR+6	Data 4	• Low byte is valid
•		
•		
	Data N	• Low byte is valid





5.5.3. Mensagem recebida no PC/Termite (enviada pelo PLC)

O programa WinProladder deve estar ativo no computador:
para transferir o programa autómato (Ladder) para o PLC,
para fazer “PLC-Run”,
e para monitorizar o PLC através da porta série do computador “COM3” e da PORT4 do PLC (figura seguinte).

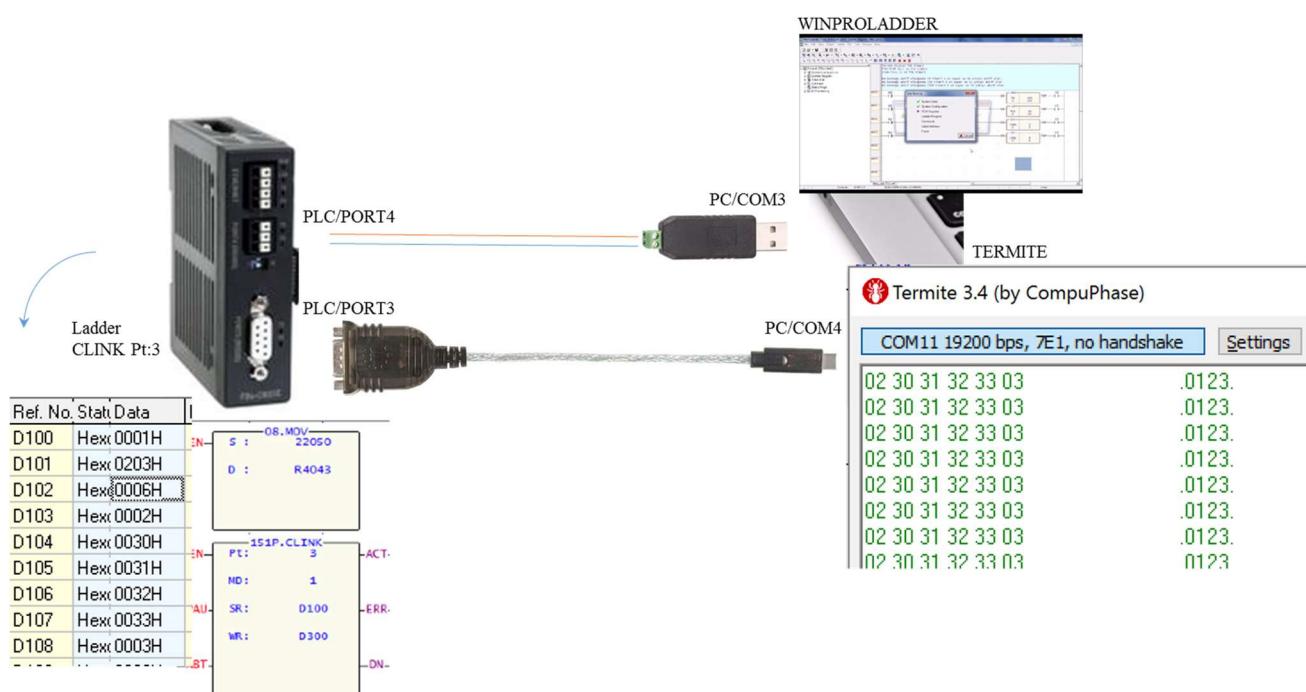
O programa “Termite” deve estar também ativo no computador, para receber na porta série “COM4” do computador as mensagens que vierem do PLC/PORT3.

Necessitamos por isso de dois cabos série para ligar o computador ao PLC:

- um dos cabos é utilizado pelo “WinProladder/COM3” para comunicar com o PLC/PORT4,
- o outro cabo série é utilizado pelo “Termite/COM4” para receber os dados enviados pelo PLC através da sua porta PORT3, (PORT3 do PLC)

Dessa forma temos duas ligações/cabos entre o PLC e o PC:

- PLC/PORT4 -----cabo1----- PC/COM3 (WinProladder)
- (o nosso programa Ladder - CLINK pt:3) PLC/PORT3 -----cabo2----- PC/COM4 (Termite)



5.5.4. Recepção e tratamento da mensagem recebida pelo PLC

Neste exemplo pretende-se receber a mensagem “56” vinda do computador. A mensagem recebida é guardada nas posições de memória D200 e posições consecutivas do PLC. De facto, a função CLINK só começa a guardar os bytes recebidos em D209 e posições consecutivas.

Apesar do PC neste exemplo querer enviar apenas dois caracteres ‘5’ e ‘6’, o computador tem de enviar os bytes de início **02** e fim de mensagem **03** para que o PLC possa saber quando começa e termina a mensagem.

Pt:4 controla a port4

MD: 1 modo de envio, Active ASCII data tx mode

SR: D100 a mensagem a enviar começa em D100

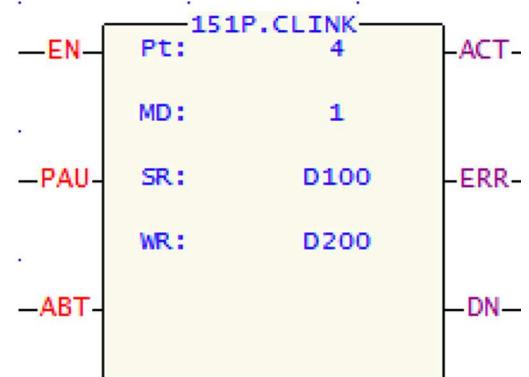
WR: D200 a msg enviada pelo PC será guardada D200...

Mensagem recebida na Port4 pela função CLINK é guardada nas posições de memória:

```
D208=4 // A função CLINK conta os bytes rx
D209=02 // 02hexadecimal = 0000 0010 binário
D210='5' // o carácter '5' = 53 dec = 0011 0101
D211='6' // o carácter '6' = 54 dec = 0011 0110
D212=03 // 03hexadecimal = 0000 0011 binário
```

São enviados para o PLC apenas os 4 bytes:

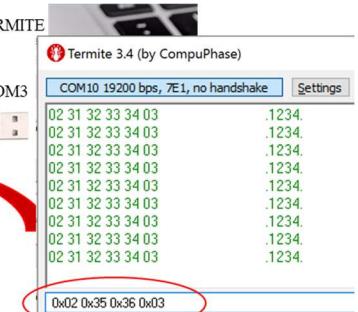
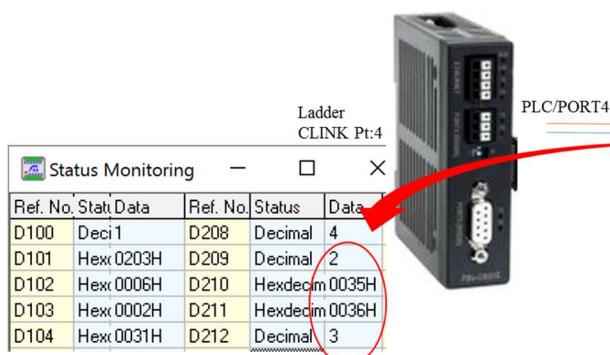
02 53 54 03



WR : Start of working register

High Byte	Low Byte	
WR+0	Result code	0
WR+1	For internal operation use	
WR+2	For internal operation use	
WR+3	For internal operation use	
WR+4	For internal operation use	
WR+5	For internal operation use	
WR+6	For internal operation use	
WR+7	For internal operation use	
WR+8	Total amount of data received	
WR+9	Data 1	
.	Data 2	
.	Data 3	
.		
.	Data N	

- Result code =0, OK ; = other values, abnormal.
- Working registers for CLINK instruction
- WR+4 : b0=1, Pending
b12 = "ACT" output indication
b13 = "ERR" output indication
b14 = "DN" output indication
- The total amount of data byte being received (the register for received data length; it includes the starting and ending code).
- The first byte of data received (if there is the starting code, it is the starting code); High byte =0.
- The second byte of data received; High byte =0.
- The third byte of data received; High byte =0.
- The N_th byte of data received (if there is the ending code, it is the ending code); High byte =0.



Bibliografia:

Ethernet Module User's Manual, informações gerais sobre a carta de expansão.

<http://www.fatek.com/en/data%2Fftp%2FPLC%2FEthernetModule%2Ffbbs-ether-enu.pdf>

Advanced Function Chapter 12 : The Communication Function of FBs-PLC, para configurar os parâmetros de comunicação RS232:

http://www.fatek.com/en/data%2Fftp%2FPLC%2FFBs_Manual%2FManual_2%2FChapter_12.pdf

Advanced Function Chapter 13 : The Applications of FBs-PLC Communication Link, para ver que registos é necessário configurar:

http://www.fatek.com/en/data%2Fftp%2FPLC%2FFBs_Manual%2FManual_2%2FChapter_13.pdf

5.6. Linguagem Ladder

Tipos de dados
Temporizadores e contadores
Move
Operações matemáticas
Conversões
Operações booleanas
Shift
Comunicações

5.6.1. Tipos de dados Fatek

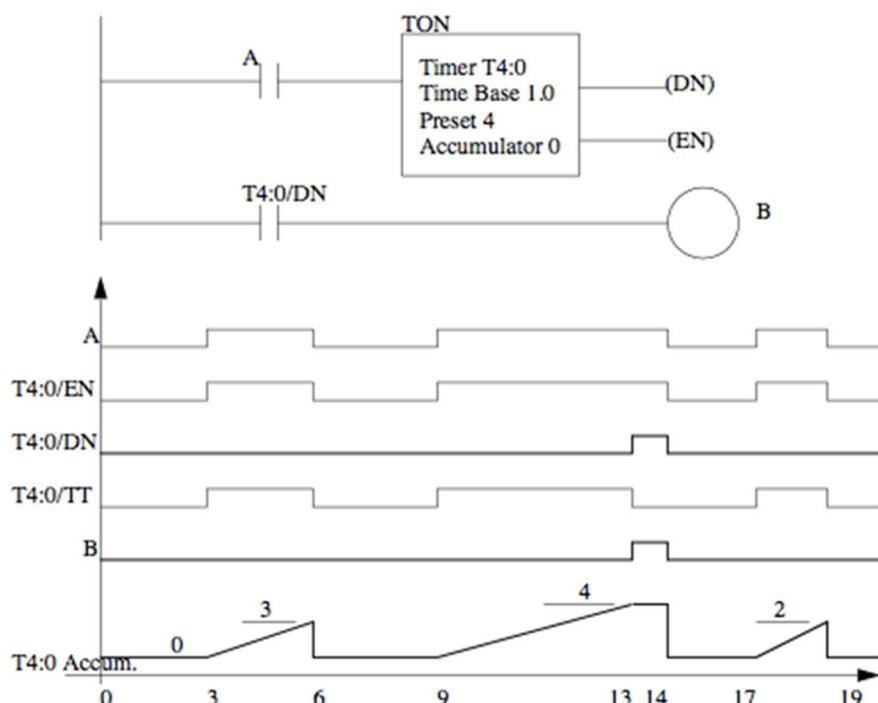
M (bit)
D (Integer de 16 bits)
R (Integer, retentivo, de 16 bits)
T (Temporizador)
C (Contador)

Name	Type	Bits	Range
BOOL	boolean	1	0 to 1
SINT	short integer	8	-128 to 127
INT	integer	16	-32768 to 32767
DINT	double integer	32	-2.1e-9 to 2.1e9
LINT	long integer	64	-9.2e19 to 9.2e19
USINT	unsigned short integer	8	0 to 255
UINT	unsigned integer	16	0 to 65536
UDINT	unsigned double integer	32	0 to 4.3e9
ULINT	unsigned long integer	64	0 to 1.8e20
REAL	real numbers	32	
LREAL	long reals	64	
TIME	duration	not fixed	not fixed
DATE	date	not fixed	not fixed
TIME_OF_DAY, TOD	time	not fixed	not fixed
DATE_AND_TIME, DT	date and time	not fixed	not fixed
STRING	string	variable	variable
BYTE	8 bits	8	NA
WORD	16 bits	16	NA
DWORD	32 bits	32	NA
LWORD	64 bits	64	NA

5.6.2. Temporizador On-delay

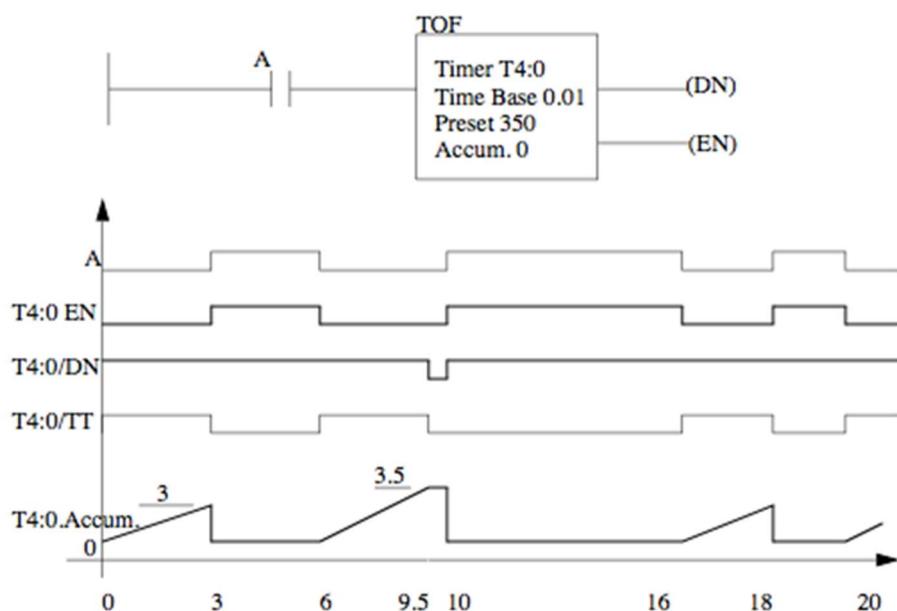
Este temporizador espera um tempo predefinido, desde que é ligado até a sua saída (DN) ser ativada. Logo que é desligado a sua saída desliga imediatamente.

Acumulador
Base de tempo
Bits: EN, DN, TT



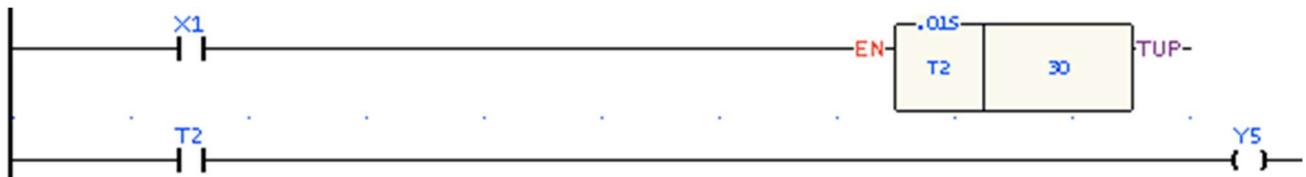
5.6.3. Temporizador Off-delay

Este temporizador liga imediatamente a sua saída (DN), mas só a desliga depois de um tempo predefinido, após a sua entrada ter sido desligada.



5.6.4. Temporizador (exemplo Fatek)

Se ativar a entrada digital X1 durante $30 \times 0,01$ segundos, o bit T2 (do Timer 2) passa a 1, e neste caso a saída digital Y5 também.



No WinProladder selecione o menu Project - Memory Allocation

Retentive Coil Totals [0 .. 1400]	600	(M800-M1399)
Retentive Step Relay Totals [0 .. 980]	500	(S500-S999)
0.01 Sec Timer Totals [0 .. 256]/[0 .. 220]	50	(T0-T49)
0.1 Sec Timer Totals [0 .. 256]/[0 .. 220]	150	(T50-T199)
1 Sec Timer Totals	56	(T200-T255)
Retentive 16 Bit Counter Totals [0 .. 200]/[0 .. 96]	140	(C0-C139)
Retentive 32 Bit Counter Totals [0 .. 56]/[0 .. 32]	40	(C200-C239)
Retentive Data Register Totals [0 .. 3840]	3000	(R0-R2999)
ROR Register Totals [0 .. 3072]	0	

No PLC existem:

600 bits retentivos (M800 a M1399) que podem ser usados no programa.

50 temporizadores (T0 a T49) com uma base de tempo de 0,01 seg.

150 temporizadores (T50 a T199) com uma base de tempo de 0,1 seg.

55 temporizadores (T200 a T255) com uma base de tempo de 1 seg.

140 contadores (C0 a C139), posições de memória de 16 bits cada.

40 contadores (C200 a C239), posições de memória de 32 bits cada.

3000 posições de memória de dados, de utilização geral, de 16 bits cada.

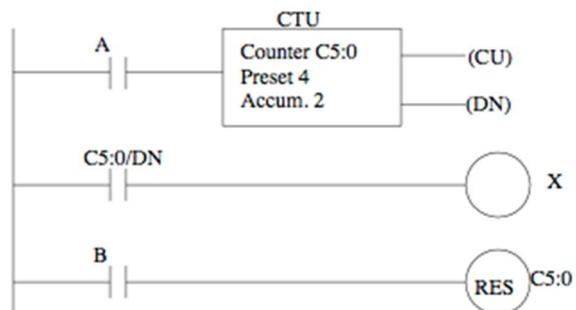
A palavra retentiva significa que essa memória, seja um Timer, um Counter, ou Data, mantém o seu valor, mesmo quando se desliga e volta a ligar o PLC .

Exercícios - Temporizador

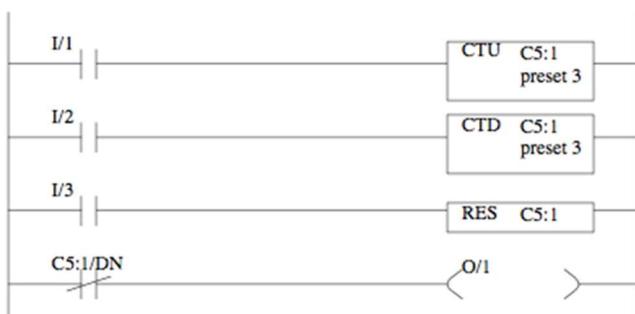
Desenvolva um programa Ladder que ligue uma saída do PLC (Y1 -luz de saída), 15 segundos após o interruptor “A” for ligado.

5.6.5. Contadores

Count-Up (CTU)
Count-Down (CTD)



Botão I1 incrementa
Botão I2 decrementa
Botão I3 limpa contador
Saída O1 é ativada quando contador é diferente de 3



5.6.6. Contadores (Exemplo Fatek)

Se o utilizador ativar 5 vezes a entrada X2 do PLC o bit C3 (do contador 3) passa a 1 e neste exemplo a saída Y6 também.



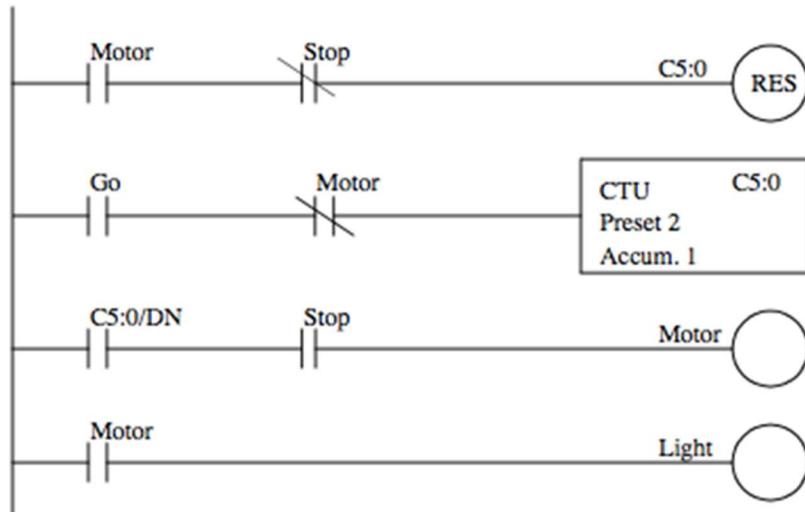
Exercício - Contadores

Desenvolva um programa Ladder para acender uma luz, depois de interruptor A tem sido fechado 10 vezes. Pressione o botão B para limpar o contador

Exercícios com temporizadores e contadores

Um motor é controlado por dois interruptores. O interruptor “Go” liga o motor, e o interruptor “Stop” irá pará-lo. Se o interruptor de paragem for usado para parar o motor, o interruptor Go deve ser pressionado duas vezes para ligar o motor. Quando o motor está ativo uma luz deve estar ligada. O interruptor “Stop” é do tipo “Normally Closed”.

Solução:

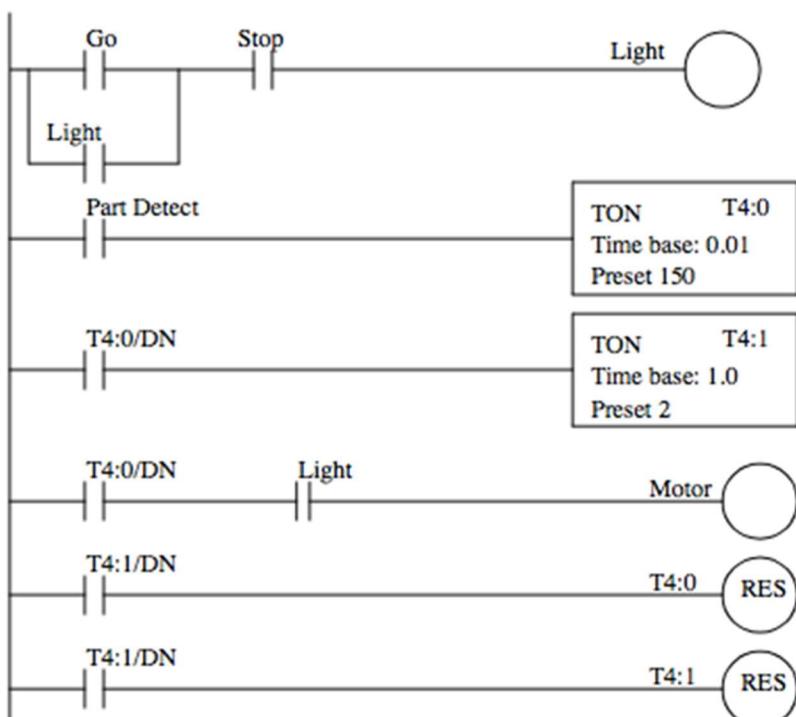


Consider:

- what will happen if stop is pushed and the motor is not running?

2. Um transportador é acionado por um motor eléctrico. Quando uma peça é detectada por um sensor óptico queremos que o tapete pare passado 1,5 segundos. Depois de estar parado durante 2 segundos deve recomeçar a transportar peças. Existe um botão de “Start”, e um botão de “Stop”. Enquanto o tapete estiver ativo, uma luz indicadora também estará ativa.

Solução:

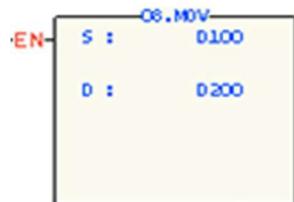


- what is assumed about part arrival and departure?

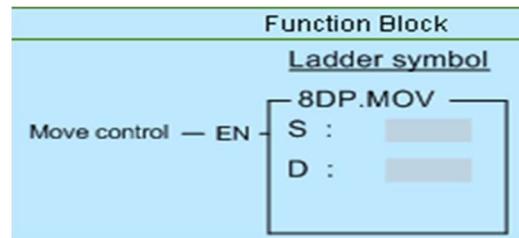
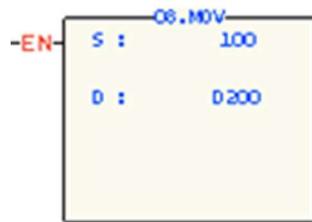
5.6.7. Atribuir valores a posições de memória interna do PLC (Mov)

MOV(valor, Destino)
MOV(Source, Destino)

D100 = D200



D200 = 100



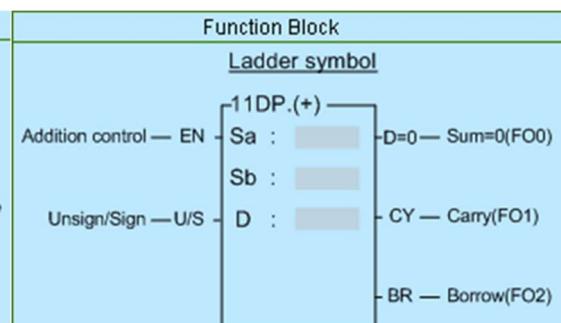
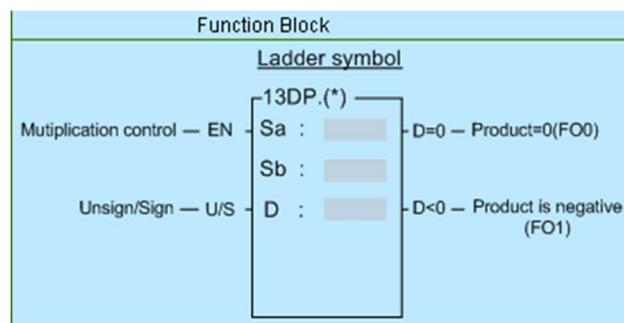
5.6.8. Operações matemáticas (+, -, /, *)

$$D = Sa * Sb$$

ex. se D100 = 10, se D200 = 25
D: D300, Sa: D100, Sb: D200
 $D300 = D100 * D100 = 250$

$$D = Sa + Sb$$

ex. se D100 = 15, se D200 = 25
D: D300, Sa: D100, Sb: D200
 $D300 = D100 + D100 = 40$

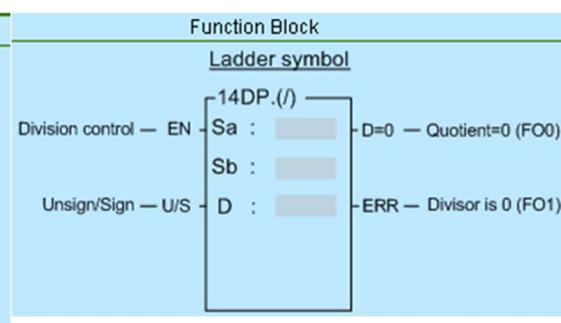
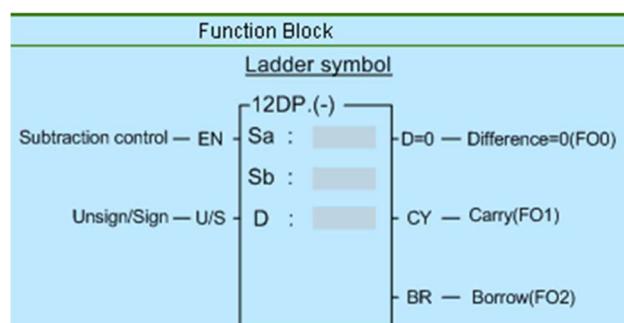


$$D = Sa - Sb$$

ex. se D100 = 15, se D200 = 25
D: D300, Sa: D100, Sb: D200
 $D300 = D100 + D100 = -25$

$$D = Sa / Sb$$

ex. se D100 = 150, se D200 = 10
D: D300, Sa: D100, Sb: D200
 $D300 = D100 / D100 = 15$

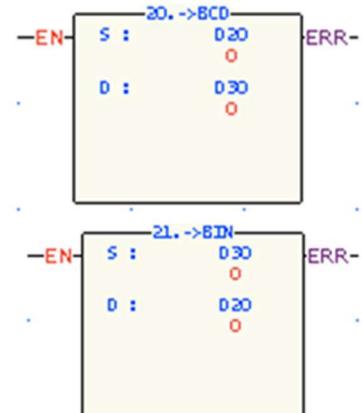


Exercício - Operações matemáticas em Ladder

Desenvolva um programa Ladder para implementar esta operação $D200 = (D100/16 + 45) * 10$

5.6.9. Conversões

Graus para radianos
Radianos para graus



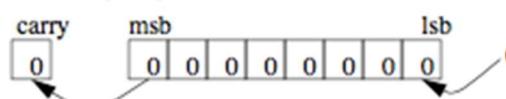
CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE
CONVERSION OF FLOATING POINT NUMBER TO INTEGER
CONVERSION OF HEXADECIMAL VALUE TO ASCII CODE
CONVERSION OF INTEGER TO FLOATING POINT NUMBER
CONVERT THE CURRENT PULSE VALUE TO DISPLAY VALUE (m)
CONVERTING THE RAW VALUE OF 4~20mA ANALOG INPUT
COS TRIGONOMETRIC INSTRUCTION
COUNTER
CRC16 CALCULATION

5.6.10. Operações deslocamento

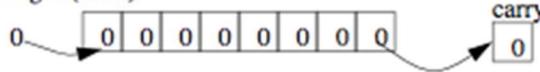
SHFL - Deslocamento de bits à esquerda,
SHFR - à direita

ROTL - Rotação de bits à esquerda
ROTR - à direita

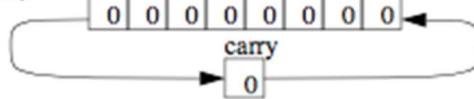
Arithmetic Shift Left (ASL)



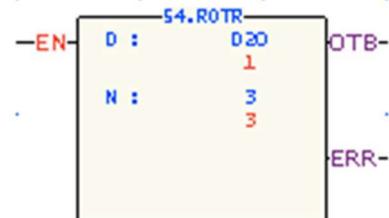
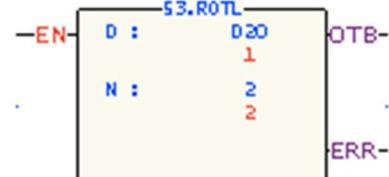
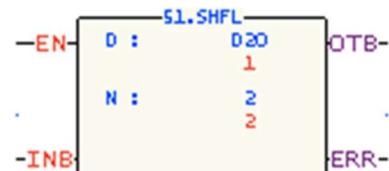
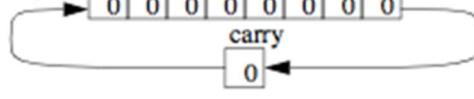
Arithmetic Shift Right (ASR)



Rotate Left (ROL)



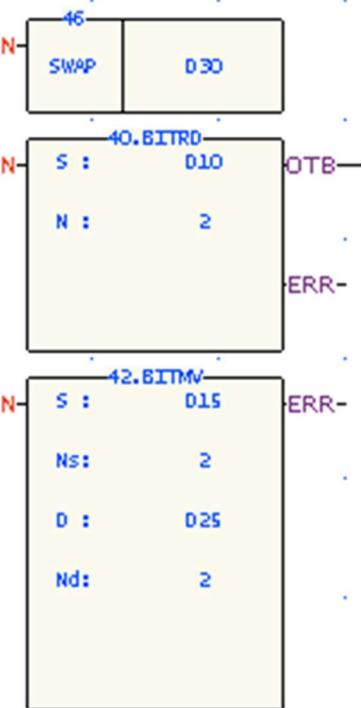
Rotate Right (ROR)



5.6.11. Operações booleanas

SWAP

Function Block	Operand
Ladder symbol <p>Swap control — EN</p>	<p>D : Register for byte data swap D may combine with V, Z, P0~P9 to serve indirect address application</p>

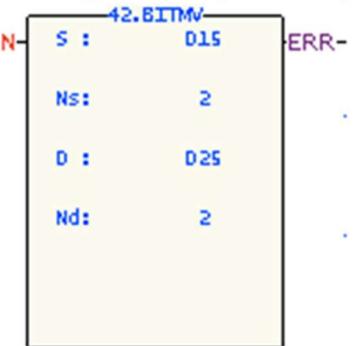


BITRD

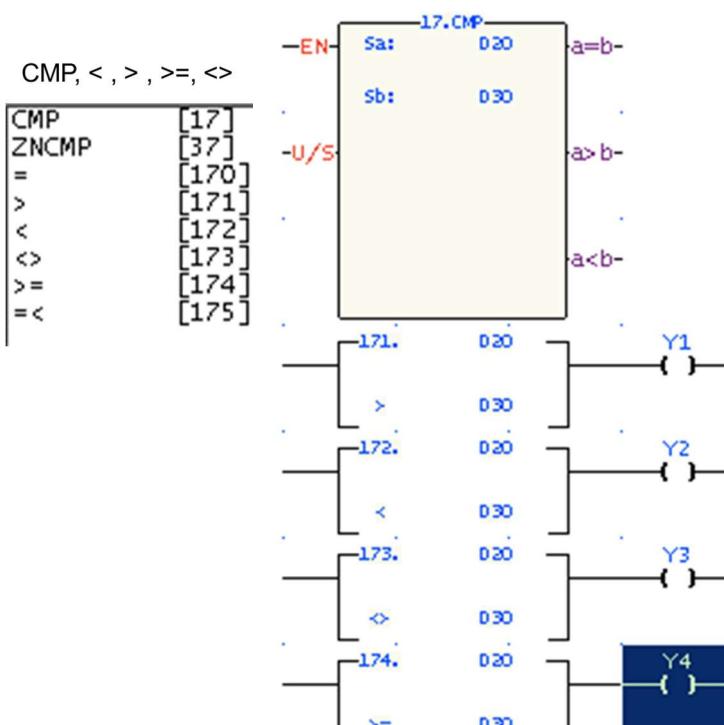
Function Block	Operand
Ladder symbol <p>Operation control — EN</p> <p>S : OBT — Output bit</p> <p>N : ERR — N value error</p>	<p>S : Source data to be read N : The bit number of the S data to be read out. S, N may combine with V, Z, P0~P9 to serve indirect address application</p>

BITWR

Function Block	Operand
Ladder symbol <p>Write control — EN</p> <p>D : ERR — N value error</p> <p>Write bit — INB</p>	<p>D : Register for bit write N : The bit number of the D register to be written. D, N may combine with V, Z, P0~P9 to serve indirect address application</p>



5.6.12. Operações Comparação



Function Block	Ladder symbol
<p>Compare control — EN</p> <p>Unsign/Sign — U/S</p>	<p>17DP.CMP</p> <p>Sa : a = b — Sa=Sb (FO0)</p> <p>Sb : a > b — Sa>Sb (FO1)</p> <p>a < b — Sa<Sb (FO2)</p>

Exercício - Operações de comparação em Ladder

Desenvolva um programa Ladder

para ligar a saída digital Y5 se D200 for superior a 15 e a saída Y6 se for inferior a 10 .



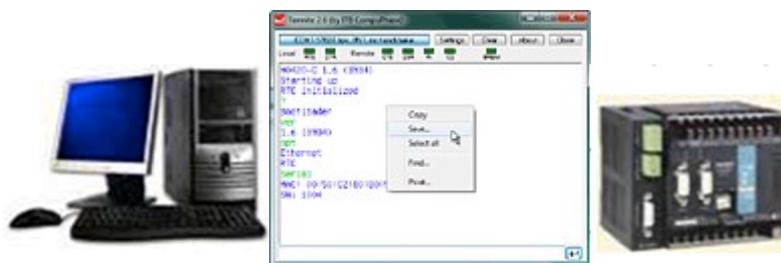
Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

Trabalho nº 4 – Programa PLC/Ladder para controlo do Reservatório

Abílio Borges , José Paulo Santos

Pretende-se criar um programa autómato em linguagem Ladder (em anexo), para configurar o módulo de comunicação RS232 dos autómatos para enviar e receber dados. O PLC utilizado será o Fatek FBs-20MC (figura em baixo).



Comunicação RS232 PLC – PC usando o software Termite

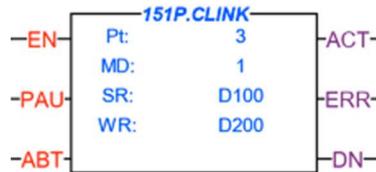
Descrição do trabalho

Usando o programa WinProladder, deve criar um programa PLC para enviar e receber mensagens de e para o PC. A ideia é fazer um código para o envio e receção de mensagens com o protocolo definido no documento de apoio para o problema de controlo de um reservatório. O presente trabalho prático tem por objectivo consolidar os conhecimentos sobre comunicações série RS232, e relembrar os conceitos gerais de programação Ladder, pretende-se criar um programa autómato em linguagem *Ladder* (em anexo), para configurar a carta RS232 dos autómatos, enviar e receber dados. O PLC utilizado será o Fatek FBs-20MC (figura em baixo).



Memória do PLC e mensagens enviadas

O programa PLC usado neste trabalho, usa as memória internas do PLC, D100 (D100 será o start register SR na figura) a D118 para criar a mensagem que será enviada para o computador. O primeiro valor a ser enviado para o computador é o conteúdo de D103. Compete à função CLINK enviar o conteúdo das memórias D103 até D118, para o computador, através da PORT3 do PLC. Assim, o programa PLC actualiza as memórias D108 a D114 com o valor de Y0,Y1,Y2,X0,X1,X2,X3.



SR : Starting register of data transmission table

SR+0	Transmit only or Transmit then Receive	<ul style="list-style-type: none">• Low byte is valid, 0: transmit only, no response from the slave 1: transmit then receive the responding message.
SR+1	Starting & Ending code for receiving	<ul style="list-style-type: none">• High byte : Start of text for receiving Low byte : End of text for receiving
SR+2	Length of Transmission	<ul style="list-style-type: none">• The maximum length of data to be transmitted is 511
SR+3	Data 1	<ul style="list-style-type: none">• Low byte is valid
SR+4	Data 2	<ul style="list-style-type: none">• Low byte is valid
SR+5	Data 3	<ul style="list-style-type: none">• Low byte is valid
SR+6	Data 4	<ul style="list-style-type: none">• Low byte is valid
•		
•		
•		
	Data N	<ul style="list-style-type: none">• Low byte is valid

Memória do PLC e mensagens recebidas

As mensagens enviadas pelo computador para o PLC são guardadas nas memórias D209 a D220 do PLC (D200 será o primeiro registo que é dado à função CLINK, WR na figura). O primeiro valor enviado pelo computador é guardado na memória D209. Compete à função CLINK guardar nas memórias D209 a D220 a mensagem que o computador enviar para a PORT3 do PLC. O programa PLC está permanentemente a ler as memórias D215 a D217 e a actualizar as saídas digitais Y0,Y1,Y2 do PLC em conformidade. Em baixo o exemplo do estado destas variáveis na memória do PLC. Em baixo a tabela

WR : Start of working register

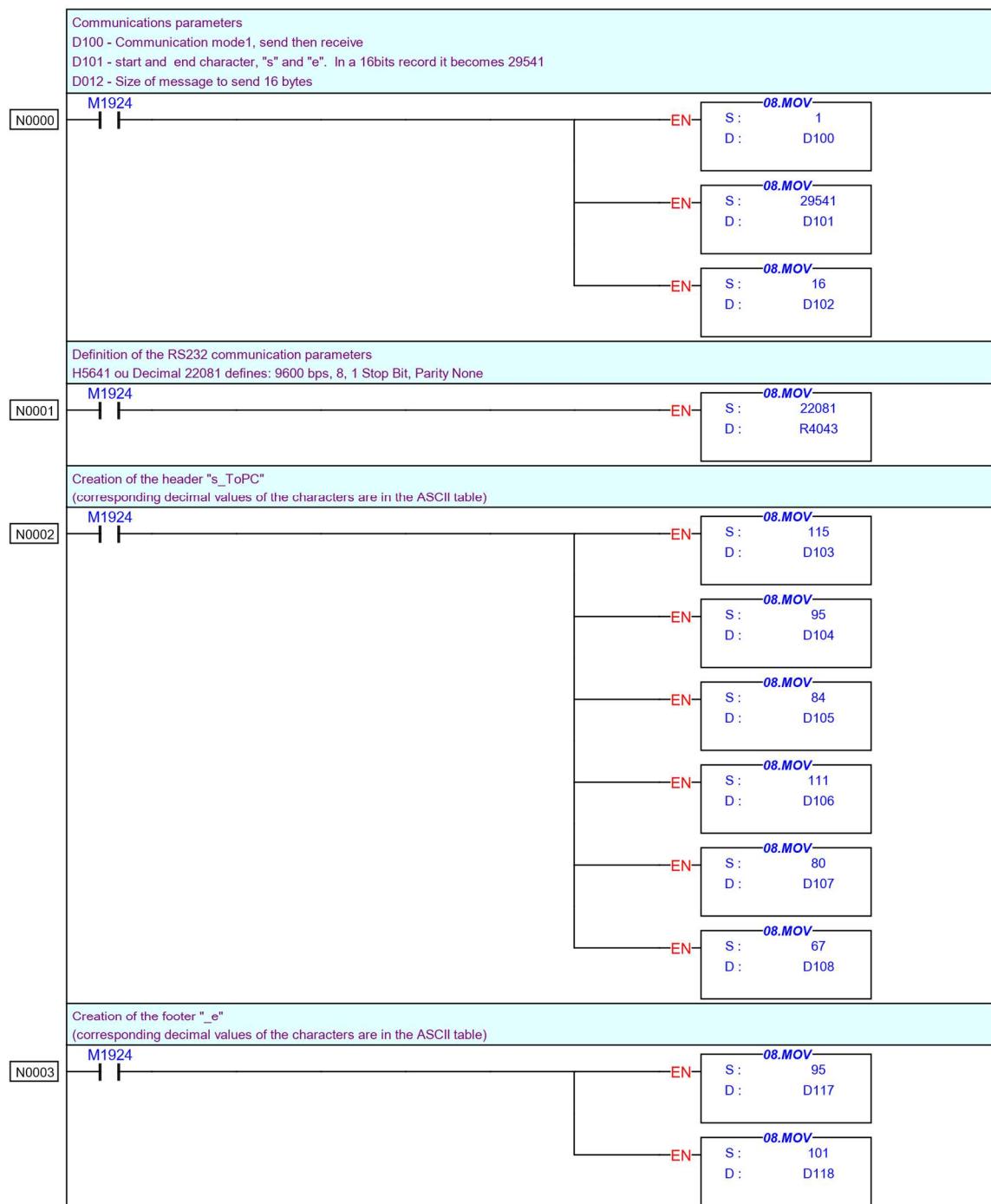
	High Byte	Low Byte
WR+0	Result code	0
WR+1	For internal operation use	
WR+2	For internal operation use	
WR+3	For internal operation use	
WR+4	For internal operation use	
WR+5	For internal operation use	
WR+6	For internal operation use	
WR+7	For internal operation use	
WR+8	Total amount of data received	
WR+9	Data 1	
•	Data 2	
•	Data 3	
•	⋮	⋮
•	Data N	

- Result code =0, OK ; = other values, abnormal.
- [Working registers for CLINK instruction](#)
- WR+4 : b0=1, Pending
b12= "ACT" output indication
b13= "ERR" output indication
b14= "DN" output indication
- The total amount of data byte being received (the register for received data length; it includes the starting and ending code).
- The first byte of data received (if there is the starting code, it is the starting code); High byte =0.
- The second byte of data received; High byte =0.
- The third byte of data received; High byte =0.
- The N_th byte of data received (if there is the ending code, it is the ending code); High byte =0.

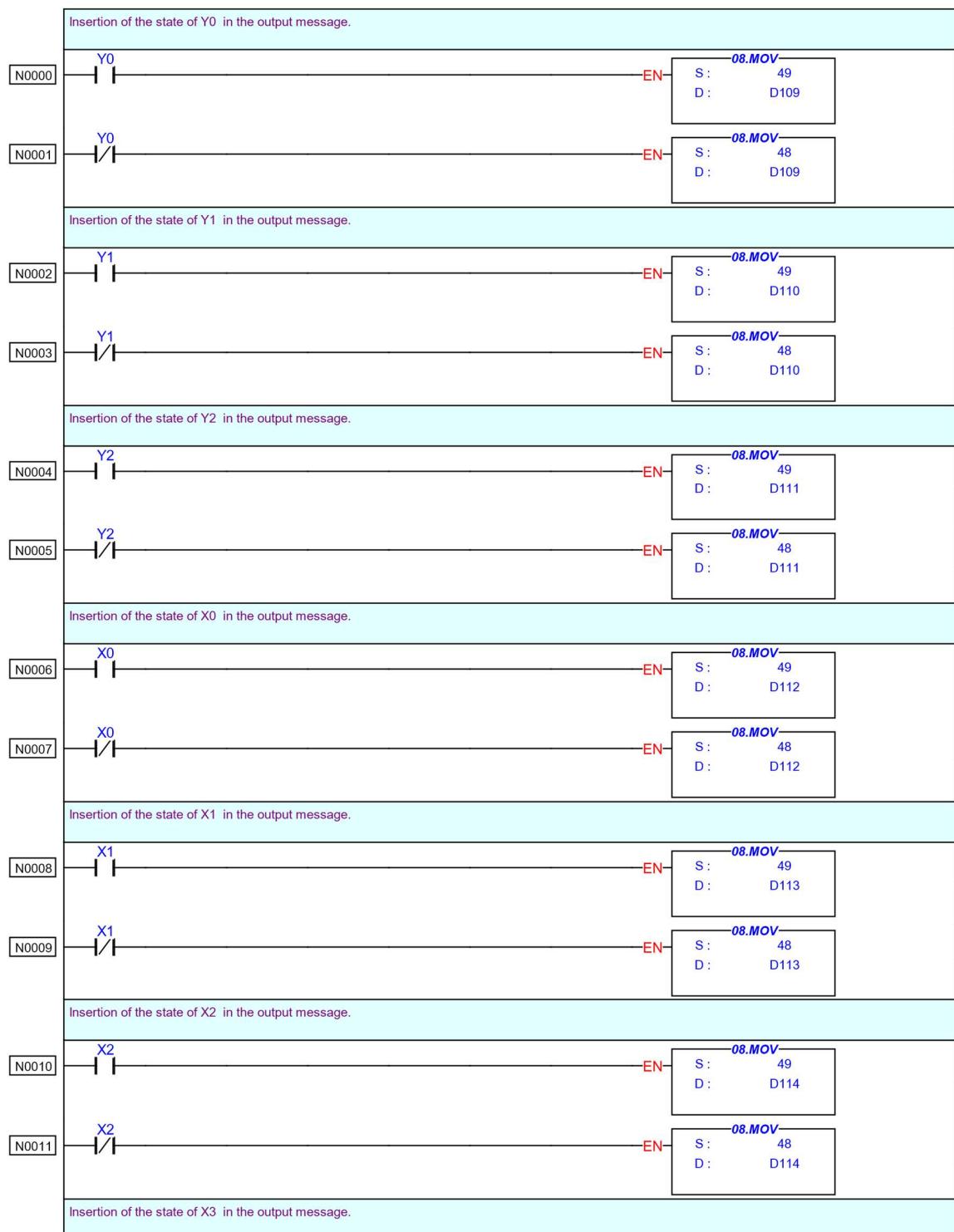
Status Monitoring									
Ref. No.	Status	Data	Ref. No.	Status	Data	Ref. No.	Status	Data	▲
D103	String	's'	D209	String	's'				▲
D104	String	'_'	D210	String	'_'				▲
D105	String	'T'	D211	String	'T'				▲
D106	String	'o'	D212	String	'o'				▲
D107	String	'P'	D213	String	'P'				▲
D108	String	'C'	D214	String	'L'				▲
D109	String	'1'	D215	String	'C'				▲
D110	String	'0'	D216	String	'1'				▲
D111	String	'0'	D217	String	'0'				▲
D112	String	'0'	D218	String	'0'				▲
D113	String	'0'	D219	String	'_'				▲
D114	String	'0'	D220	String	'e'				▲
D115	String	'0'							▼
D116	Unsigned 0								▼
D117	String	'_'							▼
D118	String	'e'							▼

Programa PLC

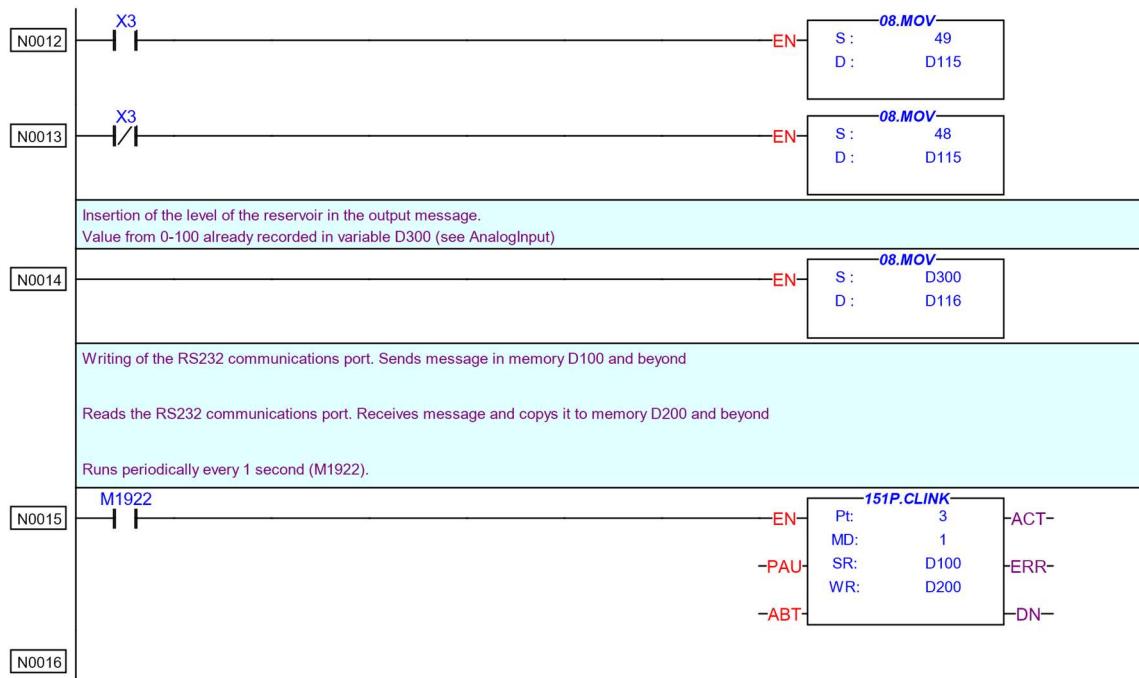
Printed Item: Ladder Diagram - InitializeCommunications



Printed Item: Ladder Diagram - MessagesOut



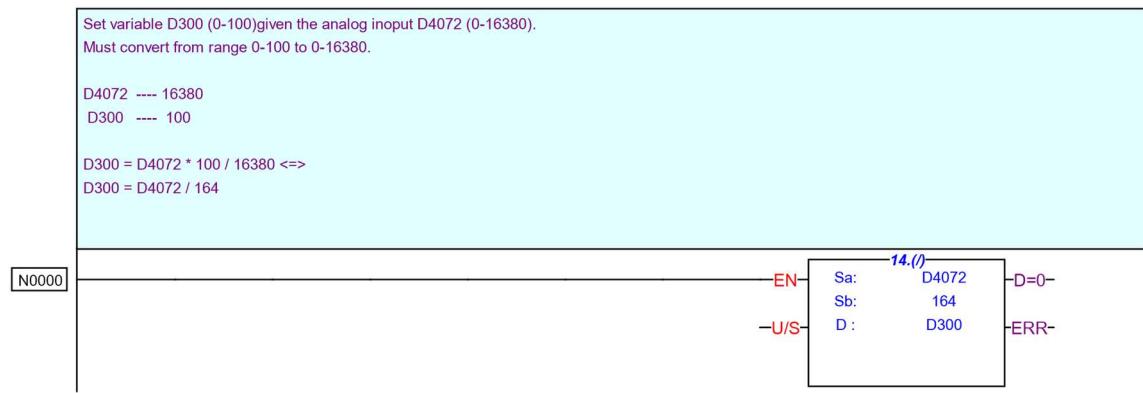
Printed Item: Ladder Diagram - MessagesOut



Printed Item: Ladder Diagram - MessageIn



Printed Item: Ladder Diagram - AnalogInput



Printed Item: Status Page - [CommunicationsIn&Out]

No.	Status	Data
D103	String	's'
D104	String	'_'
D105	String	'T'
D106	String	'o'
D107	String	'P'
D108	String	'C'
D109	String	'0'
D110	String	'0'
D111	String	'0'
D112	String	'0'
D113	String	'0'
D114	String	'0'
D209	String	's'
D210	String	'_'
D211	String	'T'
D212	String	'o'
D213	String	'P'
D214	String	'L'
D215	String	'C'
D216	String	'1'
D217	String	'0'
D218	String	'0'
D219	String	'_'
D220	String	'e'



Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

Trabalho nº 5 – Controlo e supervisão do PLC/Reservatório, a partir do PC

Introdução

Tendo em vista consolidar os conhecimentos adquiridos nas aulas teóricas sobre comunicações RS232, pretende-se adquirir, controlar e monitorizar grandezas digitais de um processo industrial. Este processo é controlado por um autómato programável equipado com cartas de entradas e saídas digitais, uma carta analógica e uma carta de comunicações RS232. Utilizando um computador com uma porta série RS232 pretende-se comunicar com o PLC para controlar o processo industrial e armazenar no disco duro do computador o valor que as grandezas digitais do PLC assumem ao longo do tempo .

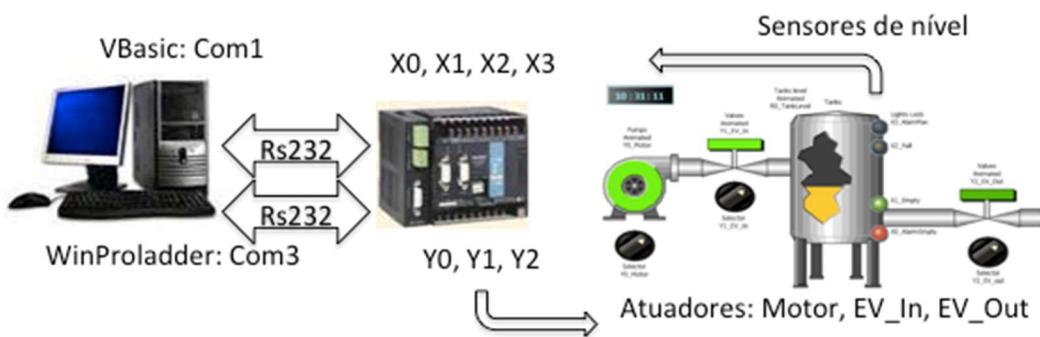


Figura : Aquisição Remota de sinais

Neste trabalho em particular, pretende-se controlar e receber no computador informações relativas a um reservatório de água (figura).

A saída digital Y0 do PLC ativa um motor que permite encher o reservatório.

A saída digital Y1 permite ativar a electroválvula de entrada de água do reservatório (EV_in) e por isso, tanto o motor como a electroválvula de entrada devem ser ativados para que o reservatório possa ser cheio. Por outras palavras, ambas as saídas digitais do PLC (Y0 e Y1) devem ser ativadas para que o reservatório possa encher.

Existe também uma electroválvula que permite a saída da água do reservatório (EV_out), ativada pela saída digital Y2 do PLC.

O Reservatório tem 4 sensores de nível que são ativados quando detectam água, nessa altura aplicam 24 volt nas entradas digitais do PLC, respectivamente nas entradas: X0, X1, X2 e X3

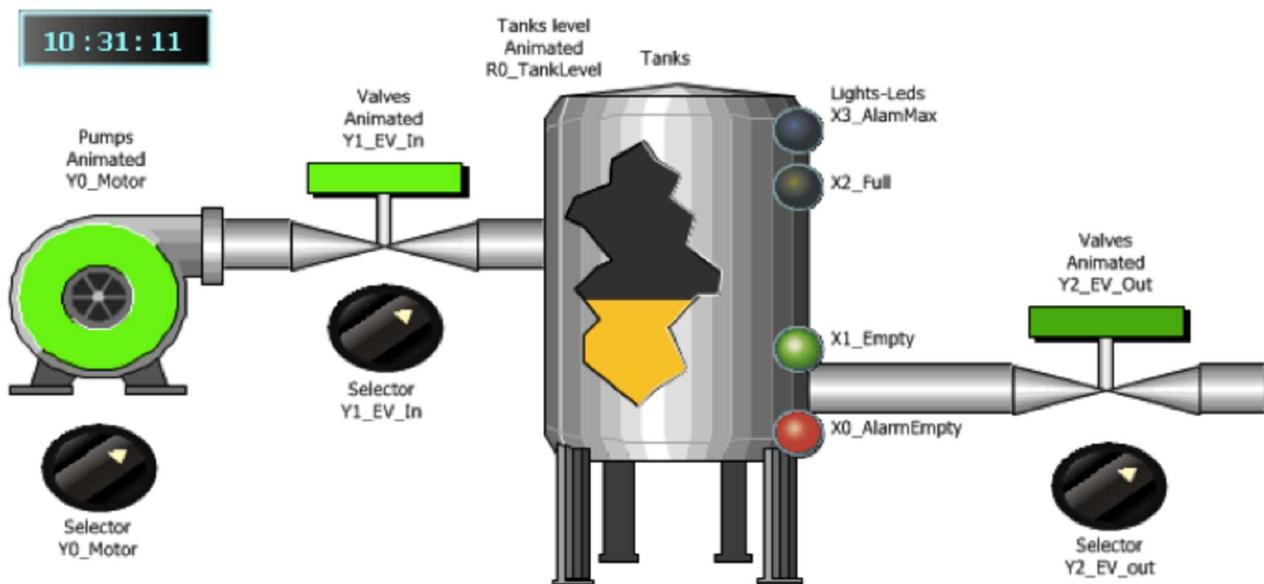
Pretende-se que o nível de água no reservatório nunca exceda o nível (X2_Full), nem seja inferior ao nível (X1_Empty).

Se por alguma anomalia, o sensor de nível “X3_AlarmMax” for ativado deve ser gerado um alarme.

Se o sensor de nível “X0_AlarmEmpty” deixar de estar ativo por o nível de água ser inferior, deve também ser gerado um alarme.

Em síntese, as entradas e saídas do PLC acima descritas são:

- Y0 - Controla o Motor (Motor)
- Y1 – Controla a ElectroValvula de entrada (EV_In)
- Y2 – Controla a ElectroVálvula de saída (EV_Out)
- X0 – Detecta a água no nível mais baixo (AlarmEmpty)
- X1 – Detecta água no nível mais baixo (Empty)
- X2 – Detecta água quando o reservatório está cheio (Full)
- X3 – Detecta água no nível de alarme mais alto do reservatório (AlarmMax)



Mensagem enviada para o PLC, via Rs232

O computador deve enviar a ordem para ligar ou desligar a(s) electroválvulas Y0, Y1, Y2 do PLC. A mensagem tem 12 bytes/caracteres. Se o programa do PLC receber a mensagem seguinte na PORT3 (ficha DB9) do PLC, desliga as saídas Y0, Y1 e Y2:

```
SerialPort1.WriteLine("TO_PLC000END")
```

Os três zeros da mensagem correspondem às três saídas digitais, respectivamente Y0, Y1 e Y2

Para ligar as três saídas digitais do PLC: Y0, Y1, Y2, o computador deve enviar para o PLC a mensagem seguinte

```
TO_PLC111END
```

Mensagem vinda do PLC, via Rs232

Sempre que o PLC receber uma mensagem do computador na sua PORT3 (ficha DB9), responde e envia para o computador uma mensagem com 15 bytes/caracteres.

A mensagem contém o estado das três saídas digitais e das quatro entradas digitais do PLC.

Se o PC receber a mensagem seguinte, significa que todas as saídas e entradas digitais do PLC estão desativadas.

```
TO_PLC00000000END
```

Os 7 zeros da mensagem correspondem, por esta ordem, ao estado das três saídas digitais Y0, Y1, Y2 e ao estado das 4 entradas digitais X0, X1, X2 e X3

Objectivos

O presente trabalho prático tem por objectivo consolidar os conhecimentos sobre comunicações série RS232, e relembrar os conceitos gerais de programação Ladder, pretende-se:

- Desenvolver um programa em *VisualBasic*, que permita configurar uma porta RS232 do PC, enviar e receber dados do PLC;
- Utilizar um programa autómato em linguagem *Ladder* (em anexo), para configurar a carta RS232 dos autómatos, enviar e receber dados;
- A familiarização com os objetos de programação: SerialPort, OpenFileDialog, SaveFileDialog, MenuStrip e com as instruções de programação: MsgBox, If Then Else, Mid, Split, InStr, etc

Programa do PLC

Usando o programa WinProladder, deve editar e transferir para o PLC o programa em anexo, através da PORT4 do PLC.

Depois de transferir o programa para o autómato não se esqueça de fazer PLC-Run.

Depois de estar a ser executado, este programa usa a PORT3 do PLC (ficha DB9) para comunicar, trocar mensagens, com o VBasic

Se não tiver o WinProladder instalado no computador, poderá obtê-lo em:

<http://www.fatek.com/en/download.php?f=data/ftp/PLC/WinProladder/software/WProlad325-19327-ENU.zip>

O manual de utilização do WinProladder, está disponível em:

http://www.fatek.com/en/download.php?f=data/ftp/PLC/WinProladder/manual/Winproladder_en.zip

Com o programa PLC descrito em apêndice, o PLC só envia uma mensagem para o computador após ter recebido uma mensagem enviada pelo computador. Por essa razão, para que o PLC possa responder com o estado das entradas e saídas digitais, o programa do computador deve, de segundo a segundo, enviar uma mensagem para o PLC.

Programa do Computador

Para desenvolver o programa do computador use o VBasic, e o objecto Serialport1 para trocar mensagens com o PLC. Sugere-se que utilize e adapte o programa desenvolvido no trabalho anterior.

- a. Possuir um interface gráfico para configuração de uma comunicação RS232;
- b. Possuir um interface gráfico onde seja possível atuar as saídas Y0, Y1 e Y2 do PLC;
- c. Possuir um interface gráfico onde seja possível visualizar o estado das saídas e entradas e do nível de água transmitidos pelo PLC;
- d. Envio de mensagens para o PLC e visualização das mensagens enviadas;
- e. Receção de dados do PLC e visualização;
- f. Extração de mensagens a partir dos dados recebidos (Mid, InStr, ...);

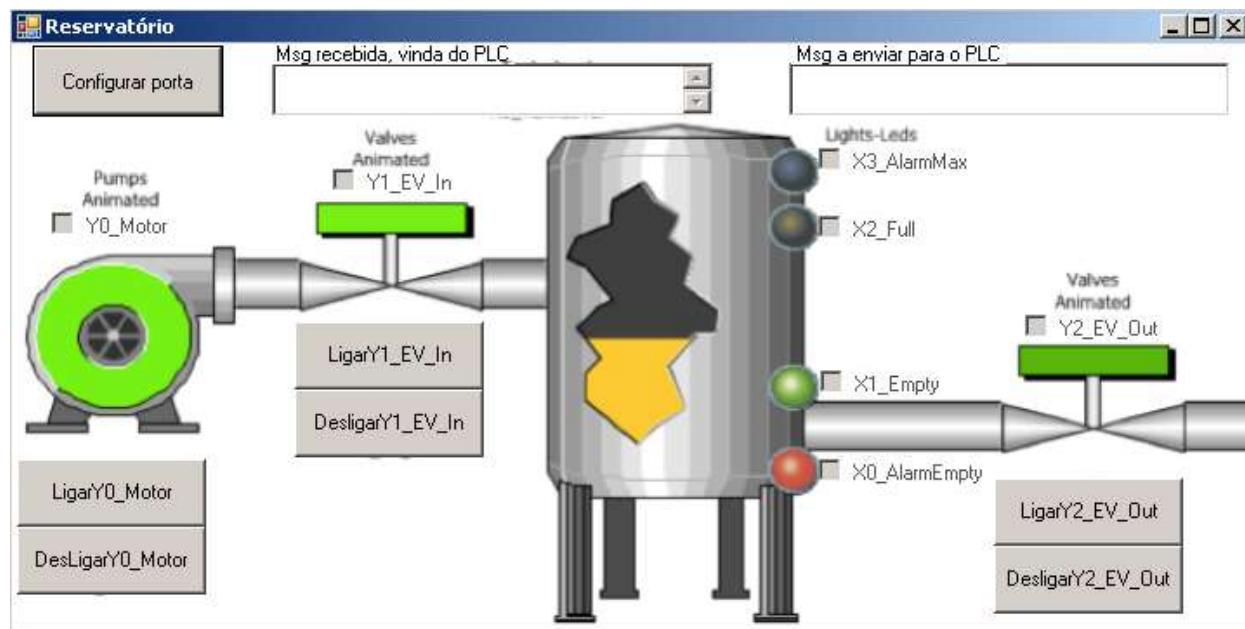
Desenvolvimento da interface

No seu Form, deverá ver o botão de configuração da porta série e as caixas de texto das mensagens enviadas e recebidas pelo computador, via Rs232, como no trabalho anterior (9600 bps, 7 Databits, Even, 1 Stop bit).

Além desses objetos, insira no Form, tal como a figura seguinte ilustra:

Sete Checkbox (Y0_Motor, Y1_EV_In, Y2_EV_Out, X0_AlarmRmpty, X1_Empty, X2_Full, X3_AlarmMax)
Seis Button (LigarY0_Motor, LigarY1_EV_In, LigarY2_EV_Out,...)

Sempre que receber uma mensagem vinda do PLC ative ou desative as 7 Checkbox, consoante o estado das três saídas digitais (Y0, Y1 e Y2) e o estado das quatro entradas digitais (X0, X1, X2, X3) presentes na mensagem recebida.



Controlo das saídas digitais do PLC

Considere o código seguinte. Quando premir o botão Y0_Motor enviará para o PLC a mensagem:

TO_PLC100END

Se enviar esta mensagem para o PLC irá ligar a saída Y0, mas também desligará as saídas Y1 e Y2.

```
Private Sub BtLigarY0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    SerialPort1.WriteLine("TO_PLC000END")
End Sub
```

Se quiser ligar ou desligar apenas uma das saídas digitais do PLC sem alterar as outras, considere antes o código seguinte:

Declare um array de “Char” com o nome “MsgToPLC”, e quando o programa “arrancar” inicialize o array com a mensagem inicial “TO_PLC000END”.

Se ao pressionar um botão, por exemplo o botão “BtLigarY1_EV_In” alterar apenas o elemento do array correto, sem alterar os outros elementos. Quando o procedimento Timer1_Tick for executado toda a mensagem é enviada para o PLC, sendo repetido de segundo a segundo o seu envio para que o PLC possa responder de segundo a segundo com o estado das suas entradas e saídas digitais.

```

Dim MsgPcToPLC() As Char = "TO_PLCO000END"

Private Sub BtLigarY0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(6) = "1"
End Sub

Private Sub BtLigarY1_EV_In_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(7) = "1"
End Sub

Private Sub BtLigarY2_EV_Out_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(8) = "1"
End Sub

Private Sub BtDesligarY0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(6) = "0"
End Sub

Private Sub BtDesligarY1_EV_In_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(7) = "0"
End Sub

Private Sub BtDesligarY2_EV_Out_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
    MsgPcToPLC(8) = "0"
End Sub

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    SerialPort1.Write(MsgPcToPLC)
    txtTextoEnviar.Text = MsgPcToPLC

```

Supervisão das saídas e das entradas digitais do PLC

Sempre que o PLC receber uma mensagem, responde ao computador com a mensagem:

TO_PC0000000END

O objecto SerialPort1 adquire e armazena os bytes/caracteres recebidos na sua memória interna. Dependendo do instante em que liga o PLC, ou o VBasic, o objecto serialPort1 pode armazenar uma mensagem completa, uma mensagem completa mais um fragmento da próxima mensagem ou o fragmento da mensagem anterior e parte da mensagem atual. O código em VBasic deve considerar todas estas hipóteses e processar apenas mensagens completas. Cada vez que o evento Tick do objecto Timer1 ocorrer, a variável MsgPlcToPc pode conter os caracteres:

Hipótese1:

MsgPlcToPc = “TO_PC0000000END” ‘ msg completa, pronta a ser processada

Hipótese2:

MsgPlcToPc = “000ENDTO_PC0000000END” ‘ contém parte da msg anterior que deverá ignorar

Hipótese3:

MsgPlcToPc = “TO_PC0000000ENDTO_PC000” ‘ contém uma msg completa e parte da seguinte

Hipótese4:

MsgPlcToPc = “000ENDTO_PC0000000ENDTO_PC00” ‘ contém parte da msg anterior, uma completa e parte da seguinte

....

O exemplo seguinte permite apenas processar corretamente mensagens completas (hipótese 1):

```

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1
    SerialPort1.Write(MsgPcToPLC)
    txtTextoEnviar.Text = MsgPcToPLC
    TxtRecebido.Text = MsgPlcToPc
    If Len(MsgPlcToPc) > 14 Then 'se recebeu mais de 14 carateres pode ter uma msg completa
        chk_Y0_Motor.Checked = Mid(MsgPlcToPc, 6, 1)
        chk_Y1_EV_In.Checked = Mid(MsgPlcToPc, 7, 1)
        chk_Y2_EV_Out.Checked = Mid(MsgPlcToPc, 8, 1)
        chk_X0_AlarmEmpty.Checked = Mid(MsgPlcToPc, 9, 1)
        chk_X1_Empty.Checked = Mid(MsgPlcToPc, 10, 1)
        Chk_X2_Full.Checked = Mid(MsgPlcToPc, 11, 1)
        chk_X3_AlarmMax.Checked = Mid(MsgPlcToPc, 12, 1)
        MsgPlcToPc = ""
    End If
End Sub

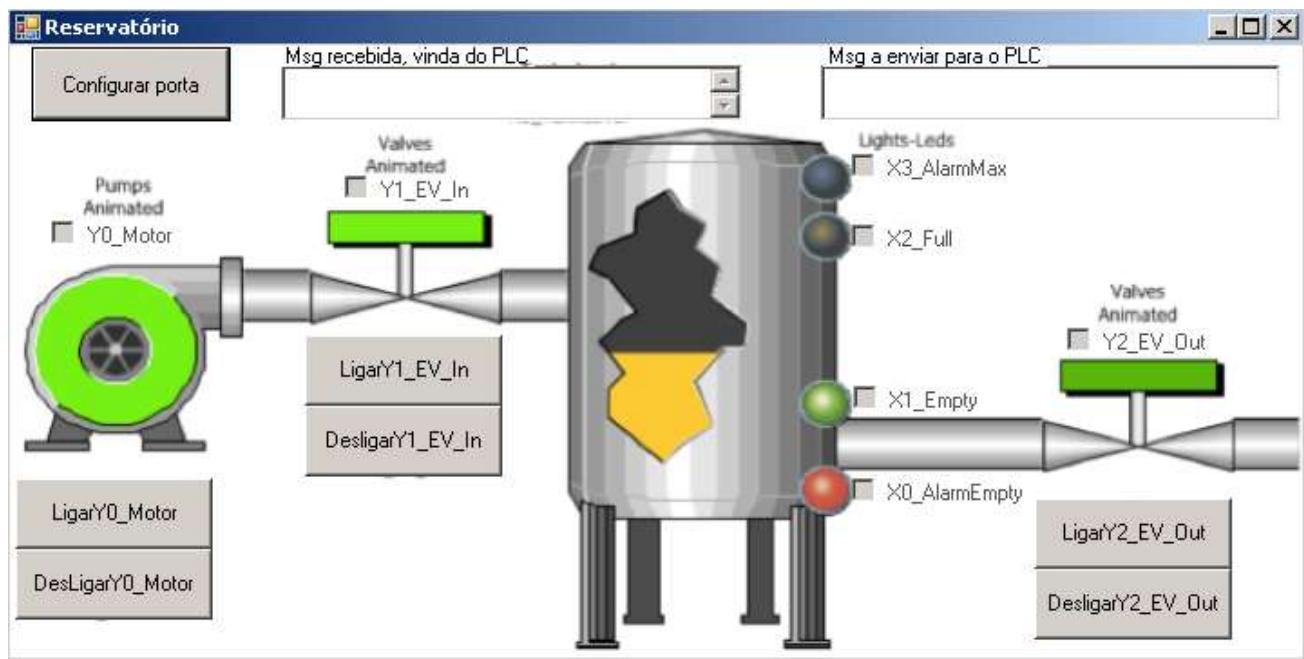
```

Trabalho de casa

- a) Se iniciar o Vbasic sem ter o conversor USB-Rs232 ligado ao computador, ou estivar na “Com” errada o programa bloqueia. Corriga o problema.
- b) O Timer1, só deveria ficar ativo depois de configurar a porta de comunicação para o programa não bloquear. Faça a modificação.
- c) Altere o código do sub procedimento Timer1_Tick(...), para identificar cada mensagem completa dentro da variável MsgPlcToPc, e “limpar” esta variável, ou parte dela, na altura certa. Para isso deverá usar as funções Mid, InStr,... em apêndice.
- d) O programa deve permitir guardar em disco as mensagens trocadas com o PLC. E a sua visualização posterior. Usando os objetos OpenFileDialog, SaveFileDialog, Streamreader, Streamwriter ou as funções FileOpen, PrintLine, ... em apêndice.

Deverá enviar para o elearning até à véspera da aula prática respectiva o código em VBasic.

5.6.5. Apêndice – Programa VBasic



```

Public Class Form1
    Dim rx As String
    Dim MsgPlcToPc As String

    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs)
        MsgPlcToPc = MsgPlcToPc & SerialPort1.ReadExisting
    End Sub

    Dim MsgPcToPLC() As Char = "TO_PLCOOOEND"

    Private Sub BtLigarY0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtLigarY0_Motor.Click
        MsgPcToPLC(6) = "1"
    End Sub

    Private Sub BtLigarY1_EV_In_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtLigarY1_EV_In.Click
        MsgPcToPLC(7) = "1"
    End Sub

    Private Sub BtLigarY2_EV_Out_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtLigarY2_EV_Out.Click
        MsgPcToPLC(8) = "1"
    End Sub

    Private Sub BtDesligarY0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtDesligarY0_Motor.Click
        MsgPcToPLC(6) = "0"
    End Sub

    Private Sub BtDesligarY1_EV_In_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtDesligarY1_EV_In.Click
        MsgPcToPLC(7) = "0"
    End Sub

    Private Sub BtDesligarY2_EV_Out_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtDesligarY2_EV_Out.Click
        MsgPcToPLC(8) = "0"
    End Sub

```

```

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    SerialPort1.Write(MsgPcToPLC)      'Envia o último comando/msg para o PLC
    txtTextoEnviar.Text = MsgPcToPLC   'Visualiza na caixa de texto a msg enviada
    TxtRecebido.Text = MsgPlcToPc     'Visualiza a msg recebida pelo SerialPort1
    If Len(MsgPlcToPc) > 14 Then     'se recebeu mais de 14 carateres pode ter uma msg completa
        chk_Y0_Motor.Checked = Mid(MsgPlcToPc, 6, 1)    'Ativa a checkbox se o 6 caracter dif de zero
        chk_Y1_EV_In.Checked = Mid(MsgPlcToPc, 7, 1)
        chk_Y2_EV_Out.Checked = Mid(MsgPlcToPc, 8, 1)
        chk_X0_AlarmEmpty.Checked = Mid(MsgPlcToPc, 9, 1)
        chk_X1_Empty.Checked = Mid(MsgPlcToPc, 10, 1)
        Chk_X2_Full.Checked = Mid(MsgPlcToPc, 11, 1)
        chk_X3_AlarmMax.Checked = Mid(MsgPlcToPc, 12, 1)
        MsgPlcToPc = ""      'Depois de processar a msg recebida é limpa
    End If
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    SerialPort1.PortName = VarPortname   'Com1, Com2, ....
    SerialPort1.BaudRate = VarBaudrate  '9600, 19200, ....
    SerialPort1.DataBits = VarDatabits   '7 ou 8 data bits
    SerialPort1.Parity = VarParity       'Nenhum bit de paridade, par ou impar
    SerialPort1.Handshake = VarHandshake 'Controlo de fluxo
    SerialPort1.StopBits = VarStopbits   '1, 1,5 ou 2 stop bit
    SerialPort1.Open()                 'IF SerialPort.isOpen = False THEN ...
    Timer1.Interval = 500   '500 mili
    Timer1.Enabled = True
    Me.BackgroundImage = Image.FromFile("reservatorio.png")
    Me.BackgroundImageLayout = ImageLayout.Stretch
End Sub

Private Sub BtSimulaMsgPlcToPc_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) :
    Dim MsgPlcToPc() As Char = "111ENDTO_PC1111111ENDTO_PC11"
    SerialPort1.Write(MsgPlcToPc, 6, 15)  'se usar um conversor USB-Rs232 com Tx ligado a RX
    'SerialPort1.Write(MsgPlcToPc, 0, 21)  '111ENDTO_PC1111111END
    'SerialPort1.Write(MsgPlcToPc, 0, 28)  '111ENDTO_PC1111111ENDTO_PC11
End Sub

Private Sub BtConfPorta_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtConfPorta.Click
    Form2.ShowDialog()  'Lançar o form2
    SerialPort1.PortName = VarPortname
    SerialPort1.BaudRate = VarBaudrate
    SerialPort1.DataBits = VarDatabits
    SerialPort1.Parity = VarParity
    SerialPort1.Handshake = VarHandshake
    SerialPort1.StopBits = VarStopbits
    SerialPort1.Open()
End Sub
End Class

```

O Form2 e o module1 são iguais aos do trabalho anterior
Module Module1

```

Public VarPortname As String = "Com2"
Public VarParity As Integer = IO.Ports.Parity.Even
Public VarBaudrate As Integer = 9600
Public VarHandshake As Integer = IO.Ports.Handshake.None
Public VarDatabits As Integer = 7
Public VarStopbits As Integer = IO.Ports.StopBits.One
End Module

```



CAPÍTULO

6

OPC

JPSantos
2023

6. OPC - Ole for Process Control

Introduction to OPC with Examples (free “Matrikon OPC server” and “Matrikon explorer/client”).

- Labview OPC Client example
- Mathlab OPC Client toolbox example
- VisualStudio example

<https://www.youtube.com/watch?v=E6ELXwzJFgE>

OPC Server & Client Data Communications – Introduction and Overview (Kepware, KEPServerEX)
<https://www.youtube.com/watch?v=u6E9uAtyhow>

6.1. Conceitos sobre OPC

OPC significava *OLE for Process Control* e não é mais que um conjunto de especificações que os programas de computador devem respeitar para monitorizar e controlar os recursos fabris. Estas especificações são o resultado do esforço conjunto de inúmeras empresas de automação que formaram a fundação OPC. Esta fundação tem por objectivo promover e definir as especificações necessárias para que as aplicações possam partilhar dados relativos à instalação fabril.

Atualmente, OPC significa *Open Process Control*.

Para ser possível comunicar com os recursos fabris a partir de um computador, é necessário respeitar um conjunto de requisitos, nomeadamente:

1. Os recursos fabris e os computadores têm de usar o mesmo tipo de interfaces/protocolos de comunicação: Rs232, Profibus, CAN, Ethernet, IP, TCP ou outra interface comum. Durante o desenvolvimento de novos programas existem alguns controlos genéricos que utilizam as interfaces de comunicação do computador, como o SeriaPort usado no capítulo 3 que permite o envio e a recepção de dados através da interface Rs232.
2. Contudo não basta transferir dados entre equipamentos usando as interfaces descritas no ponto anterior. Esses dados têm de ser corretamente interpretados e executados pelos recursos fabris.

Estes requisitos forçam o integrador de sistemas, o programador, a ter de conhecer um grande número de protocolos de comunicação e a ter de conhecer o formato dos dados que cada tipo de recurso fabril está à espera de receber. Devido à grande diversidade de recursos fabris, a integração destes recursos pode ser demorada e ter custos elevados.

Por exemplo, no capítulo 3, o integrador de sistemas tinha de conhecer o protocolo Rs232, o controlo *SeriaPort* e ainda o formato das mensagens série que o PLC estava à espera de receber ou enviar, para dessa forma conseguir controlar e monitorizar o seu funcionamento.

Procurando simplificar a comunicação e a integração dos seus recursos fabris, diversos fabricantes passaram a disponibilizar pacotes de software específicos “*proprietários*”. Se por um lado estes pacotes facilitam a comunicação com os seus recursos respectivos, continuavam a não permitir uma fácil integração de diferentes recursos fabris, de diferentes fabricantes, existentes numa instalação fabril.

Para colmatar essa limitação e agilizar o controlo integrado dos recursos numa instalação fabril, algumas organizações como a *OPC foundation* definiram um conjunto de regras comuns a todos os recursos fabris que queiram aderir às *especificações OPC*.

Estes pacotes de software dizem-se abertos e todos eles disponibilizam para o programador o mesmo tipo de métodos e funcionalidades, apesar de continuarem a comunicar com os seus recursos respectivos de forma proprietária.

Servidores OPC

Quando essas aplicações, fornecidas pelos fabricantes dos recursos fabris, obedecem à especificação OPC comportam-se como servidores OPC. Competindo a essas aplicações utilizar as interfaces físicas do computador para comunicar com os seus recursos fabris. Na secção 5.2 é apresentado um exemplo de um programa Windows que actua como um servidor OPC.

Clientes OPC

Os fabricantes dos recursos fabris fornecem também pacotes de software, por vezes sob a forma de controlos/activeX, disponíveis para o VBasic, que nos permitem comunicar com essas aplicações servidoras “dentro” do ambiente Windows, ou entre computadores diferentes.

Podemos adicionar ao nosso programa esses controlos, usando a “*Toolbox*” ou a opção “*Project – Add Reference*”, disponíveis no ambiente de desenvolvimento integrado do *Visual Basic Express Edition*. Estes controlos disponibilizam um conjunto de métodos/funções que em geral nos permitem ler e escrever em posições de memória desses recursos fabris.

A especificação “OPC” define o tipo de interacção entre as nossas aplicações e as aplicações desenvolvidas pelos fabricantes dos recursos fabris. Estas últimas atuam como servidores OPC e as nossas aplicações atuam como clientes OPC.

OPC Data Access Custom Interface Standard (v3.00)

A maior parte dos fabricantes fornece aplicações Windows que implementam uma interface *OPC data Access custom interface* para as outras aplicações Windows e por outro lado usam as interfaces de comunicação do computador para aceder aos recursos fabris, permitindo ler e escrever dados nesses recursos.

OPC security specification (v1.0)

Para evitar consequências graves, ou mesmo avarias, os dados relativos aos processos industriais (PLC e outros equipamentos) precisam de ser protegidos contra o acesso de entidades não autorizadas. Por esta razão a fundação OPC definiu um conjunto de mecanismos de segurança que todos os servidores OPC devem respeitar.

Existem dois níveis de segurança: o acesso aos dados pode ser controlado pelos objetos *DCOM – Distributed component Object Model*, num ambiente Windows, ou pode ser também assegurada pela especificação OPC.

OPC Alarms and Events Custom Interface (v1.10)

Esta especificação, quando implementada no servidor OPC evita que o cliente OPC esteja sempre a inquirir o servidor até que um evento ou alarme ocorra. De acordo com esta especificação, o cliente só necessita de informar previamente servidor de que evento pretende ser notificado, a partir dessa altura o servidor toma a iniciativa de notificar o cliente de que esse evento teve lugar. Esse evento pode ser um alarme, uma acção de um operador, ou outras.

OPC Data eXchange specification (v1.0)

A especificação *OPC data access custom Automation interface* define como um cliente OPC pode interagir com um servidor OPC, mas não prevê a possibilidade de um servidor OPC trocar dados com outro servidor OPC.

A especificação *Data Exchange DX* resolve esta limitação, os servidores OPC que implementem esta especificação podem estabelecer ligações DX entre eles. Podem também estabelecer ligações com servidores OPC – DA. Estas ligações podem ser vista como uma ligação só de leitura de itens OPC DA.

OPC Historical Data Access specification (v1.20)

Para facilitar a integração de recursos mais antigos, u que não possuam servidores OPC, a fundação OPC propôs a *Historical Access Data specification*. De acordo com esta especificação é possível definir servidores OPC que façam a interface entre os clientes OPC e essas soluções proprietárias.

OPC Complex Data (v1.00)

Os servidores OPC que implementem apenas a OPC Data Access Custom Interface Standard (v3.00) permitem que os clientes OPC leiam ou escrevam dados simples, como bits, bytes, words.

Os servidores que implementem o *OPC complex data* suportam a leitura e escrita de estruturas de dados compostas mais complexas.

OPC XML- DA (v1.01)

As especificações OPC anteriores previam uma interacção entre clientes e servidores apenas no ambiente Windows. Com esta nova especificação os clientes e servidores OPC passam a poder enviar mensagens de texto, escritas de acordo com o *XML – eXtended Markup Language*, através da internet. Mais exactamente, estas mensagens são enviadas dentro de envelopes *SOAP – Simple Object Access*

Protocol, que por sua vez são enviados através do protocolo *HTTP- Hypertext Transfer Protocol* entre computadores.

Esta especificação tem a grande vantagem de poder ser implementada em várias plataformas computacionais, nomeadamente UNIX, LINUX, entre outras.

Existem diversas aplicações comerciais que são Clientes OPC, no entanto é possível também desenvolver clientes OPC em várias linguagens de programação, permitindo assim personalizar a aplicação de acordo com as nossas necessidades.

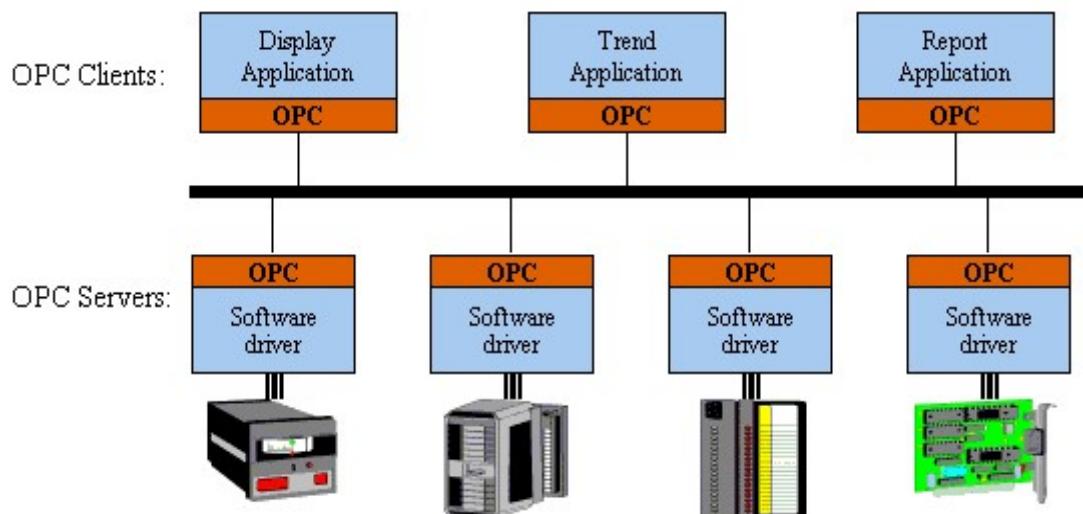


Figura: Interações entre Clientes e Servidores OPC

A especificação “*OPC Data Access Automation Interface*” define detalhadamente a arquitetura disponibilizada, pelos respectivos objetos e suas propriedades, métodos e eventos.

A especificação “*OPC Data Access Automation Interface*” é implementada (aproximadamente) pelo programa “FaconSvr.exe”.

6.2.FACONSRV Servidor da Fatek

O servidor OPC utilizado nesta secção é o *FaconSrv*, desenvolvido pela *FATEK*.

Antes de ser possível utilizar este programa para aceder ao PLC é necessário definir o tipo de ligação física existente entre o PLC e o computador onde reside o *FaconSrv*, é necessário definir que itens do PLC pretendemos aceder (ex. X1, Y2, M20,...). Podemos organizar esses itens em grupos (*group*) para cada um dos PLC (*station*) disponíveis e podemos organizar vários PLC num só canal de comunicação (*channel*).

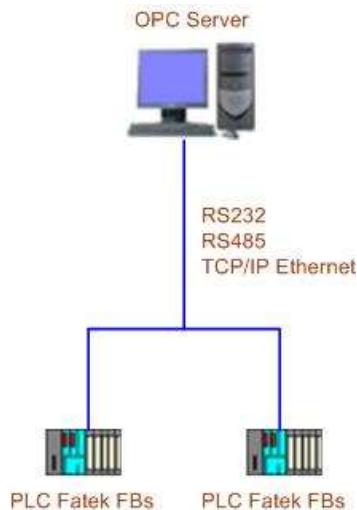


Figura: Comunicação Servidor OPC ↔ PLCs

FACON Server for Win8

http://www.fatek.com/en/download.php?f=data/ftp//PLC/Fatek_Server/FaSrv116-16523_en.zip

Depois de descargar o FaconSrv e antes de o instalar no seu computador, altere as propriedades do ficheiro que “descarregou” o “FaSrv116-16523-ENU.exe” para correr como administrador e ser compatível com o Windows XP. Para isso deve selecionar o ficheiro “FaSrv116-16523-ENU.exe” com o botão direito do rato e alterar as suas propriedades.



Figura: Fatek communication server - Definição de itens

Através da interface do FaconSrv é possível definir que itens se pretende monitorizar ou controlar é possível associá-los em grupos (*Group*) em cada PLC (*station*) e também é necessário definir qual a interface física de comunicação entre o computador e o PLC (*channel*).

Depois de o utilizador definir todo estes itens pode salvar em disco as opções que fez. Esse ficheiro tem a extensão (*.fcs), por exemplo “servidor.fcs”.

Esta organização dos itens em grupos e vários grupos de itens para cada PLC segue aproximadamente a organização proposta pela *OPC foundation* na especificação “*Data Automation Interface*” presente na figura seguinte.

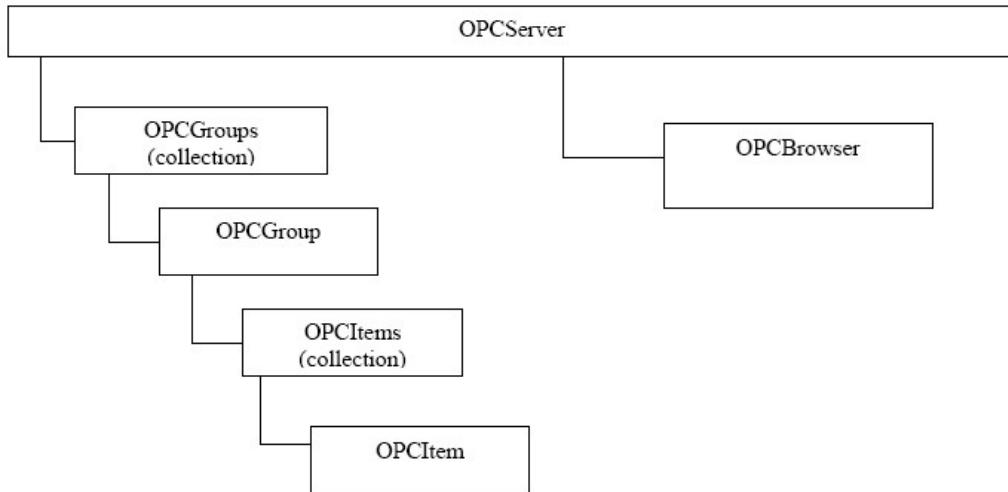


Figura: Hierarquia do “*Data Access Automation Interface*”, *OPC Foundation*.

6.2.1. Configuração do FACONSRV – exemplo

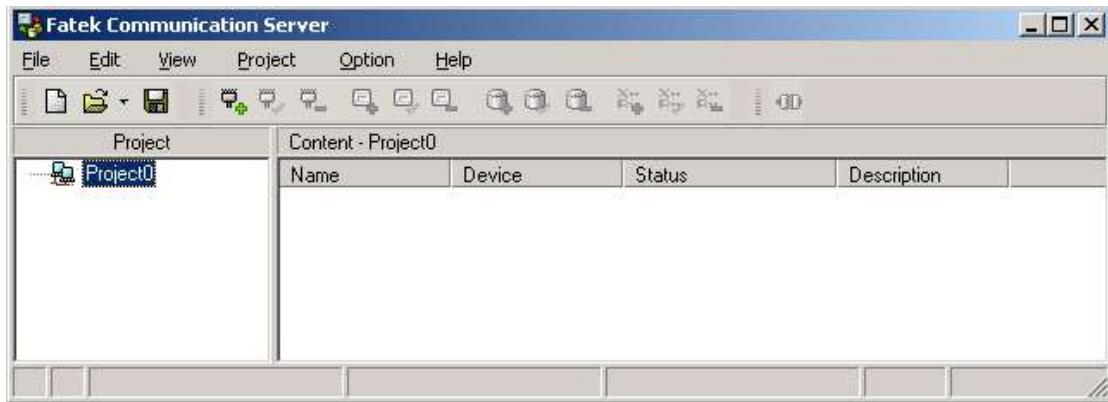
Neste exemplo, vamos utilizar o FaconSrv para monitorizar e controlar alguns itens do PLC, a partir da interface do FaconSrv, mesmo sem utilizarmos ainda o VBasic.

Itens : X0,X1,X2,X3, Y0,Y1,Y2, D4072, D4073, D4076

Como certamente se recorda:

- X0,X1, X2e X3 são entradas digitais do PLC, se aplicarmos uma tensão externa de 24 volt a essas entradas ativamo-las e o nosso programa autómato, ou o FaconSrv, pode detetar essa ocorrência.
- Y0,Y1,Y2 são saídas digitais do PLC e se o nosso programa autómato, ou se o FaconSrv, as ativar poderão disponibilizar 24 v para alimentar um dispositivo externo.
- A memória interna D4072 do PLC corresponde sua à entrada analógica nº0 (Analog Input 0 – AI0). Se aplicarmos uma tensão externa de 10V na entrada AI0, sem intervenção do nosso programa autómato, o PLC faz a aquisição, amostragem e quantificação do sinal eléctrico e guarda o valor 16380 na memória D4072. Consoante a tensão aplicada na entrada AI0 variar de 0 volt a 10 volt o valor de D4072 varia de 0 a 16380.
- A memória D4073 corresponde à entrada analógica nº 1 do PLC (AI1).
- A memória D4076 corresponde à saída analógica número zero do PLC (Analog Output nº 0 – AO0). Se o nosso programa autómato, ou o FaconSrv, escrever um número de 0 a 16380 na memória D4076 do PLC, surge uma tensão de 0V a 10 Volt na saída analógica AO0.

Quando ativar o FaconSrv no seu computador, selecione a opção File-new project..



Com o botão direito do rato sobre a palavra “Project0” selecione a opção “Add Device”. Para aceder ao novo dispositivo é necessário definir qual o canal de comunicação que será utilizado para comunicar com o PLC, por exemplo Rs232 9600,Even,7 bit de dados,1 stop bit.

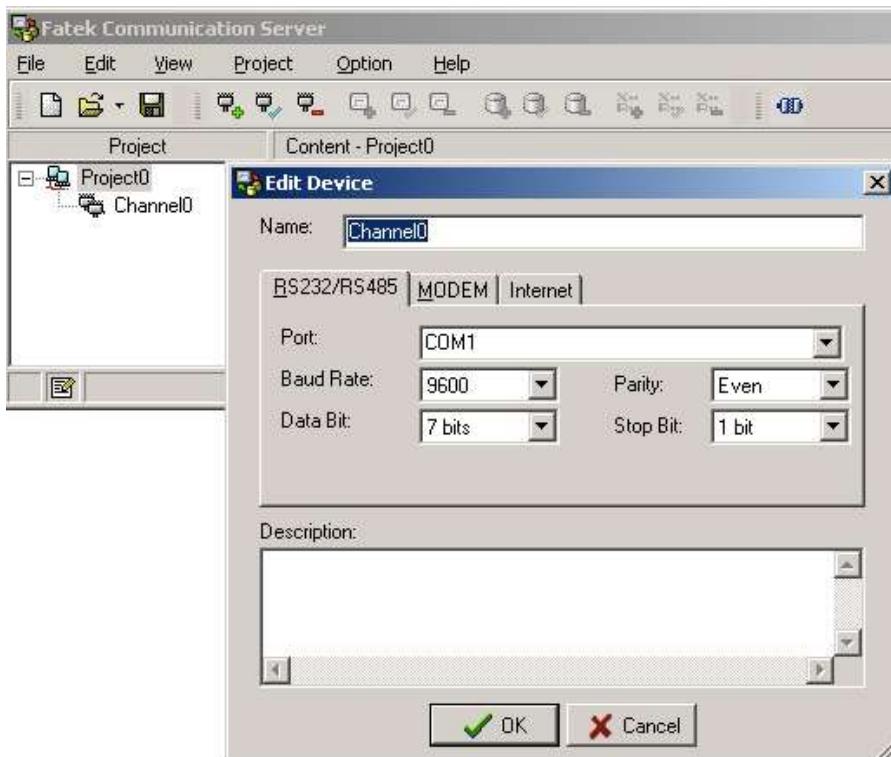
Com o botão direito do rato sobre a palavra “Channel0” agora criado, crie uma ou várias “Station”. Cada “Station” corresponde a um PLC.

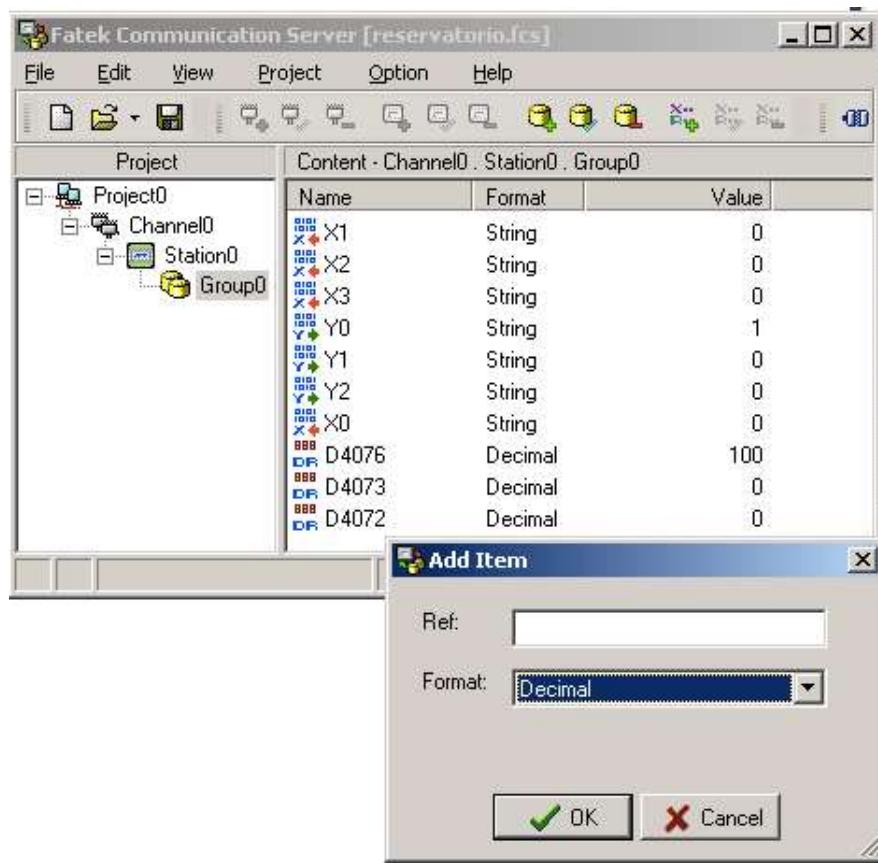
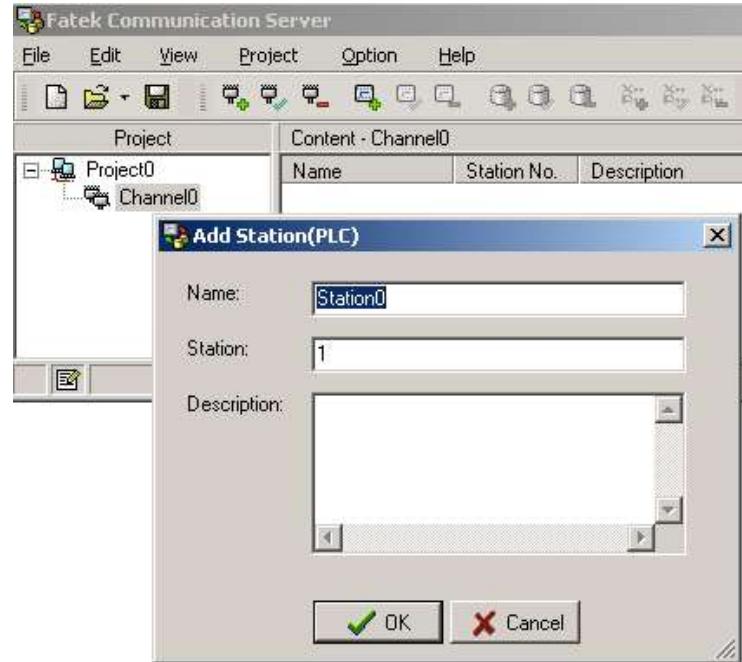
Com o botão direito do rato sobre a “Station0”, crie um grupo de items “Group0”.

Com o botão direito do rato sobre o “Group0” crie os itens “Add Item”: X0,X1,X2,X3, Y0, Y1,Y2, D4072,D4073,D4076

Depois de criados todos os Itens do Group0, da Station0 (PLC), acedida através do canal Channel0, deve gravar esta configuração num ficheiro em disco fazendo

File -Save Project. Grave o projeto em disco com o nome “Reservatorio.fcs”.





Depois de criar o projeto deve estabelecer a ligação do FaconSrv com o PLC, selecionando a opção “Project-Connect”, para que os itens do FaconSrv fiquem iguais aos itens reais do PLC.

Depois da ligação estabelecida pode “clicar” sobre o D4076 ou sobre o Y2 e atribuir-lhe um novo valor que essa alteração será imediatamente refletida no PLC, a sua saída digital será ativada, e a sua memória D4076 passará a ter o valor que escreveu no FaconSrv.

6.3. VBasic como cliente do servidor FACONSRV

Para poder desenvolver em VBasic um programa que seja cliente do FaconSrv.exe é necessário adicionar a referência “*FaconSrv Library 1.0*” ao projeto VB, selecionando a opção “Project – AddReference”, escolhendo a o separador “COM” e o “FaconSrv Library”, como na figura

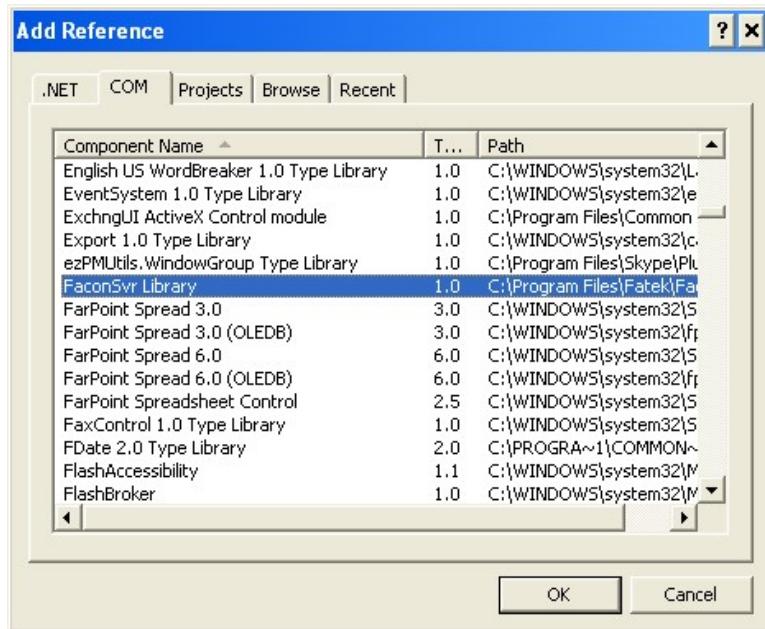


Figura: VBasic, adicionar a referência à “*FaconSrv Library*” no projeto

6.3.1. Objetos do tipo FaconSrv

Esta referência disponibiliza os métodos que se descrevem na tabela seguinte:

http://www.fatek.com/Download Page/English/Fatek_Server/FatekServer ActiveX-enu.pdf

Methods Description - FATEK (Doc.V1.0 05/13/2003)

<i>OpenProject</i>	Open the previous saved project file (with .fcs file extension). The function is the same as [open project] menu in the Facon Server Windows
<i>SaveProject</i>	Save the configuration data into the project file. The function is the same as [save project] menu in the Facon Server Windows
<i>Connect</i>	Start to connect the PLCs and retrieve the data continuously. The function is the same as [connect] menu in the Facon Server Windows
<i>Disconnect</i>	Terminate the connection with PLCs. The function is the same as [Disconnect] menu in the Facon Server Windows
<i>AddGroup</i>	Add a new data group. The function is the same as [Add group] menu in the Facon Server Windows
<i>EditGroup</i>	Edit the data group. The function is the same as [Edit group] menu in the Facon Server Windows
<i>DeleteGroup</i>	Delete the data group. The function is the same as [Delete group] menu in the Facon Server Windows
<i>AddItem</i>	Add a new data item for automatic retrieving. The function is the same as [Add item] menu in the Facon Server Windows
<i>DeleteItem</i>	Delete a data item. The function is the same as [Delete item] menu in the Facon Server Windows
<i>GetItem</i>	Get the value of a data item
<i>SetItem</i>	Write value into a data item

6.3.2. Exemplo: Cliente OPC – Facon server

Pretende-se desenvolver uma aplicação em *VBasic*, que seja cliente OPC. Esta aplicação deve ligar-se ao FaconServer através da interface “*OPC Data Access Automation*”, ser capaz de ler o estado da variável M10 do PLC e alterar o seu valor.

Não se esqueça de adicionar ao seu Projecto VB, o activeX *Faconsrv*. Para isso deverá aceder ao menu *Project* seleccionar a opção *Add reference*, o separador *COM* e finalmente selecionar o *Facon Srv Library 1.0*.



Figura: VBasic, Cliente OPC

6.3.3. Exemplo II: Cliente OPC – Facon server

The screenshot displays three windows related to the OPC Client example:

- Cliente do servidor OPC (FaconSrv)**: A Windows application window showing a control panel. It has two sections: "Supervisão" and "Controlo". In "Supervisão", there is a checkbox "Y0_Motor" checked, and a text input field "Vel_D4076" containing "100". In "Controlo", there are two buttons: "Ligar Y0_Motor" and "Desligar Y0_Motor", with a text input field "Ctr_Vel_D4076" containing "100". Below these are buttons for "Ligar FaconSrv ao PLC", "Desligar FaconSrv do PLC", and "Sair".
- Fatek Communication Server [reservatorio.fcs]**: A software interface for configuring PLC projects. It shows a tree view of "Project0", "Channel0", "Station0", and "Group0". Under "Group0", there are several items: X1, X2, X3, Y0 (checked), Y1, Y2, X0, D4076 (value 100), D4073 (value 0), and D4072 (value 0). A table below lists these items with their names, formats, and current values.
- Código Fonte (C#) - Form1.cs**: The source code for the OPC Client application. It defines a class `Form1` with methods for loading the project, setting up a timer, updating controls, and handling button clicks for connecting and disconnecting from the PLC.

```

Public Class Form1

    Dim fs As New FaconSrv.FaconServer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.OpenProject("c:\reservatorio.fcs")
        Timer1.Interval = 1000
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
        txtVel_D4076.Text = fs.GetItem("Channel0.Station0.Group0", "D4076")
        chk_Y0_Motor.Checked = fs.GetItem("Channel0.Station0.Group0", "Y0")
    End Sub

    Private Sub Vel_D4076_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.SetItem("Channel0.Station0.Group0", "D4076", txtCtr_Vel_D4076.Text)
    End Sub

    Private Sub Bt_Ligar_Y0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.SetItem("Channel0.Station0.Group0", "Y0", "1")
    End Sub

    Private Sub Bt_Desligar_Y0_Motor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.SetItem("Channel0.Station0.Group0", "Y0", "0")
    End Sub

    Private Sub BtLigarFaconPLC_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.Connect()
    End Sub

    Private Sub BtDesligarFaconPLC_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        fs.Disconnect()
    End Sub

    Private Sub Bt_Sair_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        End
    End Sub
End Class

```



Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

Trabalho nº 6- Controlo e supervisão de um PLC através de um servidor OPC

6.3.4. Introdução

Neste trabalho, pretende-se monitorizar e controlar o reservatório de água como no trabalho anterior, mas desta vez sem usar o objecto SerialPort1, que permitia aceder diretamente à porta Rs232 do computador e trocar mensagens com o PLC.

Para consolidar os conhecimentos adquiridos nas aulas teóricas sobre as especificações OPC (Open Process Control), neste trabalho pretende-se fazer o mesmo que no trabalho anterior, mas agora, através de um servidor OPC (Prog.FaconSrv).

Como a figura ilustra, o servidor OPC fornecido pelo fabricante do autómato (Prog.FaconSrv) é uma aplicação Windows que comunica por um lado com a nossa aplicação em Visual Basic (Prog. VBasic), e com o PLC através das interfaces de comunicação do computador (ex. Rs232, TCP/IP).

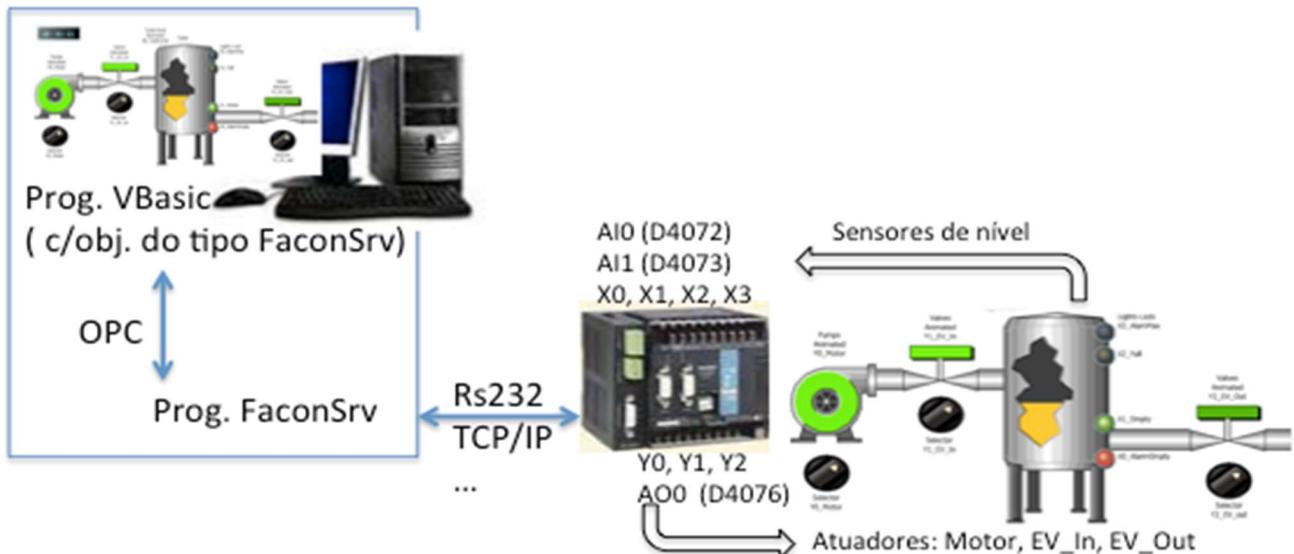


Figura : Controlo e supervisão de um PLC, via OPC

Na aplicação Windows a desenvolver (em VBasic) é necessário incluir um objecto do tipo “FaconSrv” para comunicar com o Servidor OPC (Prog.FaconSrv).

O PLC vem de fábrica com um programa autómato capaz de comunicar com o Servidor OPC (Prog.FaconSrv).

Como no trabalho anterior dispomos de um motor “***Y0_Motor***” e uma electroválvula de entrada de água “***Y1_EV_In***” para enchimento do reservatório. Uma electroválvula de saída “***Y2_EV_Out***” que permite a saída da água do reservatório, se o nível de água no reservatório for superior ao nível “***X0_AlarmEmpty***”. E, 4 sensores de nível:

- “***X0_AlarmEmpty***” - Alarme depósito vazio,
- “***X1_Empty***”, - Depósito vazio,
- “***X2_Full***” - Depósito cheio
- “***X3_AlarmMax***”. - Alarme depósito cheio

Os sensores ficam ativos quando detetam a presença de água, a entrada digital respetiva, recebe 24 V DC, ou seja, a entrada está a “True”.

O motor e a electroválvula de entrada de água devem ser acionados, quando o nível de água for inferior ao nível “***X1_Empty***” e desligados quando o nível de água igualar ou exceder o nível “***X2_Full***”.

Devem ser gerados alarmes no écran do computador se o nível de água atingir ou exceder o nível “***X3_AlarmMax***” ou passar a baixo do nível “***X0_AlarmEmpty***”.

Utilizar a entrada analógica AI0 (**D4072**) para ler a tensão/nível de água do sensor de nível ultrassónico.

Utilizar a saída digital AO0 (**D4076**) para que o PLC possa gerar uma tensão analógica de 0 a 10 volt. Nos próximos trabalhos poderá ser usada para variar a velocidade/RPMs do motor de enchimento do reservatório.

Resumo das entradas e saídas do PLC:

- Y0 - Comando do Motor (Motor) de enchimento de água;
- Y1 – Comando da Electroválvula de entrada de água (EV_In);
- Y2 – Comando da Electroválvula de saída de água (EV_Out);
- X0 – Nível de alarme de água baixo (AlarmEmpty);
- X1 – Nível de água baixo (Empty);
- X2 – Nível de água quando o reservatório está cheio (Full);
- X3 – Nível de Alarme de água alto do reservatório (AlarmMax);
- D4072 - Entrada analógica/ sensor de nível ultrassónico;
- D4076 - Saída analógica (0..10V) para uso futuro;

6.3.2. Objetivos

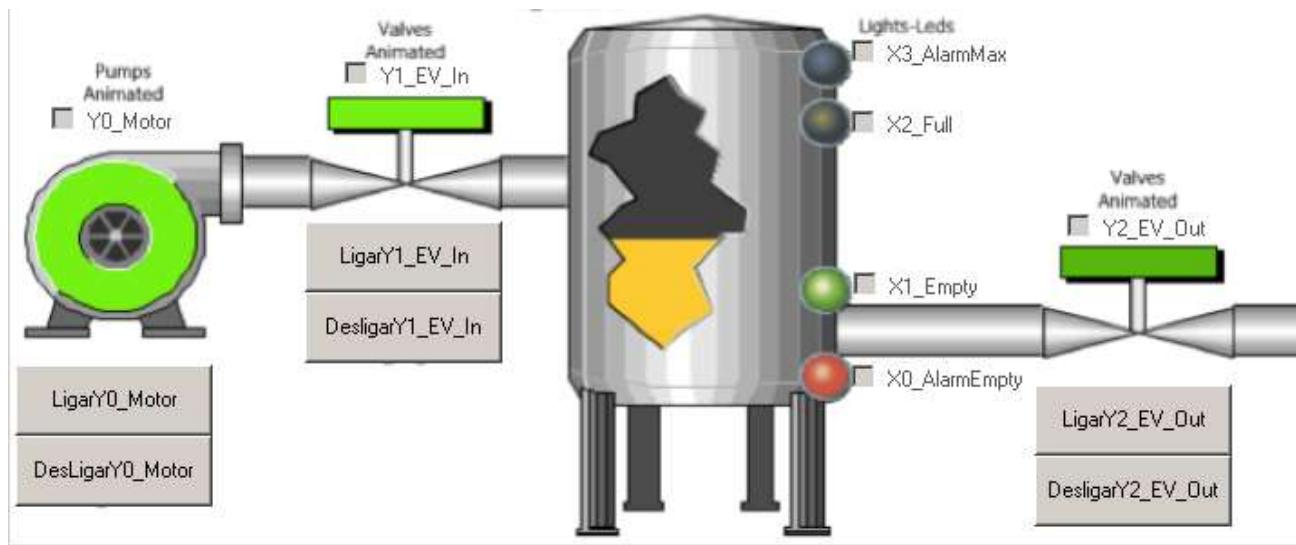
O trabalho tem por objectivo consolidar os conhecimentos sobre as especificações OPC, pretende-se:

- Aprender a instalar e a configurar um servidor OPC, neste caso o programa FaconSrv.
- Aprender a usar um objecto do tipo “cliente OPC”, no projecto em VBasic, para ler e escrever nos Items do PLC (M, D, R, X, Y,...).

6.3.3. Descrição do trabalho (1 semana)

Deve ser desenvolvida uma aplicação em VisualBasic, que comunicando com um servidor OPC da Fatek (FaconSrv ou Fatek OPC server), permita a supervisão e o controlo remoto das variáveis do processo.

- Supervisão das entradas: X0,X1,X2,X3, das saídas: Y0,Y1,Y2 e da entrada analógica AI0 (memória D4072).
- Controlo manual das saídas do PLC: Y1,Y2,Y3 e da saída analógica AO0 (memória D4076).



Depois de instalar o Prog.FaconSrv, siga os passos de configuração do programa FACONSRV server no documento de apoio à aula, e os exemplos.

Trabalho de casa

- O VBasic deve fazer o controlo automático das saídas do PLC em função dos níveis de água do reservatório.

Importante:

- Neste trabalho o PLC não tem programa, todo o controlo e supervisão é efectuado a partir do computador/VBasic.
- Deverá enviar o código desenvolvido para o elearning até à véspera da aula prática seguinte, sob pena da nota obtida no questionário não ser considerada.
- O trabalho será avaliado por questionário individual, na semana seguinte à entrega do mesmo.

6.3.4. Bibliografia

Fatek FACONSRV (programa)

http://www.fatek.com/en/download.php?f=data/ftp//PLC/Fatek_Server/FaSvr116-16523_en.zip

Fatek FACONSRV (Manual)

http://www.fatek.com/en/data%2Fftp%2FFPLC%2FFatek_Server%2FFatekServerActiveX_enu.pdf

OPC Foundation

<https://opcfoundation.org>

Introduction to OPC for Factory Automation and Plant Process Control (4 min)

https://www.youtube.com/embed/16fM_7vZ_FE

What is OPC? Part1: OPC Overview (6 min)

<https://www.youtube.com/watch?v=mK-OL03LaGg>

What is OPC? Part 1: OPC UA Overview (1 min)

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

What is OPC UA

<http://www.matrikonopc.com/opc-ua/index.aspx>



CAPÍTULO

7

ETHERNET

JPSantos
2023

7. ETHERNET

7.1. Introdução

O protocolo Ethernet foi proposto pela empresa Xerox e baseou-se numa rede de comunicação desenvolvida pela Universidade de ALOAH no Havai. Como a rede do Havai usava sinais de radiofrequência e o meio de transmissão era o ar “Éter”, a nova rede “Net” proposta pela Xerox adoptou a designação “Ether Net”.

Em 1983 o *IEEE - Institute of Electrical and Electronics Engineers* aprovou este protocolo sob a designação *IEE802.3 CSMA/CD*.

Em traços gerais, este protocolo define que cada equipamento antes de enviar os seus dados/sinais deve escutar o meio de transmissão (CS - Carrier Sense). Quando não existir atividade no meio deve então enviar os seus dados. Por outro lado, este protocolo permite que vários equipamentos possam aceder ao meio de transmissão (MA - Multiple Access).

Apesar de cada equipamento escutar previamente o meio de transmissão, pode acontecer dois equipamentos iniciarem o envio de dados precisamente no mesmo instante, dando origem a uma distorção dos sinais enviados. Por esta razão, cada equipamento deve verificar se os sinais que está a enviar correspondem aos sinais presentes no meio de comunicação (CD - Collision Detect).

No caso de ocorrer uma colisão o equipamento emissor espera um tempo aleatório (Backoff time) e volta a tentar enviar os dados.

Resumidamente, tal como todos os outros protocolos de comunicação, este protocolo tem por objectivo a transferência de dados entre equipamentos. Neste caso podem ser transferidos de cada vez, em cada mensagem Ethernet, 1500 bytes.

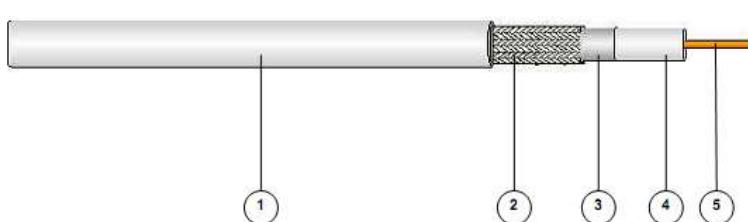
7.2.Camada física

O protocolo Ethernet prevê a existência de vários meios de transmissão: cabo coaxial fino, cabo coaxial grosso, par trançado ou fibra óptica.

7.2.1. 10Base2 - Meio de transmissão – Coaxial

A designação “Ethernet 10Base2” significa que o meio de transmissão consiste num um *cabo coaxial fino* com 6,3 mm de diâmetro com 50 Ohm de impedância (RG58). De acordo com o protocolo são enviados 10 milhões de bits por segundo através deste cabo.

Cabo coaxial fino RG58



1. Revestimento exterior
2. Malha de blindagem
3. Fita de blindagem
4. Dielétrico
5. Condutor central



Figura : Cabo coaxial

Características médias dos cabos coaxiais flexíveis		Tipo de Cabo Coaxial				
		RG59	RG6	RG7	RG11	
Diâmetro do cabo coaxial (mm)		6,14	6,93	8,08	10,29	
Revestimento Exterior Composição		PVC	PVC	PVC	PVC	
Malha de blindagem (simples ou dupla)	Composição	Cobre ou alumínio	Cobre ou alumínio	Cobre ou alumínio	Cobre ou alumínio	
		Diâmetro (mm)	4,49	5,38	7,32	
Fita de blindagem (simples ou dupla)		Cobre ou alumínio	Cobre ou alumínio	Cobre ou alumínio	Cobre ou alumínio	
Dielétrico	Composição	Poliétileno	Poliétileno	Poliétileno	Poliétileno	
		Diâmetro (mm)	3,65	4,57	5,72	
Condutor Central	Composição	Aço cobreado	Aço cobreado	Cobre estanhado	Cobre estanhado	
		Diâmetro (mm)	1 x 0,81	1 x 1,016	1 x 1,30	
Capacidade Nominal (pF/m)		53	53	53	53	
Impedância Característica (Ω)		75	75	75	75	

Topologia

A rede local Ethernet prevê diversos meios de transmissão de dados e várias topologias. No caso da Ethernet 10Base2 a topologia definida é o barramento (*Bus*).

Na figura seguinte pode ver um segmento de **200 metros** de cabo coaxial onde estão ligados todos os equipamentos daquele segmento.

Cada segmento pode ter até 185 metros, podem no entanto ser utilizados repetidores eléctricos para ligar até 5 segmentos iguais aos da figura.

Com este meio de comunicação o diálogo entre equipamentos é do tipo *half-duplex*.

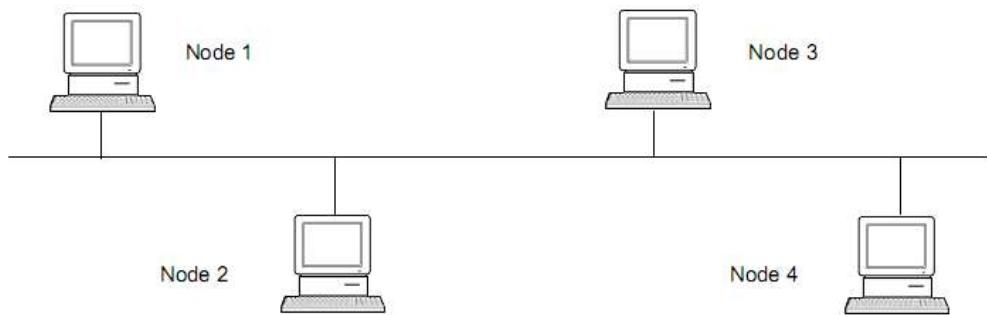


Figura : Topologia - cabo coaxial

Derivações T: Para ligar um equipamento ao cabo coaxial é necessário cortar o cabo coaxial e aplicar um T.



Placa de rede: Cada equipamento necessita de uma placa de rede Ethernet. Cada placa liga ao cabo coaxial através de um T



Terminadores: Em cada extremo do segmento é necessário aplicar uma resistência de 50 Ohm, entre a malha e o vivo. Habitualmente coloca-se o terminador no T dos equipamentos mais afastados.

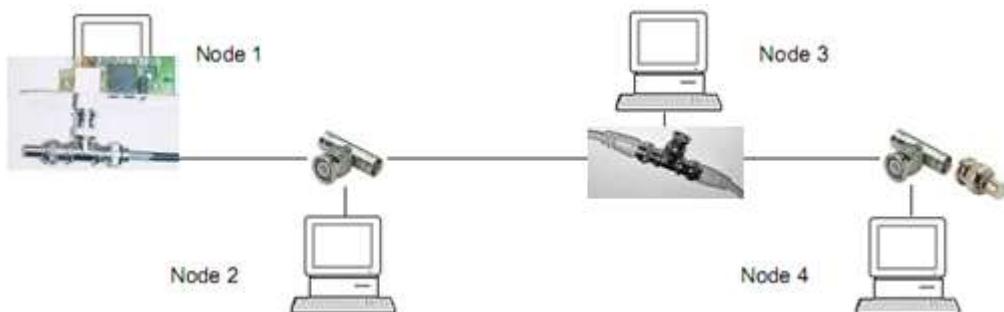


Figura : Placas de rede, fichas e terminadores

Sinais eléctricos

No cabo coaxial os sinais eléctricos podem variar de -2,5 a 2,5 volt.

7.2.2. 10Base5 - Meio de transmissão – Coaxial

Para uma ligação 10Base5 a velocidade de transmissão contínua a ser de 10Mbps mas o cabo coaxial utilizado tem um diâmetro maior (13 mm) e uma atenuação menor, o que possibilita ter troços de 500 metros cada. A topologia definida continua a ser a descrita na figura (barramento). Este cabo coaxial tem cor amarela ou laranja e tem uma impedância de 50Ω .

Com este meio de comunicação o diálogo entre equipamentos é do tipo *half-duplex*.

7.2.3. 10Base-T: Meio de transmissão – UTP, sinais e topologia

O protocolo Ethernet também prevê a possibilidade dos equipamentos estarem ligados através de pares entrançados “*UTP – Unshielded Twisted Pair*” a repetidores eléctricos (Hubs), assumindo uma *topologia em estrela* como a da figura (10Base-T).



Figura : Cabo UTP e Ficha RJ45

Ficha 10BaseT

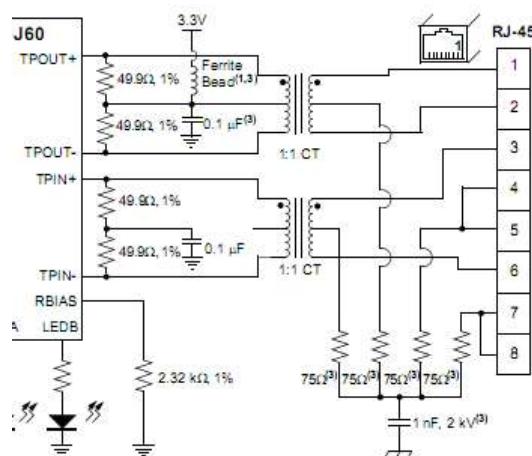


Figura : Ficha RJ45

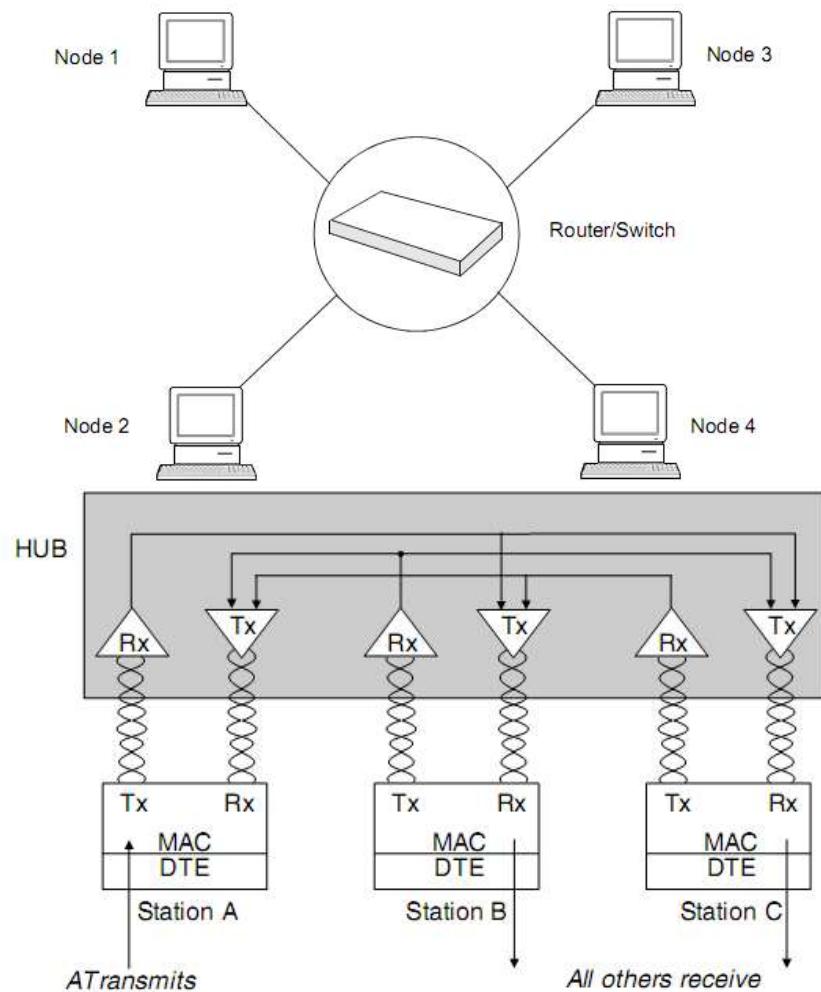


Figura : Hub

A distância máxima entre cada equipamento e o HUB é de 100 metros.

São utilizados cabos 24AWG, um par de condutores para enviar e outro par para receber dados. Com este meio de comunicação o diálogo entre equipamentos é do tipo *full-duplex*.

Um cabo UTP de categoria 5 permite taxas de transferência até 1 gigabit por segundo.

Um cabo UTP de categoria 6 permite taxas de transferência até 10 gibabits por segundo.

Sinais eléctricos

Os sinais eléctricos aplicados a um par entrançado podem variar de -2,5 a 2,5 volts.

É utilizada a codificação de Manchester: o bit '0' é representado como uma transição do sinal de TX+ de +2,5v para -2,5v e o bit '1' é representado como uma transição do sinal TX+ de -2,5v para 2,5V.

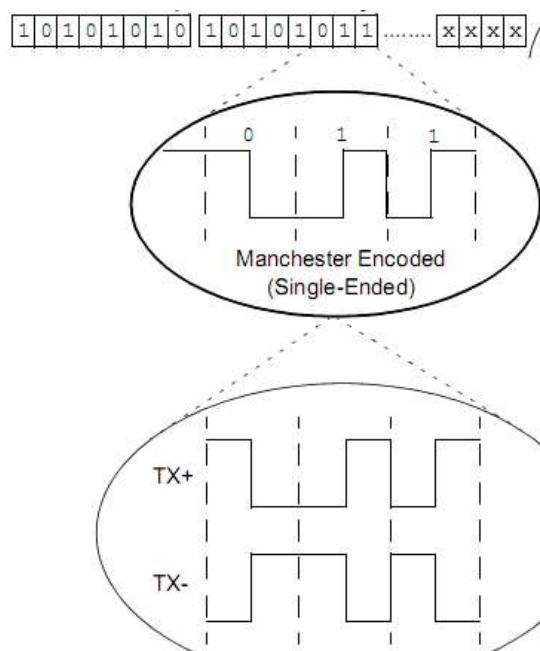


Figura : Sinais eléctricos

O sinal elétrico demora alguns microsegundos a propagar-se no condutor até poder chegar ao equipamento mais distante. Este tempo é conhecido por “transmission path delay”.

Por esta razão existe um intervalo de tempo conhecido por “slot time” em que o equipamento emissor, mesmo depois de ter terminado de enviar os dados, deve continuar a escutar o meio de transmissão para poder detetar uma eventual colisão de dados.

O slot time corresponde ao dobro do tempo de propagação mais um tempo de segurança. Para uma comunicação a 10 Mbps de uma mensagem de 512 bit

7.3.Mensagem Ethernet

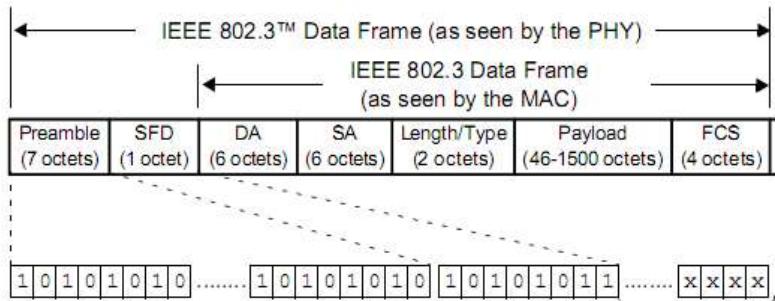


Figura : Mensagem Ethernet

Preambulo

Cada mensagem Ethernet começa pelo envio de sete bytes com o valor 0x55. Isto permite que o receptor se sincronize com a frequência dos sinais eléctricos enviados.

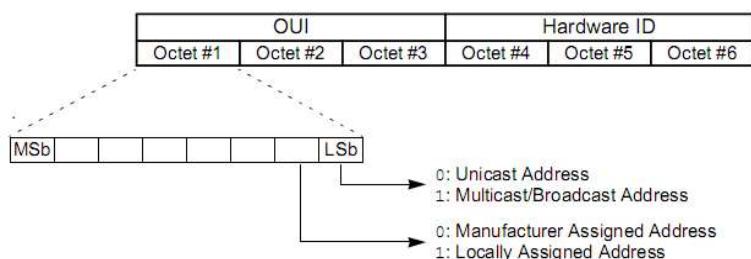
Start of frame

Depois do preâmbulo, o emissor envia o byte ‘10101011’ indicado o início da mensagem.

DA e o SA

O endereço de destino e de origem têm 6 bytes cada um. São conhecidos como *MAC address* e são definidos em geral pelo fabricante da placa Ethernet.

Broadcast: FF-FF-FF-FF-FF-FF



Example: A Microchip owned MAC Address.

OUI			Hardware ID		
Octet #1	Octet #2	Octet #3	Octet #4	Octet #5	Octet #6
00	04	A3	00	00	01

Figura : MAC address (AN1120)

Lenght ou type

Este campo é composto por dois bytes e pode conter um número de 0 a 65535. Este valor pode indicar o comprimento do campo de dados, ou em alternativa pode indicar que tipo de informação vai no campo de dados. Se esse número for menor que 1500 indica o nº de bytes que o campo de dados contém. Se esse número for superior a 1536 indica o tipo de dados que o campo de dados contém. Por exemplo:

IPv4 =0x0800

IPv6 = 0x86DD

ARP = 0x0806
RARP = 0x8035

Dados

O campo de dados pode conter no máximo 1500 bytes e no mínimo 46 bytes.

PAD

Se o campo de dados transportar menos de 46 bytes é necessário acrescentar bytes a zero até perfazer os 46 bytes.

Frame Check Sequence

Este campo é composto por 32 bits que são gerados pelo equipamento emissor segundo algoritmo CRC32.

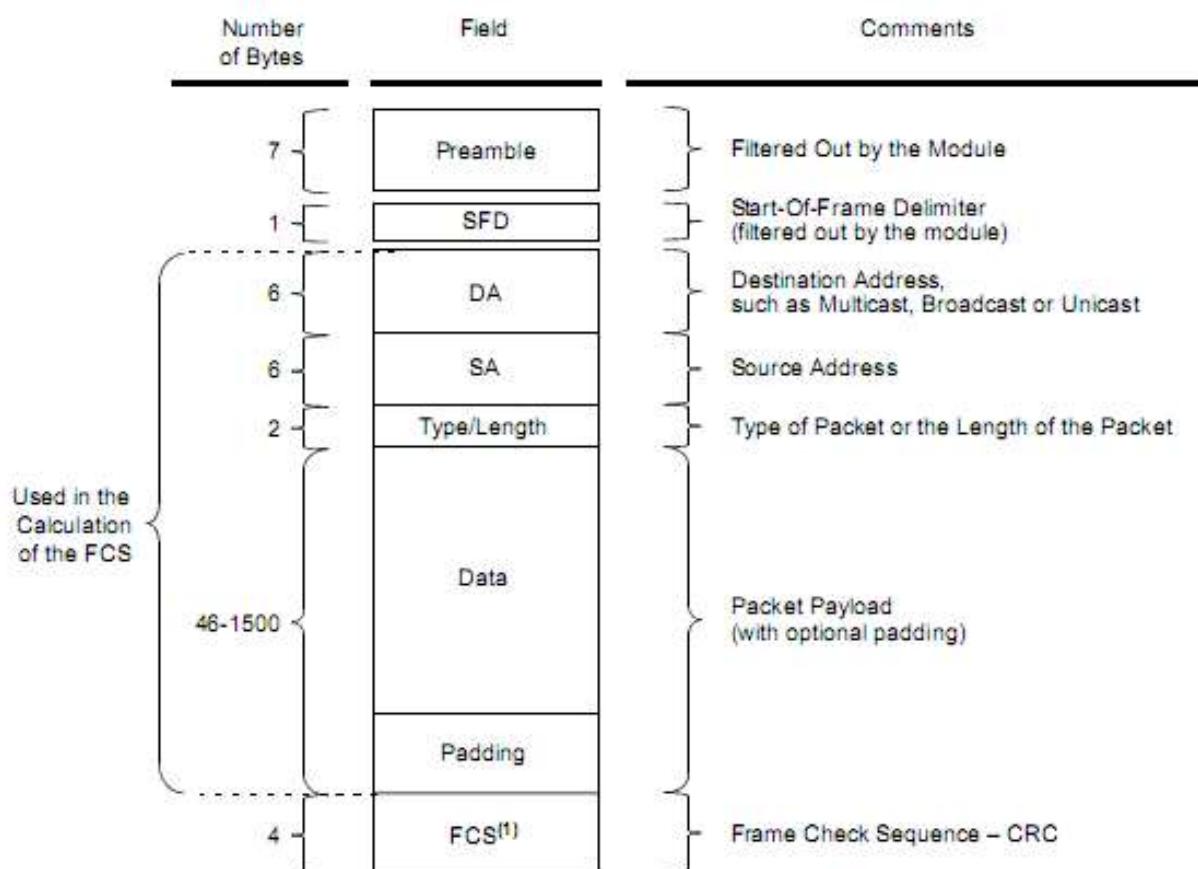
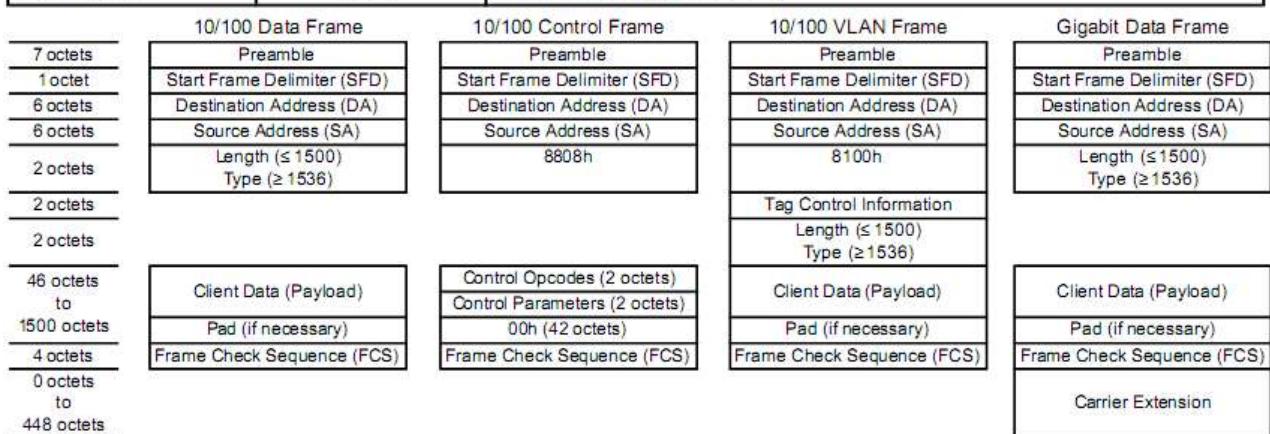


Figura : CRC da Mensagem Ethernet

Existem outros tipos de mensagens Ethernet

IEEE 802.3a	1985	10Base-2 Thin Ethernet
IEEE 802.3c	1985	10 Mb/s Repeater Specification
IEEE 802.3d	1987	Fiber Optic Inter-Repeater Link
IEEE 802.3i	1990	10Base-T Twisted Pair
IEEE 802.3j	1993	10Base-F Fiber Optic
IEEE 802.3u	1995	100Base-T Fast Ethernet and Auto-Negotiation
IEEE 802.3x	1997	Full-Duplex Standard
IEEE 802.3z	1998	1000Base-X Gigabit Ethernet (SX, LX, CX)
IEEE 802.3ab	1999	1000Base-T Gigabit Ethernet over Twisted Pair
IEEE 802.3ac	1998	Frame Size Extension to 1522 Octets for VLAN Tagging
IEEE 802.3ad	2000	Link Aggregation for Parallel Links
IEEE 802.3af	2003	Power Over Ethernet (PoE)



Bibliografia

[AN1120] Ethernet theory of operation, ApplicationNote AN1120, Microchip.

[802.3] IEEE Standard for Information technology Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, IEEE Std 802.3-2002

[Mackay] Mackay, S., et.al. - *Practical Industrial Data Networks: Design, Installation and Troubleshooting*, Elsevier, 2004
capítulo 15 – Industrial Ethernet overview



CAPÍTULO

7

INTERNET

JPSantos
2019

8. IP - INTERNET PROTOCOL

8.1. Conceitos sobre o protocolo IP

O protocolo IP foi desenvolvido pelo departamento de defesa dos Estados Unidos no âmbito no projecto *DARPA – Defense Advanced Research Projects Agency*.

Este protocolo foi desenvolvido para permitir a troca de blocos de dados (*datagram*) entre computadores. Cada equipamento tem um endereço definido pelo próprio protocolo, de forma a permitir encaminhar os blocos de dados de equipamento em equipamento até ao equipamento de destino. Este protocolo permite também a fragmentação dos dados a enviar, em mensagens mais pequenas, que possam ser transmitidas através de redes locais como a Ethernet. Convém relembrar que as mensagens Ethernet têm um comprimento máximo de 1500 bytes de dados. Os equipamentos que implementarem este protocolo podem também reagrupar esses fragmentos quando os receberem reconstruindo dessa forma a mensagem completa.

Este protocolo permite o envio de dados de forma independente independentemente da(s) rede(s) físicas que existirem entre eles (Telefónica, Ethernet, X.25, Token bus,). Para isso este protocolo cria uma rede virtual (Rede Internet) baseado nos endereços Internet.

Este protocolo fornece à camada de transporte um serviço de transferência de informação não confirmado, sem estabelecimento de ligação (*Connection Less*). Desta forma a troca fiável de mensagens entre as camadas de transporte não é garantida. O percurso destas mensagens através dos vários sistemas intermédios não é constante, podendo por isso chegarem fora de ordem ao destino.

8.1.1. Mensagem Internet

A unidade de transferência de dados deste protocolo chama-se mensagem IP. Esta mensagem é constituída pelo cabeçalho e pelos dados. Os dados podem ser as mensagens da camada de transporte. O cabeçalho contém entre outras coisas os endereços origem e destino das mensagens Internet. O comprimento máximo desta mensagem não está limitado pelo protocolo no entanto devemos ter em mente que esta mensagem irá ser “transmitida “dentro de mensagens da camada inferior”, por exemplo dentro de mensagens Ethernet, até ao equipamento de destino.

A mensagem IP é enviada para a rede depois de encapsulada na mensagem da camada lógica ou diretamente (*MAC-frame*) de uma rede local, habitualmente na mensagem Ethernet. Assim cada mensagem IP (cabeçalho e dados) usa campo de dados da uma mensagem Ethernet para ser transmitida ao longo de cada uma das redes públicas ou locais que necessitar de atravessar para alcançar o

Bits																
4	4	8	16	16	3	3	12	8	16	32	32	variable	8	variable		
Version	Header Length	Service Type	Length of Datagram	Datagram ID	Flags	Fragment Offset	Time to Live	Protocol	Header Checksum	Source IP Address	Destination IP Address	Options	Padding	Data		

equipamento de destino.

Figura : Mensagem IP

Vers: Estes quatro bits indicam qual a versão do Protocolo IP usado. No caso da versão do mensagem IP ser diferente da versão usada pelos gateways ou pelo receptor, os mesmos pode não ser aceites.

Comprimento IHL: Comprimento do cabeçalho em palavras (32 bits). O comprimento do cabeçalho pode ser variável dado que algumas mensagens usam o campo Opções enquanto outros não.

Tipo de Servico: Cada mensagem tem uma prioridade que pode variar entre 0 e 7. Pode ser pedido a rede que entregue o a mensagem com pequeno atraso, fiabilidade e no caso de existirem vários caminhos à escolha que seja usado o mais rápido (maior largura de banda).

Este campo usa 8 bits. Em geral os gateways não implementam estes serviços assim este campo é usado para pedir e não exigir à rede a prestação destes serviços.

Bits 0-2: Precedence.

- 111 - Network Control
- 110 - Internetwork Control
- 101 - CRITIC/ECP
- 100 - Flash Override
- 011 - Flash
- 010 - Immediate
- 001 - Priority
- 000 - Routine

Bit 3: 0 = Normal Delay, 1 = Low Delay.

Bits 4: 0 = Normal Throughput, 1 = High Throughput.

Bits 5: 0 = Normal Reliability, 1 = High Reliability.

Bit 6-7: Reserved for Future Use.

0	1	2	3	4	5	6	7
			D	T	R	0	0

Comprimento total: Este campo indica o comprimento da mensagem IP (cabecalho e dados) em bytes.

Ident: Todos os fragmentos da mensagem possuem o mesmo valor no campo Identificador. Assim este campo permite ao receptor identificar e reagrupar os vários fragmentos da mesma mensagem original.

Flags: Este campo tem 3 bits apenas.

- Bit 0: reserved, must be zero
- Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
- Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

Fragment Offset: Como o percurso seguido pelas mensagens IP não é constante, a sua ordem de chegada não é sempre igual à de envio. Assim o receptor necessita de saber de que parte da mensagem é o fragmento recebido. Este campo especifica o offset do fragmento na mensagem original. O valor do offset é medido em unidades de 8 bytes.

Tempo: Time to Live. Este campo especifica o tempo de vida em segundos de uma mensagem (ou fragmento). Este tempo de vida é decrementado por cada gateway/router que transmite o fragmento. Desta forma não se corre o risco de uma mensagem circular indefinidamente na rede.

Protocolo: Este campo é usado pelos protocolos de mais alto nível (TCP ou UDP) para indicar qual deles está a usar a mensagem IP.

Cabeçalho Checksum: Permite verificar a integridade do cabeçalho do pacote IP. A máquina que envia o pacote calcula um inteiro com base no cabeçalho do seu pacote e a máquina receptora faz a mesma coisa quando recebe esse pacote. Se os dois inteiros calculados não forem iguais significa que houve um erro de transmissão. Como o cálculo do *CheckSum* só se aplica ao cabeçalho, os dados transmitidos não são confirmados sendo por isso da responsabilidade das camadas superiores a sua confirmação.

Endereço de origem (IP Address):

Endereço de destino (IP Address): O endereço de destino e o endereço origem das mensagens IP são também conhecidos por endereços Internet (*Internet Address*). Este tipo de endereços ocupa 4 bytes.

Opções: Este campo não é usado em todas as mensagens IP, apenas é usado nas mensagens de controlo e teste da rede. O comprimento deste campo é variável. Na realidade este campo pode ter várias opções começando sempre cada uma delas pelo byte, do código dessa opção. Esse byte com o código da opção tem três campos, *Copy*, *Option Class*, e *Option number*. O campo de *Copy* se estiver a "1" faz com que os gateways ao fragmentarem uma mensagem copiem esta opção para todos os fragmentos. O campo *Option Class* e *Option number* indicam a classe e a opção pretendida.

Padding: Este campo é o último do cabeçalho e não contém nenhuma informação útil, apenas existe para garantir que o número de bits do cabeçalho é múltiplo de 32. Quando o cabeçalho não ocupa um número inteiro de *words* é acrescentado ao cabeçalho um conjunto de bits a zero.

Dados: Neste campo circulam os cabeçalhos e dados das camadas superiores. O comprimento deste campo é variável e depende do tamanho das tramas da camada lógica (ou *MAC-frames*) onde as mensagens IP, viajam através da rede.

Fragmentação

Como cada mensagem IP poderá ter de passar por várias redes “físicas” até alcançar o seu equipamento de destino, pode ser necessário fragmentá-la em unidades mais pequenas, com um comprimento máximo variável (Maximum Transfer Unit - MTU). Cada um dos fragmentos tem o mesmo cabeçalho da mensagem original, apenas é alterado o seu campo (Fragment offset) e o seu campo de dados, passando este fragmento a conter apenas parte dos dados originais. Cada um dos fragmentos será transmitido até ao destino e só depois será desfragmentado pelo equipamento de destino. Se alguns fragmentos se perderem o equipamento de destino desfaz-se dos restantes fragmentos, cabendo aos protocolos superiores da camada OSI o seu reenvio.

8.1.2. Endereço IP

Como já vimos, a Internet é uma rede virtual construída para interligar vários tipos de redes à custa de Routers. É o endereço Internet que permite a este protocolo encaminhar as mensagens através dos vários tipos de redes como se o equipamento emissor estivesse na mesma rede do equipamento de destino. O endereço Internet foi criado para facilitar as decisões de encaminhamento.

Existem três classes de Endereços Internet: cada um deles tem 32 bits e encontram-se organizados da seguinte forma:

tabela 8.1: Classes de Endereços Internet

Classe	Três bits Iniciais	Número de bits para identificar a rede	Número de bits para identificar o equipamento	Máscara da rede
A	0XX	7	24	FF000000
B	10X	14	16	FFFF0000
C	110	21	8	FFFFFF00

Estes endereços são usados nos mensagens-IP. O endereço não só identifica o equipamento destino mas também a rede onde o equipamento se encontra.

Os Routers analisam os bits do endereço relativos à rede de destino para tomarem decisões de encaminhamento.

Se os bits identificadores da rede forem iguais a zero referem-se à própria rede.

Se os bits identificadores da rede do endereço destino forem todos iguais a um, indicam que a mensagem-IP deve ser enviada a todos os equipamentos dessa rede (*broadcast*).

Como o Endereço de Internet permite o encaminhamento das mensagens-IP, a mudança de rede de um equipamento implica a alteração do seu endereço.

8.1.3. Internet Control Message Protocol (ICMP)

Já vimos que o protocolo IP presta um serviço de entrega de mensagens não confirmado, sem estabelecimento de ligação (modo Pacote). Mas se um router ou gateway não consegue reencaminhar ou entregar uma mensagem ele precisa de informar a camada superior para que sejam tomadas as acções correctas.

O protocolo ICMP pode ser usado pelos routers para enviarem a informação de erro ou controlo. As mensagens ICMP viajam dentro do campo de dados das mensagens-IP e são trocadas entre entidades ICMP transparentemente ao utilizador.

A vantagem das mensagens ICMP viajarem dentro das mensagens IP é óbvia quando essas mensagens têm de atravessar várias redes. Cada mensagem ICMP tem o seu formato próprio:

Source Quench Message: Pede ao equipamento emissor que diminua a taxa de envio de pacotes.

Echo Request: O equipamento ou router que receber esta mensagem responde com um Echo Replay o que permite descobrir que o equipamento destino está contactável.

Redirect Message: Permite ao equipamento actualizar a sua tabela de encaminhamento (routing).

8.1.4. Address Resolution Protocol (ARP)

O endereço Internet é atribuído independentemente do endereço físico, no entanto é sempre necessário obter o endereço físico (menos para as mensagens broadcast) para poder enviar as mensagens IP até ao destino, encapsulados em mensagens Ethernet. Por isso, é necessário resolver dinamicamente o endereço IP através do protocolo ARP para obter o endereço físico correspondente. Esta resolução dinâmica tem a vantagem dos equipamentos não precisarem de terem tabelas com a relação entre os endereços físicos e endereços Internet.

Quando um equipamento quer saber o endereço físico de outro, envia um pacote ARP (endereço físico-broadcast) com o endereço Internet do equipamento pretendido. O equipamento que tiver esse endereço IP responde com o seu endereço físico. Cada equipamento tem uma ARP cache que evita a maior parte dos ARP request.

ARP Explained - Address Resolution Protocol
<https://www.youtube.com/watch?v=cn8Zxh9bPio>

Address Resolution Protocol (ARP) Explained
<https://www.youtube.com/watch?v=xTOyZ6TWQdM>

Learn Address Resolution Protocol (ARP) in just 7 Minutes - ARP Tutorial. TCP/IP Explained
<https://www.youtube.com/watch?v=ULpPIVln6nI>

8.1.5. Reverse Address Resolution Protocol (RARP)

Este protocolo faz o inverso do anterior, com base no endereço físico de um recurso determina qual o seu endereço IP.

8.1.3. Routing

O encaminhamento das mensagens IP, ao longo dos equipamentos intermédios, é feito de forma transparente para o equipamento emissor. Como já vimos a rede Internet é formada por muitas redes ligadas entre si através de gateways ou routers.

Existem dois tipos de encaminhamento, o directo e o indirecto:

Encaminhamento Directo: O equipamento destino está na mesma rede física que o equipamento emissor.

Encaminhamento Indirecto: O equipamento destino está noutra rede e os Pacotes-IP têm de passar através de routers ou gateways para alcançarem o equipamento destino.

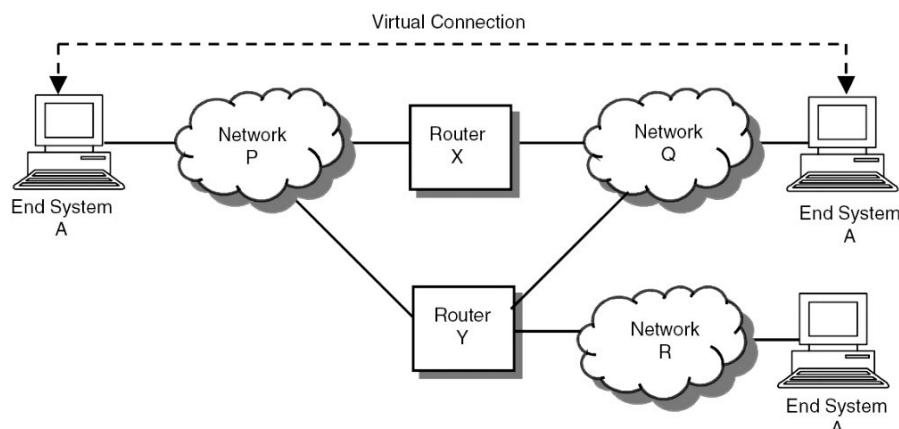


Figura : Routing

O equipamento emissor sabe se o equipamento receptor se encontra na sua rede, ou se precisa de enviar o mensagens-IP para um *Router*, analisando o endereço de destino dessa mensagem. O equipamento emissor extraí do endereço destino a parte relativa à rede (o endereço IP contém o endereço de rede e o endereço da máquina) compara com o endereço da sua rede e sabe se o equipamento de destino está ou não na sua rede (no seu domínio IP). Se o equipamento destino não está na sua rede (encaminhamento indirecto) o emissor determina qual o endereço físico do router para onde deve enviar uma trama contendo a mensagem-IP. Este Router enviará a mensagem-IP para outro Router até que o último Router envia directamente a mensagem-IP (encapsulado no campo de dados de uma mensagem Ethernet por exemplo) para o equipamento de destino.

Tabela de IP-Routing:

Cada equipamento *Router* possui uma tabela deste tipo para saber se deve enviar uma mensagem-IP directamente para o equipamento de destino ou para outro router. Um router ao receber uma mensagem-IP extraí o seu endereço de destino, mais exactamente a parte do endereço relativa à rede de destino, e com base nele fica a saber se pode enviar directamente a mensagem-IP para o equipamento de destino ou para um router intermédio que o encaminhará para o destino.

Hub, Switch or Router? Network Devices Explained (7 min)

https://www.youtube.com/watch?v=Ofjsh_E4HFY

Packet Traveling - How Packets Move Through a Network ??

<https://www.youtube.com/watch?v=rYodcvhh7b8>

Routers, Switches, Packets and Frames

<https://www.youtube.com/watch?v=zhIMLRNY5-4>

8.1.4. Endereços Internet Públicos e Privados

Os endereços privados podem começar por :

10.x.x.x (classe A)

127.16.x.x (classe B)

192.168.1.x.x (classe C)

Estes endereços podem ser atribuídos livremente por um router (domínio IP), como o das nossas casas, aos nossos equipamentos. Desta forma em habitações diferentes podem existir equipamentos com o mesmo IP local.

Mas não podem existir dois IP públicos iguais.

No mesmo domínio privado, na mesma habitação, também não podem existir IP privados iguais, mas em habitações diferentes sim.

8.1.5. Network Address Translation (NAT)

Os equipamentos/programas que implementam estas especificações, habitualmente os routers, conseguem converter os endereços IP privados em endereços IP públicos e vice-versa.

Para isso cada router tem uma tabela interna “NAT forwarding table”.

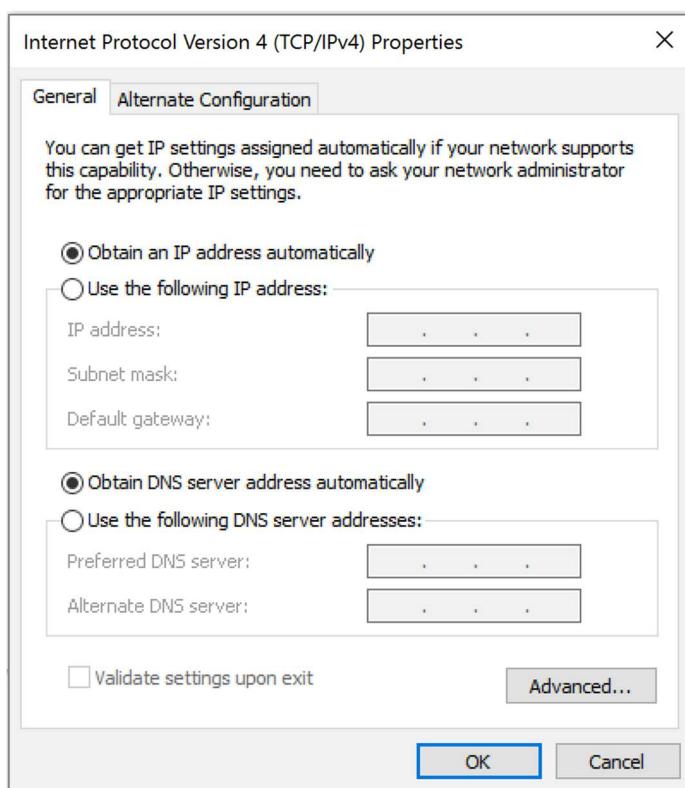
How Network Address Translation Works (10 min)

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

8.1.6. Atribuição estática ou dinâmica de endereços IP aos equipamentos quando se ligam à rede

Num domínio IP pode ser implementado o Dynamic Host Configuration Protocol (DHCP). Nas nossas habitações é o equipamento/router que atribui um IP local para cada um dos equipamentos do seu domínio. Nos nossos computadores também podemos definir/forçar um IP estático específico, nesse caso não ativamos o protocolo DHCP, o nosso computador não inicia uma troca de mensagens com o router para pedir a atribuição dinâmica de um IP.

Podemos definir o IP do router/gateway onde o nosso computador se deve ligar, ou deixar essa atribuição também ao cuidado do protocolo DHCP.



Automatic IP Address Assignment: How DHCP Works
<https://www.youtube.com/watch?v=RUZohsAxPxQ>

8.1.7. Domain Name Service (DNS)

Como converter um URL como o "www.ua.pt.ua" no endereço IP correspondente ?

Só conhecendo o IP do equipamento é que podemos comunicar com ele através da "Internet".

Os equipamentos que permitem esta conversão, usam a especificação DNS.

Esta especificação transcende os conceitos a aprender nesta disciplina, mas podem obter mais informação em:

Inside the Domain Name System (13 min)
<https://www.youtube.com/watch?v=GIZC4Jwf3xQ>

Bibliografia:

IP protocol specification <http://www.rfc-editor.org/rfc/rfc791.txt>

04 - Understanding *Internet Protocol* (50 min)

https://www.youtube.com/watch?v=EkP4Ap_QQHc

[Farrel] Farrel, A. - *The Internet and its protocols: A comparative approach*, Elsevier,

[Blank] Blank, G.A. - *TCP/IP Foundations*, SYBEX, 2004

[Clark] Clark, G., Reynders.D., Wright. E., - *Practical Modern SCADA protocols: DNP3, 60879.5 and related systems*. Newnes – Elsevier, 2004

capítulo 2 – Fundamentals of SCADA communications

capítulo 12 – Ethernet and TCP/IP networks

capítulo 13 – Fieldbus and SCADA communication systems

[Mackay] Mackay, S., et.al. - *Practical Industrial Data Networks: Design, Installation and Troubleshooting*, Elsevier, 2004

capítulo 3 – EIA-232 overview

capítulo 4 – EIA-485 overview

capítulo 7 – Modbus overview

capítulo 13 – Profibus PA/DP/FMS overview

capítulo 15 – Industrial Ethernet overview

capítulo 16 – TCP/IP overview

ARP Explained - Address Resolution Protocol

<https://www.youtube.com/watch?v=cn8Zxh9bPio>

Address Resolution Protocol (ARP) Explained

<https://www.youtube.com/watch?v=xTOyZ6TWQdM>

Learn Address Resolution Protocol (ARP) in just 7 Minutes - ARP Tutorial. TCP/IP Explained

<https://www.youtube.com/watch?v=ULpPIVln6nI>

Hub, Switch or Router? Network Devices Explained (7 min)

https://www.youtube.com/watch?v=Ofjsh_E4HFY

Packet Traveling - How Packets Move Through a Network ??

<https://www.youtube.com/watch?v=rYodcvhh7b8>

Routers, Switches, Packets and Frames

<https://www.youtube.com/watch?v=zhlMLRNY5-4>

How Network Address Translation Works (10 min)

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

Automatic IP Address Assignment: How DHCP Works

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

Inside the Domain Name System (13 min)

<https://www.youtube.com/watch?v=GIZC4Jwf3xQ>



CAPÍTULO

8

TCP

JPSantos
2023

9. TCP - TRANSMISSION CONTROL PROTOCOL

9.1. Conceitos sobre o protocolo TCP

O protocolo *TCP – Transmission Control Protocol*, tal como o protocolo IP, foi desenvolvido pela *DARPA - Defense Advanced Research Projects Agency*.

O protocolo Internet – IP descrito no capítulo anterior fornece um serviço não confirmado de entrega de mensagens, pode mesmo acontecer que as mensagens IP cheguem fora de ordem ou não cheguem de todo ao destino.

Assim, para os programas que necessitem de enviar grandes quantidades de informação com fiabilidade é necessário usar também o protocolo TCP. Este protocolo fornece um serviço fiável de transferência de dados, apesar de usar os serviços prestados pelo protocolo IP.

Para isso, os programas que implementam o protocolo de comunicação TCP, nos equipamentos de origem e destino, estabelecem entre si uma ligação virtual, onde a recepção dos dados é sempre confirmada pelo receptor e permitem a retransmissão automática das mensagens perdidas. Este protocolo também permite que várias aplicações no mesmo equipamento comuniquem através da mesma ligação de rede IP. Para isso as várias ligações de transporte são identificadas pelo número da porta virtual usada. Este protocolo garante que não há pacotes perdidos e que chegam na ordem correcta.

9.1.1. Estabelecimento de uma ligação virtual

Antes de iniciar a transferência de dados os programas que implementam o protocolo TCP nos dois equipamentos, negoceiam o identificador de chamada, o tamanho máximo dos pacotes a transferir, a taxa de transmissão. (*Connection Oriented*)

O equipamento que pretende estabelecer a ligação virtual, envia uma mensagem com um formato específico para o equipamento de destino. Se o equipamento de destino estiver ligado poderá aceitar a ligação virtual respondendo com outra mensagem específica. Estas mensagens são enviadas dentro das mensagens IP (*IP Datagram*), que por sua vez são enviadas para o meio físico de transmissão dentro de mensagens Ethernet (*MAC frame*).

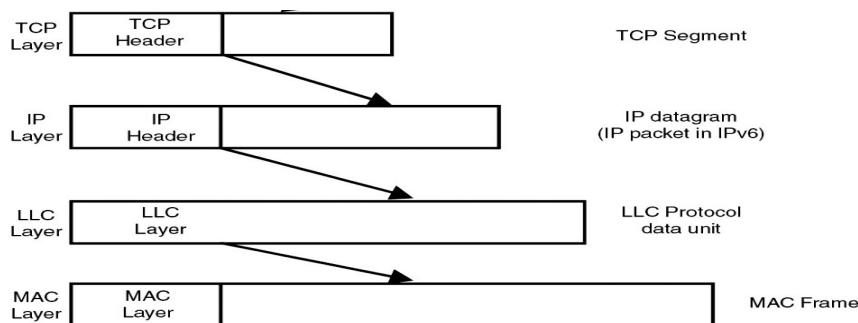


Figura: Encapsulamento

Uma ligação virtual é estabelecida não só entre equipamentos, mas em particular entre dois programas, um em cada equipamento. Por esta razão a mensagem enviada para estabelecer uma ligação: contém os endereços físicos dos equipamentos de origem e de destino (*MAC address*), contém os endereços IP dos equipamentos de destino e de origem (*IP address*), e também contém um número (*TCP port number*) que identifica o tipo de ligação pretendida e consequentemente o tipo de programas (*TCP port number*) que nos dois equipamentos vão enviar e receber essas mensagens.

<i>TCP port Number</i>	<i>Application</i>	<i>Description</i>
21	FTP	File Transfer Protocol
23	Telnet	remote login
25	SMTP	Simple Mail Transfer Protocol
37	Time	Time
42	Nameserver	hostname server
53	Domain	Domain Name Server
79	Finger	Finger
80	http	World Wide Web
103	x400	X.400 messaging service
113	Auth	authentication service

Figura : Portas TCP

9.1.2. Estrutura da mensagem TCP

Os parágrafos seguintes descrevem cada um dos campos da mensagem TCP.

PORTE ORIGEM (16bits)
PORTE DESTINO (16bits)
Nº de SEQUENCIA (16bits)
Nº de CONFIRMAÇÃO (16bits)
OFFSET (4bits)
RESERVADO (4bits)
CÓDIGO (8bits)
JANELA (16bits)
CHECKSUM (16bits)
URGENTE (16bits)
OPÇÕES (24bits)
PADDING (variável)
DATA (variável)

Figura : Estrutura das mensagem TCP

Porta de origem:

Porta de destino:

Estes campos indicam as portas TCP que as entidades dialogantes utilizam. Estas portas indicam às entidades desta camada que serviço é pretendido ou a que aplicação cliente, os dados se destinam (*TCP port number*).

Número de sequência: Como uma mensagem pode ter um comprimento superior ao comprimento máximo de um segmento TCP pode ser necessário fragmenta-la em vários segmentos. Assim é necessário numerar os vários fragmentos de uma mesma mensagem. Não esquecer que o protocolo Internet usado pelo TCP não garante a ordem de chegada.

Número de confirmação: Este número indica qual a posição do último byte recebido pela máquina que envia este fragmento. Assim a máquina a que se destina este segmento fica a saber que os dados por ela antes enviados, foram bem recebidos.

Offset:

Como o cabeçalho dos segmentos TCP tem comprimento variável (o seu campo Opções é variável) é necessário indicar onde começa o campo de dados desse segmento.

Reservado:

Reservado para uso futuro.

Código:

Alguns segmentos transportam dados (tem campo de dados) outros apenas informação de controlo, assim o seu formato varia conforme o

seu código (Tabela 9.3). Desta forma o receptor pode saber qual o tipo de segmento recebido e como o tratar. Existem seis tipos de segmentos:

tabela 9.2: Código dos segmentos TCP

UR G	Segmento com dados urgentes.
AC K	Segmento de confirmação dos dados enviados.
PS H	
RS T	Termina definitivamente a ligação.
SY N	Sincroniza o número de sequência dos fragmentos.
FIN	O emissor já enviou todos os dados.

Janela: Através deste campo (usado nos segmentos de dados e controlo para controlo de fluxo) a máquina que o envia indica quantos segmentos já recebeu correctamente

Checksum: Através deste campo é possível ao receptor confirmar que os dados recebidos estão correctos. O emissor escreve neste campo um inteiro positivo (16 bits). Esse inteiro é calculado pelo emissor com base nos bytes enviados por esta entidade à sua homóloga. Na recepção o mesmo algoritmo é aplicado ao segmento TCP. Se o inteiro produzido for igual ao que veio no segmento, a transmissão foi bem sucedida.

O algoritmo usado para o cálculo do inteiro é aplicado em todos os bytes do segmento TCP (excepto o próprio campo de *Checksum* que considera igual a zero). Além dos bytes do segmento, o algoritmo é também aplicado a um pseudo cabeçalho criado e também enviado nos pacotes IP, juntamente com o segmento TCP.

O pseudo cabeçalho contém os endereços destino e origem (endereços IP), a versão do protocolo TCP e ainda o comprimento do segmento TCP. Este pseudo cabeçalho é criado para que possa ser verificado que o segmento se destina efectivamente àquela máquina.

Urgente: Este campo indica que o segmento onde viaja possui dados urgentes que devem ser retransmitidos (*gateways*) ou processados antes dos restantes dados.

Opções: Neste campo pode ser negociado com o computador destino determinados serviços. Por exemplo este campo pode ser usado para definir qual o comprimento dos segmentos a usar.

Padding: Este campo tem um comprimento variável de forma que o seu comprimento mais o do campo de opções seja múltiplo de 32 bits.

Data: Este campo contém os dados do utilizador desta camada. O seu comprimento é variável e dependente do comprimento dos pacotes IP.

9.1.3. Envio das mensagens TCP

Durante a transferência de dados (em cada canal de transporte) é feito o controlo de fluxo e de erros. Esse controlo é feito através de TPDU (Transmission Protocol Data Unit) internos deste protocolo, trocados entre as duas entidades, transparentemente ao utilizador.

Bibliografia:

[Blank] Blank, G.A. - *TCP/IP Foundations*, SYBEX, 2004

[Clark] Clark, G.,Reynders.D., Wright. E., - *Practical Modern SCADA protocols: DNP3, 60879.5 and related systems*. Newnes – Elsevier, 2004
capítulo 12 – Ethernet and TCP/IP networks

[Mackay] Mackay, S., et.al. - *Practical Industrial Data Networks: Design, Installation and Troubleshooting*, Elsevier, 2004.
capítulo 16 – TCP/IP overview

9.2. VBasic - como configurar e utilizar a interface TCP do computador

Para implementar os protocolos TCP/IP, o *VisualBasic* disponibiliza objetos do tipo *TcpListener* e *TcpClient*, disponíveis nas bibliotecas “System.Net” e “System.Net.Socket”.

```
Imports System.Net  
Imports System.Net.Socket
```

Um objeto do tipo *TcpListener*, neste exemplo com o nome *EsperaLigacaoTCP*, depois de configurado pode aceitar pedidos de ligações TCP/IP que cheguem ao computador vindos da internet e se destinem à sua porta TCP com o número 80.

```
Dim EsperaLigacaoTCP As New TcpListener(80)
```

O objecto *EsperaLigacaoTCP*, do tipo *TcpListener*, tem diversas propriedades (variáveis internas) e métodos/funções:

```
EsperaLigacaoTCP.Start() // o objecto fica à escuta de pedidos de ligação  
EsperaLigacaoTCP.Pending // assume o valor True se existir um pedido de ligação à espera de ser aceite  
EsperaLigacaoTCP.AcceptTcpClient() //aceita a ligação e devolve informação a um objecto do tipo TcpClient
```

Nessa altura, um novo tipo de objecto, do tipo *TcpClient*, neste exemplo com o nome *LigacaoTCP*, recebe a informação relativa à ligação TCP/IP previamente aceite.

```
Dim LigacaoTCP As New TcpClient  
LigaçãoTCP= EsperaLigacaoTCP.AcceptTcpClient()
```

O objecto *LigacaoTCP* , do tipo *TcpClient*, tem diversas propriedades (variáveis internas) e métodos (funções):

```
LigacaoTCP.Connected // assume o valor True enquanto a ligação TCP/IP se mantiver estabelecida  
LigacaoTCP.Available // contém o número de bytes recebidos  
LigacaoTCP.GetStream() // devolve os bytes recebidos a um objecto do tipo NetworkStream
```

Um objecto do tipo *NetworkStream*, contém a referência à ligação TCP, contém um array de bytes que serão enviados ou recebidos por TCP/IP, e um conjunto de métodos que permitem ler e escrever esses bytes. Neste exemplo o objecto *MsgTxRx* é do tipo *NetwokStream*.

```
MsgTxRx  
.Read // retorna os bytes existentes no objecto, por referência, para um array de bytes externo.  
.Write // recebe um array de bytes e envia-o por TCP/IP
```

Como as funções Write e Read do tipo *NetworkStream* não recebem nem devolvem strings, mas sim arrays de bytes, impõem-se saber converter array de bytes, em strings, e vice-versa.

No exemplo seguinte um array de bytes, com os números 48, 49, 50 e 51 é convertido na string “1234” e visualizada na *Textbox1*.

Imports System.Text

```
Dim bb As Byte= 48,49,50,51  
TextBox1.Text= Encoding.Default.GetChars(bb)
```

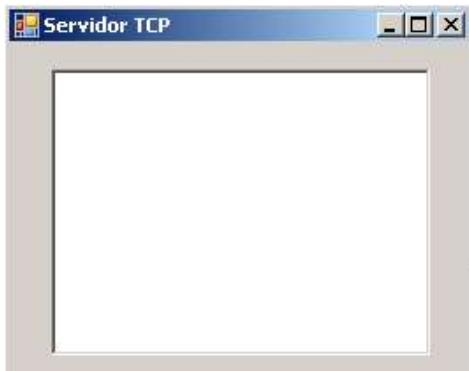
Pelo contrário, para converter uma string num array de bytes

```
Dim bb() As Byte  
Dim ss As String = "1234"  
bb = Encoding.Default.GetBytes(ss)
```

```
Dim EsperaLigaçaoTCP as TcpListener(80)  
EsperaLigaçaoTCP.Start()  
....  
If (AceitaLigaçaoTCP.Pending=1) then  
    Dim LigacaoTCP as TcpClient  
    LigacaoTCP = AceitaLigaçaoTCP.AcceptTcpClient()  
    End if  
....  
If (LigaçaoTCP.Available >0) then  
    Dim MsgTxRx as NetworkStream  
    MsgTxRx = LigaçaoTCP.GetStream  
    ...  
    MsgTxRx.Read(ArrayBytes, 0 , nº bytes para ler)  
    Textbox1.text = Encoding.Default.GetChars(ArrayBytes)  
    End if
```

9.2.1. Exemplo de um servidor TCP/IP

Este servidor aceita automaticamente uma ligação TCP/IP, recebe um conjunto de bytes, e responde com o mesmo conjunto de bytes. De facto, faz o eco da mensagem recebida.



```
Imports System.Net
Imports System.Net.Sockets
Imports System.Text          ' Permite converter Bytes em Caracteres (Tabela ASCII)

Public Class Form1
    Dim EsperaLigacaoTCP As New TcpListener(85) ' Objecto TCP/IP – espera pedido de ligação na porta 85
    Dim LigacaoTCP As New TcpClient           ' Objecto TCP/IP – gere a ligação
    Dim MsgTxRx As NetworkStream            ' Usa um objecto do tipo TcpClient para enviar e receber dados
    Dim ss(1000) As Byte
    Dim nBytes, i As UInteger

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        EsperaLigacaoTCP.Start()           ' Fica à espera de um pedido remoto de ligação TCP
        Timer1.Interval = 500             ' Ativa o timer
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If EsperaLigacaoTCP.Pending = True Then      ' espera pedido de ligacao
            LigacaoTCP = EsperaLigacaoTCP.AcceptTcpClient   ' aceita ligacao
        End If

        nBytes = LigacaoTCP.Available           ' devolve o numero de bytes recebidos
        If nBytes > 0 Then                     ' há bytes para ler

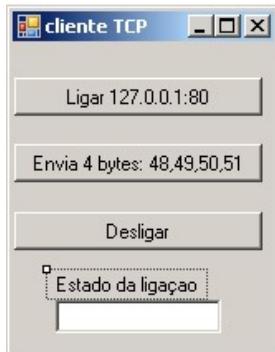
            MsgTxRx = LigacaoTCP.GetStream
            MsgTxRx.Read(ss, 0, nBytes)
            For i = 0 To nBytes
                TextBox1.Text = TextBox1.Text + Chr(ss(i))
            Next i
            TextBox1.Text = Encoding.Default.GetChars(ss)     ' converte-os para caracteres

            ' Enquanto a ligacao TCP/IP está estabelecida, envia mensagem
            MsgTxRx.Write(ss, 0, nBytes)

        End If
    End Sub
End Class          ' Timer1
                    ' Form1
```

9.2.2. Exemplo de um cliente TCP/IP

Esta aplicação estabelece uma ligação TCP/IP com o computador remoto, na porta TCP número 80 do computador com o endereço IP 127.0.0.1 e envia 4 bytes com os valores 48, 49, 50, 51. Pode-se observar o estado da ligação TCP na caixa de texto txtEstadoLigacao.



Pressupõe-se que uma aplicação TCP_servidora, está ativa no computador 127.0.0.1 e à escuta da porta 80, para poder receber e aceitar o pedido de ligação TCP/IP feito pelo programa seguinte:

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Public Class Form1
    Dim ClienteTCP As New TcpClient      ' Gere a ligação TCPIP
    Dim MsgTxRx As NetworkStream        ' Usa um objecto do tipo TcpClient para envia e receber dados

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 500
        Timer1.Enabled = True
    End Sub

    Private Sub BtConnect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
        ClienteTCP.Connect("127.0.0.1", 80)          ' Envia pedido de ligação para o computador remoto
        If ClienteTCP.Connected Then
            MsgTxRx = ClienteTCP.GetStream()        ' Devolve a referencia da ligação TCP, ao objecto MsgTxRx
        End If
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        txtEstadoLigacao.Text = ClienteTCP.Connected      ' Devolve o valor True se a ligação está ativa
    End Sub

    Private Sub BtEnviar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ....
        Dim bb() As Byte = {48, 49, 50, 51}           ' Cria um array de bytes, "bb", com 4 valores: 48, 49, 50, 51
        MsgTxRx.Write(bb, 0, 4)                      ' Envia para o computador remoto o array de bytes "bb"
    End Sub

    Private Sub BtDesligar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
        ClienteTCP.Close()                         ' Termina a ligação TCP e o objecto do tipo TcpClient
        ClienteTCP = New TcpClient                 ' Cria um novo objecto do tipo TcpClient
    End Sub

End Class      ' Form1
```

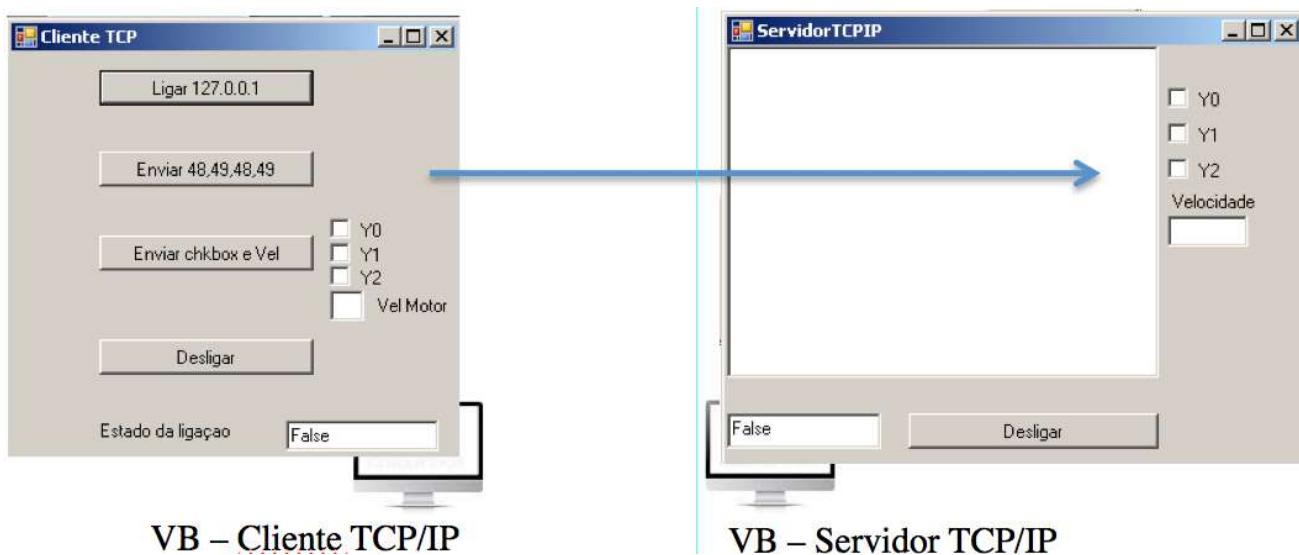


Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

9.3.Trabalho prático nº 7 – TCPIP comunicação entre computadores

Neste trabalho pretende-se enviar mensagens de texto entre dois programas Windows, desenvolvidos em VBasic. Um dos programas atua como cliente TCPIP e o outro como servidor TCPIP (ver Figura). Depois do programa servidor estar ativo, o programa cliente pode estabelecer uma ligação TCPIP entre ambos. Enquanto a ligação estiver estabelecida ambos os programas podem trocar dados entre si.

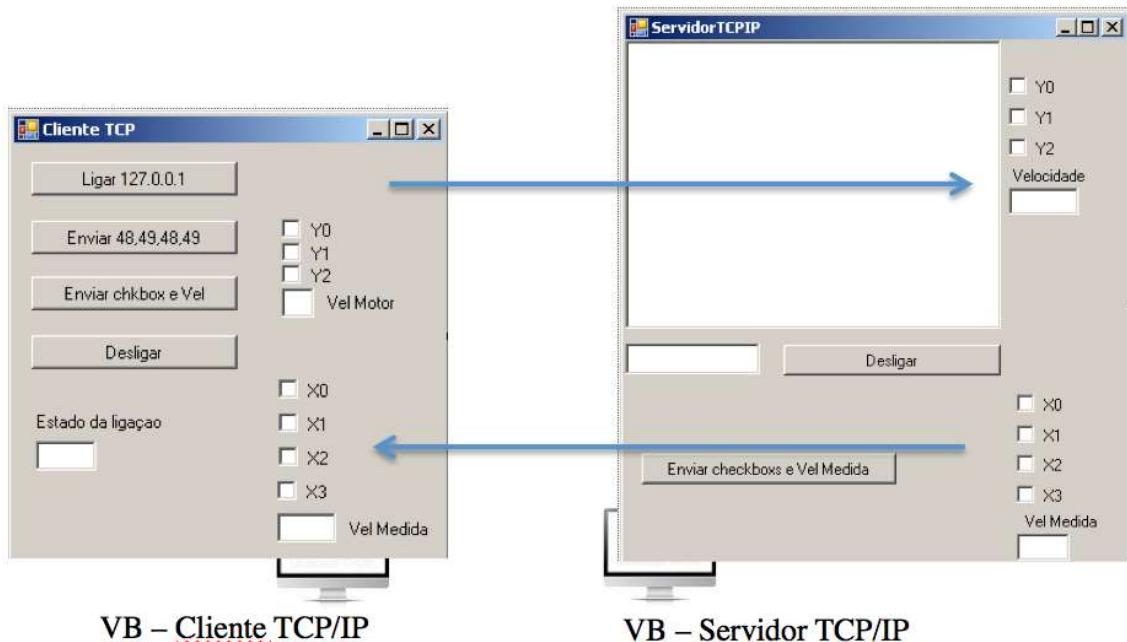


Pretende-se apresentar objetos do tipo TcpClient, TcpListener, e NetworkStream. Estes objetos são necessários para desenvolver em VBasic programas Windows que atuem como clientes e servidores TCPIP, capazes de transmitir dados através da Internet.

9.3.1. Descrição do trabalho

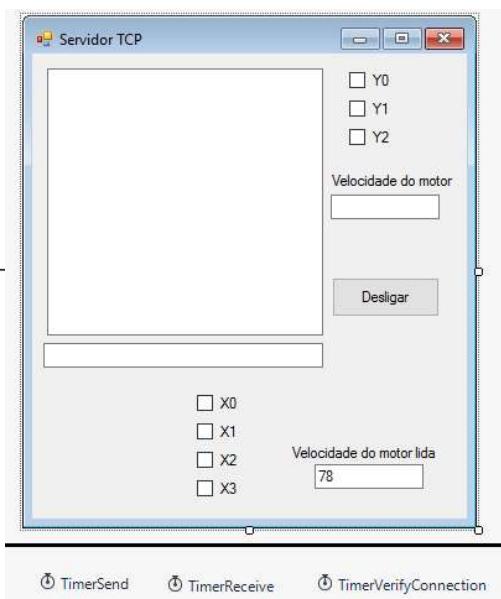
A partir do programa cliente (lado esquerdo da figura) pretende-se selecionar as checkbox Y0, Y1, Y2 e escrever a velocidade na caixa de texto “VelMotor”, quando o utilizador premir o botão “Enviar ChkBox e Vel” esses dados devem ser enviados para o programa servidor, visualizados nas checkbox e na caixa de texto “Velocidade” do servidor. Analise e implemente os exemplos apresentados no final deste documento.

Na aplicação servidora, crie 4 checkbox (X0,X1,X2,X3) e uma caixa de texto (VelMedida). Enquanto a ligação TCP/IP estiver estabelecida (pela aplicação Cliente), o servidor deve enviar, de segundo a segundo, o estado das checkbox X0,X1,X2,X3, e a VelMedida para a aplicação cliente. A aplicação Cliente deve mostrar ao utilizador o estado das checkbox, e a velocidade definida na aplicação servidora.



9.3.2. Exemplo de um Servidor TCPIP

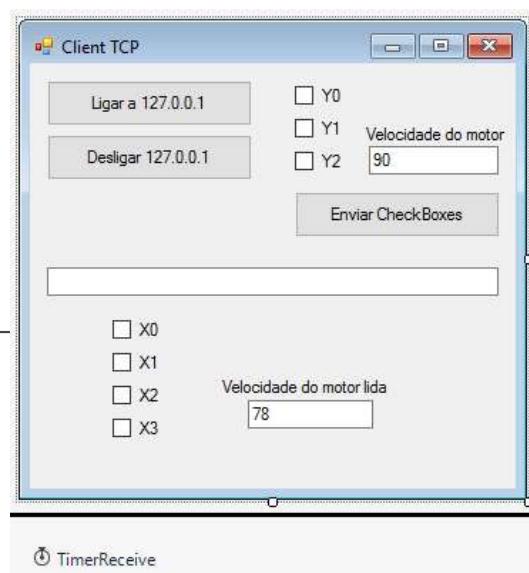
```
...II_2017-2018\Pratica\Aula8\TCPserver\TCPserver\Form1.vb
1 Imports System.Net
2 Imports System.Net.Sockets
3
4 Public Class Form1
5     'The ip address of the server
6     Dim local_address As IPAddress = IPAddress.Parse("127.0.0.1")
7
8     'A server object will accept connection requests
9     Dim server As New TcpListener(local_address, 81)
10
11    'A connection object will receive data
12    Dim connection As New TcpClient
13
14    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
15        'Start the TCP server. Will wait for clients to connect
16        server.Start()
17    End Sub
18
19    Private Sub TimerReceive_Tick(sender As Object, e As EventArgs) Handles TimerReceive.Tick
20
21        'Accept a connection if there is a pending request
22        If server.Pending() = True Then
23            connection = server.AcceptTcpClient()
24        End If
25
26        If connection.Connected Then
27            Dim message_size As Integer = connection.Available
28
29            'Read message if size is > 0
30            If message_size > 0 Then
31
32                'Copy stream to byte array buffer
33                Dim message_in_stream As NetworkStream = connection.GetStream()
34                Dim buffer(5000) As Byte 'a buffer to copy the received data
35                message_in_stream.Read(buffer, 0, message_size)
36
37                'Convert byte array buffer to string message_in
38                Dim message_in As String = ""
39                Dim i As Integer
40                For i = 0 To message_size - 1 'copy from buffer to message_in
41                    message_in = message_in + Chr(buffer(i))
42                Next i
43
44                'Copy to textbox
45                TextBoxReceber.Text = message_in + vbCrLf + TextBoxReceber.Text
46
47                'Process message to get values for Y0, Y1, Y2 and Level
48                CheckBoxY0.Checked = Mid(message_in, 1, 1)
49                CheckBoxY1.Checked = Mid(message_in, 2, 1)
50                CheckBoxY2.Checked = Mid(message_in, 3, 1)
51                TextBoxVelocidadeMotor.Text = Asc(Mid(message_in, 4, 1))
52            End If
53        End If
54    End Sub
```



```
55
56     Private Sub TimerVerifyConnection_Tick(sender As Object, e As EventArgs) Handles TimerVerifyConnection.Tick
57         TextBoxEstadoLigacao.Text = connection.Connected
58     End Sub
59
60     Private Sub TimerSend_Tick(sender As Object, e As EventArgs) Handles TimerSend.Tick
61         If connection.Connected = True Then
62             'Declare a byte array and set it according to the state of X0 to X3
63             Dim buffer(5) As Byte
64             buffer(0) = CheckBoxX0.CheckState + 48
65             buffer(1) = CheckBoxX1.CheckState + 48
66             buffer(2) = CheckBoxX2.CheckState + 48
67             buffer(3) = CheckBoxX3.CheckState + 48
68             buffer(4) = CInt(TextBoxVelocidadeMotorLida.Text)
69
70             'Now send the array
71             Dim message_out_stream As NetworkStream
72             message_out_stream = connection.GetStream()
73             message_out_stream.Write(buffer, 0, 5)
74         End If
75     End Sub
76 End Class
```

9.3.3. Exemplo de um Cliente TCPIP

```
...II_2017-2018\Pratica\Aula8\TCPClient\Form1.vb
1 Imports System.Net
2 Imports System.Net.Sockets
3
4 Public Class Form1
5     'The ip address of the server
6     Dim ip_address As IPAddress = IPAddress.Parse("127.0.0.1")
7
8     'A client object will request connection req
9     Dim client As New Sockets.TcpClient
10
11    Private Sub ButtonLigar_Click(sender As Object, e As EventArgs) Handles ButtonLigar.Click
12        client.Connect(ip_address, 81)
13    End Sub
14
15    Private Sub TimerReceive_Tick(sender As Object, e As EventArgs) Handles TimerReceive.Tick
16        'Update connection state
17        TextBoxEstadoLigacao.Text = client.Connected
18
19        If client.Connected Then
20            Dim message_size As Integer = client.Available
21
22            'Read message if size is > 0
23            If message_size > 0 Then
24                'Copy stream to byte array buffer
25                Dim message_in_stream As NetworkStream = client.GetStream()
26                Dim buffer(5000) As Byte 'a buffer to copy the received data
27                message_in_stream.Read(buffer, 0, message_size)
28
29                'Convert byte array buffer to string message_in
30                Dim message_in As String = ""
31                Dim i As Integer
32                For i = 0 To message_size - 1 'copy from buffer to message_in
33                    message_in = message_in + Chr(buffer(i))
34                Next i
35
36                'Process message to get values for X0 to X3
37                CheckBoxX0.Checked = Mid(message_in, 1, 1)
38                CheckBoxX1.Checked = Mid(message_in, 2, 1)
39                CheckBoxX2.Checked = Mid(message_in, 3, 1)
40                CheckBoxX3.Checked = Mid(message_in, 4, 1)
41                TextBoxVelocidadeMotorLida.Text = Asc(Mid(message_in, 5, 1))
42            End If
43        End If
44    End Sub
45
46    Private Sub ButtonDesligar_Click(sender As Object, e As EventArgs) Handles ButtonDesligar.Click
47        client.Close()
48        client = New Sockets.TcpClient
49    End Sub
50
51    Private Sub ButtonEnviarCheckBox_Click(sender As Object, e As EventArgs) Handles ButtonEnviarCheckBox.Click
52        If client.Connected = True Then
```



```
53      'Declare a byte array and set it according to the state of X0 to X3
54      Dim buffer(5) As Byte
55      buffer(0) = CheckBoxY0.CheckState + 48
56      buffer(1) = CheckBoxY1.CheckState + 48
57      buffer(2) = CheckBoxY2.CheckState + 48
58      buffer(3) = CInt(TextBoxVelocidadeMotor.Text)
59
60      'Now send the array
61      Dim message_out_stream As NetworkStream
62      message_out_stream = client.GetStream()
63      message_out_stream.Write(buffer, 0, 4)
64  End If
65 End Sub
66 End Class
```



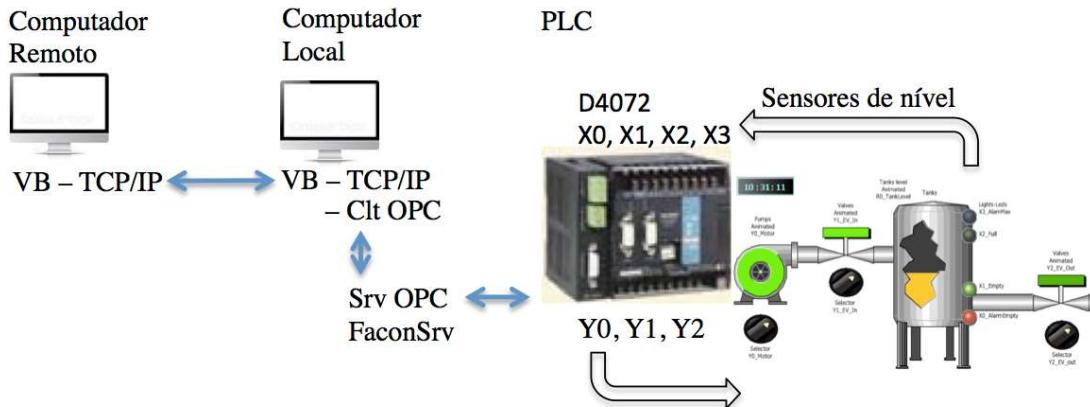
Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

Trabalho prático N° 8B
Supervisão e controlo remoto de um PLC
Via TCP/IP e OPC

Trabalho prático nº 8 - TCP/IP e OPC, Supervisão e controlo remoto do PLC

Neste trabalho (uma semana), pretende-se que um utilizador em qualquer ponto do mundo possa comunicar com um PLC e, dessa forma, controlar e monitorizar remotamente um processo (ex. reservatório de água).



Nos trabalhos anteriores, o controlo e a supervisão remota do PLC não podia exceder algumas centenas de metros, quando era utilizada uma ligação Rs232 entre o PLC e o computador local a partir do qual se efetuava a supervisão e o controlo.

Neste trabalho serão utilizados dois computadores e um PLC (figura):

- Um dos computadores, o **computador local**, encontra-se próximo do PLC e comunica com o PLC através de uma **ligação Rs232**. A comunicação Rs232 pode ser conseguida utilizando localmente o controlo “SerialPort” ou como no último trabalho, utilizando o servidor OPC (**FaconSrv**).
- O segundo computador, **computador remoto**, utiliza uma **ligação TCP/IP** para comunicar com o computador local, e possui a interface de supervisão e controlo remoto do PLC (Reservatório de água).

Devem ser desenvolvidas duas aplicações em VBasic:

- A **aplicação local**, desenvolvida em VBasic, deverá utilizar um objecto **TCPListener** (Servidor) para receber ligações TCP/IP e um objecto do tipo cliente “**FaconSrv**” para comunicar com o PLC.
- A **aplicação remota** utiliza um objecto do tipo **TCPClient** para comunicar com a aplicação local por TCP/IP, permitindo a supervisão e o controlo remoto das variáveis do processo.

A aplicação remota deve ter uma interface que permita **visualizar o estado dos sensores de nível** do reservatório da figura, e o **estado das saídas**: o estado do motor e das electroválvula de entrada e de saída (figura seguinte).

Tal como nos trabalhos anteriores, um motor “**Y0_Motor**” e uma electroválvula de entrada “**Y1_EV_In**” permitem encher o reservatório.

Acoplado ao reservatório existem **4 sensores de nível**:

- “**X0_AlarmEmpty**”,
- “**X1_Empty**”,
- “**X2_Full**” e
- “**X3_AlarmMax**”.

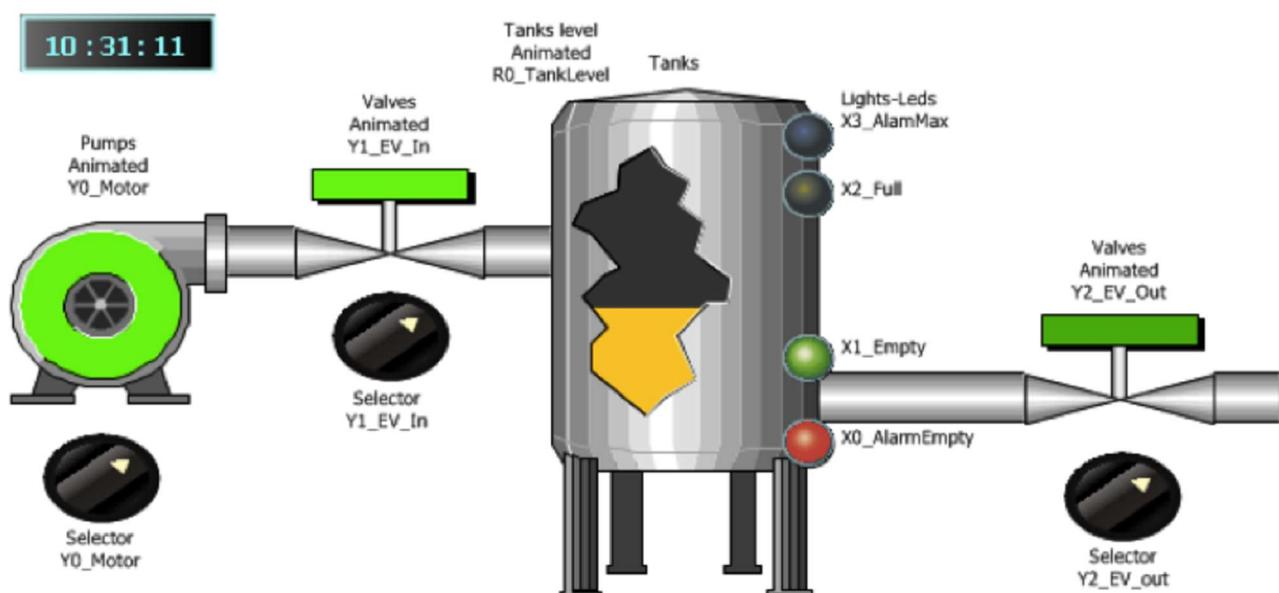
Assume-se que os sensores ficam ativos quando detectam água fazendo com que o PLC, na entrada digital respectiva, receba 24 V.

Existe uma electroválvula de saída “**Y2_EV_Out**” que permite a saída da água do reservatório se o nível de água no reservatório for superior ao nível “**X0_AlarmEmpty**”.

O motor e a electroválvula de entrada só devem ser acionados se o nível de água for inferior ao nível “**X1_Empty**” e devem ser desligados se o nível de água igualar ou exceder o nível “**X2_Full**”.

Devem ser gerados alarmes no écran do computador remoto se o nível de água atingir ou exceder o nível “**X3_AlarmMax**” ou passar a baixo do nível “**X0_AlarmEmpty**”.

As saídas digitais Y0, Y1 e Y2 do PLC permitem, respectivamente: ligar o motor de entrada, a electroválvula de entrada e a electroválvula de saída.



Conhecimentos a adquirir

Introdução às comunicações TCP/IP e a realização de Servidores/Cientes.

Familiarização com os objetos de comunicação TCP/IP e com o servidor OPC :

- TcpClient
- TcpListener
- NetworkStream
- FaconSrv

Importante:

- Neste trabalho o PLC não tem programa, todo o controlo e supervisão é efectuado a partir do computador remoto/VBasic.
- O trabalho será avaliado por questionário individual, na semana seguinte à entrega do mesmo.
- Deve enviar por e-mail os programas desenvolvidos e o(s) documento(s) escrito(s) sob pena da nota obtida no questionário não ser considerada.

Bibiografia

- Slides e apontamentos sobre comunicações TCP/IP, com exemplos em VisualBasic
- [Steve 2004] **Practical Industrial Data Networks, Design, Installation and Troubleshooting**, Mackay Steve et.al , Elsevier, 2004.
- [Halsall] **Data Communications, Computer Networks and OSI**, Fred Halsall, Addison-Wesley. Cota:681.324G.25
- [Rosen 2003] **Web Application Architecture, Principles, Protocols and Practices**, Leon Shklar, Richard Rosen, John Wiley & Sons, 2003

Normas TCP / IP

IP protocol specification <http://www.rfc-editor.org/rfc/rfc791.txt>

TCP protocol specification <https://www.rfc-editor.org/rfc/rfc793.txt>

TCP/IP tutorial <https://tools.ietf.org/html/rfc1180>

01 - Understanding Local Area Networking (60 min)

LAN: Data, nodes, Client, Server, Peer, Net adapter, hub, switch, router, meio, Transport protocol, Largura de banda, wireless access point, serial, data rate transfer, IP address, Virtual LAN, Topologia, Ethernet, Frames, Tipos de Servidores

<https://www.youtube.com/watch?v=t9TmvFvYfWw>

04 - Understanding Internet Protocol (50 min)

IP address: classes, loopback, Binary to decimal, public and private address, DNS, Gateway, NAT, IPv4, IPv6, subnetting

https://www.youtube.com/watch?v=EkP4Ap_OOHc

OSI Model Explained (animated)

https://www.youtube.com/watch?v=O_rsqVtaloI

Hub, Switch or Router? Network Devices Explained (7 min)

https://www.youtube.com/watch?v=Ofjsh_E4HFY

How Network Address Translation Works (10 min)

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

Automatic IP Address Assignment: How DHCP Works

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

Inside the Domain Name System (13 min)

<https://www.youtube.com/watch?v=GIZC4Jwf3xQ>



CAPÍTULO

9

HTTP

JPSantos
2023

10. HTTP – HiperText Transfer Protocol

O protocolo *HTTP* é a pedra angular da WEB. Este protocolo, juntamente com a linguagem *HTML*, foi proposto por Tim Berners-Lee enquanto investigava no *CERN*. De facto Tim Berners propôs um sistema de gestão de informação baseado em Hipertexto. Na altura o termo *World Wide Web -WWW* ainda não existia, mas o sistema proposto por Tim Berners já definia:

1. Uma linguagem para formatar documentos de texto (*HTML – HyperText Markup Language*). Esta linguagem e a sua sintaxe são apresentadas no capítulo seguinte.
2. Um protocolo para transferir esses documentos entre equipamentos (*HTTP – Hyper Text Transfer Protocol*).
3. Um esquema de endereços para identificar e aceder a esses recursos na rede (*URL- Uniform resource locator*).

Convém relembrar que quando Tim Berner propôs o protocolo HTTP já existia a Internet, já existiam servidores e clientes *FTP – File Transfer Protocol* e por isso já era possível transferir documentos entre equipamentos, mas não era fácil localizá-los, nem existia uma forma fácil de um desses documentos fazer uma referência a outro documento (*Hiperligação*). Além disso a forma como cada documento estava formatado e guardado em disco variava de processador de texto para processador de texto.

10.1. Conceitos sobre HTTP

O protocolo **HTTP** define um conjunto de interacções entre as aplicações clientes (*Browsers WEB*) e as aplicações servidoras (*Servidores WEB*) e define a sintaxe de um conjunto de mensagens que todos os clientes e servidores **HTTP** reconhecem e sabem processar.

10.1.1. Localização dos recursos na Internet

Tim Berners propôs uma forma de identificar e localizar um documento, um recurso, na Internet e chamou-lhe **URL – Uniform Resource Locator**, mais tarde passou a chamar-se **URI - Uniform Resource Identifier**.

De uma forma simples e incompleta, podemos dizer que o URL não é mais que um texto, composto pelo nome do computador de destino e pela localização do documento no equipamento de destino (ex. no disco duro).

O URL tem a estrutura seguinte:

Protocolo://[computador:[tcpport number]]/localização do documento dentro do computador de destino
[? Parâmetros do pedido] [# ancore]

Alguns exemplos de URLs

http://www.ua.pt/index.html
ftp://...../.....

10.1.2. Interacções HTTP

A interacção entre programas **HTTP** é do tipo “*cliente – servidor*”. Isto significa que um programa (*browser web*) toma a iniciativa de enviar uma mensagem **HTTP** com um pedido, ao passo que o outro programa “limita-se” a processar e a responder a esse pedido com uma mensagem **HTTP** de resposta (*servidor web*).

1. Pedido de ligação

No entanto, antes de ser possível fazer pedidos HTTP, ou seja, antes de enviar mensagens HTTP, a aplicação cliente tem de estabelecer uma ligação TCP com a aplicação remota.

Na figura seguinte, através de um browser Web, o utilizador pretende aceder a um documento remoto com o seguinte URL:

<http://localhost/index.html>

Através do URL, o browser sabe que deve usar o protocolo http e tem de estabelecer uma ligação TCP com o computador “localhost”

2. Envio de uma mensagem HTTP para o servidor e a mensagem de resposta

Depois da ligação TCP estar estabelecida, o browser pede ao servidor remoto o documento index.html, enviando para isso uma mensagem HTTP do tipo GET:

GET /index.html HTTP/1.1

.....

....

Quando o *servidor WEB* consegue processar a *mensagem HTTP* descrita na figura, gera uma mensagem de resposta começada por “*HTTP/1.1 200 OK*”, acede ao documento pedido, copia o seu conteúdo e acrescenta esse conteúdo à mensagem de resposta.

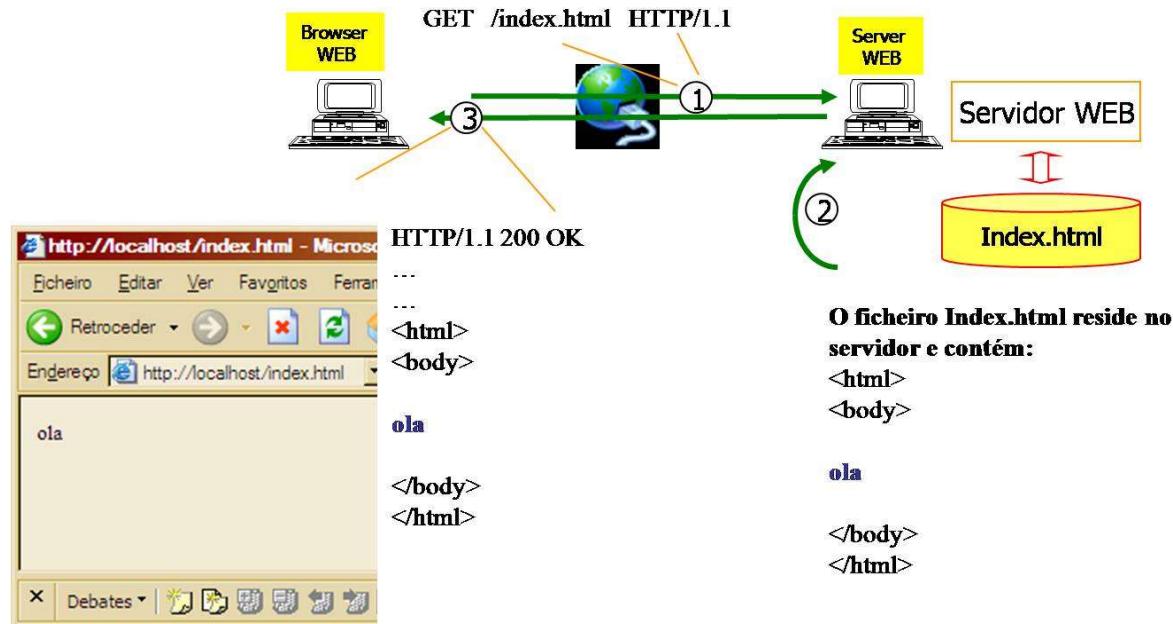


Figura: Troca de mensagens HTTP

3. Fim de ligação

Depois da troca de mensagens http tanto o browser como o servidor Web podem terminar a ligação TCP.

10.1.3. Mensagens HTTP

Depois da ligação TCP ter sido previamente estabelecida o browser Web pode enviar um de vários tipos de pedidos para o servidor, por exemplo: GET, HEAD, POST, etc.

GET

As mensagens do tipo GET levam o servidor a responder com informações sobre o documento pedido, incluindo o seu conteúdo.

Na mensagens GET o cliente (ex. o Browser) pode enviar também dados para o servidor, neste exemplo os “parâmetros URL” x=10 e y=20 :

GET /index.php?x=10&y=20

HEAD

O browser pode pedir ao servidor para lhe enviar apenas informação sobre as características de um documento, por exemplo, pedir-lhe que confirme se esse documento existe no servidor, mas sem pedir o seu conteúdo.

POST

O browser pode enviar um pedido de actualização de um documento no servidor. Neste caso a mensagem enviada pelo browser é do tipo POST e contém também os novos dados que o servidor irá utilizar para actualizar o documento ou executar algumas acções.

Respostas do servidor

Consoante o tipo de pedido efectuado pelo browser e dependendo da forma como o servidor foi capaz de reconhecer e executar esse pedido, o servidor Web responde com um de vários códigos possíveis

HTTP/1.1 código de resposta

.....

.....

Código de resposta =	"200" ; OK
	"201" ; Created
	"202" ; Accepted
	"204" ; No Content
	"301" ; Moved Permanently
	"302" ; Moved Temporarily
	"304" ; Not Modified
	"400" ; Bad Request
	"401" ; Unauthorized
	"403" ; Forbidden
	"404" ; Not Found
	"500" ; Internal Server Error
	"501" ; Not Implemented
	"502" ; Bad Gateway
	"503" ; Service Unavailable

10.2. VBasic – HTTP

Um *ServidorWEB* troca *Mensagens HTTP* com um, ou vários, *Browsers*, através da Internet. De facto a Internet é usada por várias aplicações para transmitirem as suas mensagens através de todo o mundo. Alguns exemplos são os *ClienteFTP/ServidorFTP*, *Telnet/ServidorTelnet*, e o *Email/Servidor Email*. Usa-se habitualmente a expressão Internet mas de facto além do *Internet Protocol IP* é também usado, simultaneamente, o *Transmission Control Protocol TCP*.

O computador de destino, onde reside o servidor, ao receber uma dessas mensagens pela Internet, com base no seu “*TCP port number*”, entrega-a à aplicação receptora mais adequada, pois só esta a saberá processar correctamente.

De uma forma simplista podemos dizer que um *ServidorWEB* mais não é que um *ServidorTCP/IP* que recebe e envia páginas WEB (ex. mensagens de texto escritas em HTML) para um *BrowserWEB*. Entre outros, os programas *Word*, *Power Point*, e *Excel* permitem gravar documentos num formato *HTML* criando dessa forma páginas WEB. Estes programas permitem também incluir num documento referências a outros documentos (“Links”).

Utilizando objetos do tipo *TcpClient*, *TcpListener*, e *NetworkStream* é possível desenvolver clientes e servidores Web em *VBasic*. A figura seguinte ilustra os métodos dos objetos *VBasic* que devem ser utilizados para implementar um cliente ou um servidor WEB. Pode também observar a sequência das mensagens *HTTP* trocadas entre o cliente e o servidor WEB.

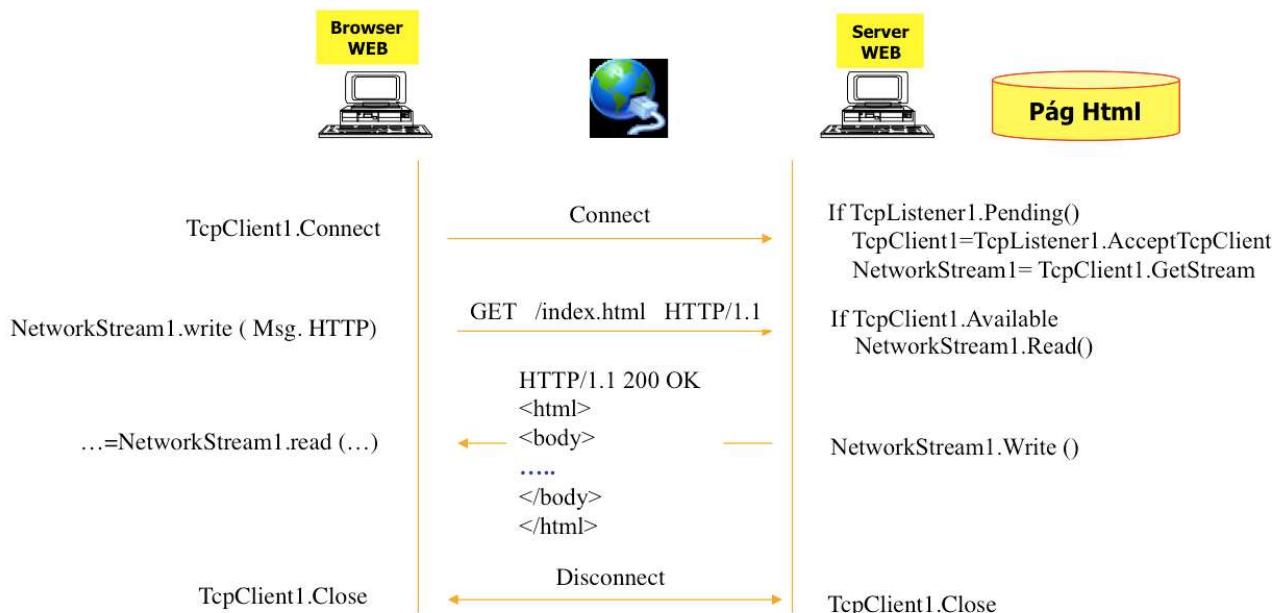
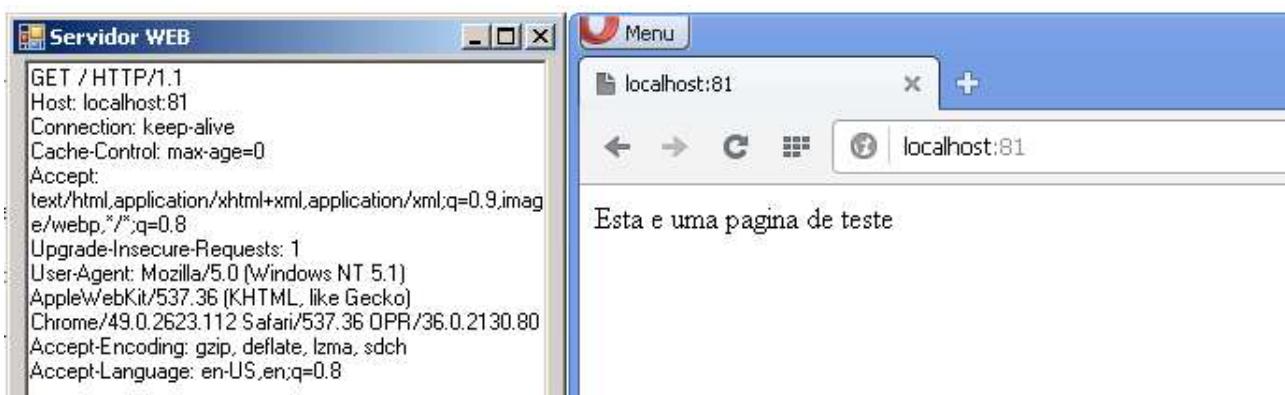


Figura : Troca de mensagens HTTP usando os objetos *TcpClient*, *TcpListener*, e *NetworkStream*

10.2.1. Exemplo – Servidor WEB elementar

Tendo em mente o desenvolvimento de um *ServidorWEB*, vamos começar por utilizar o servidor TCP/IP desenvolvido no capítulo anterior para receber pedidos vindos de um *BrowserWEB* e analisar o tipo de mensagens por ele enviado. Com base nessa análise vamos adaptar o servidor TCP/IP para criar um *ServidorWEB*.

Esta aplicação aceita pedidos de ligação vindos de um *BrowserWEB* e envia-lhe uma página WEB inicial.



```
Dim LocalIP As IPAddress = IPAddress.Parse("127.0.0.1")
Dim EsperaLigacaoTCP As New TcpListener(LocalIP, 80)

Imports System.Net
Imports System.Net.Sockets ' permite usar TcpClient, TcpListener, NetworkStream
Imports System.IO ' manipulação de ficheiros em disco
Imports System.Text ' encoding converte array bytes em string

Public Class Form1
    Dim EsperaLigacaoTCP As New TcpListener(81) 'espera pedidos de ligação TCP/IP
    Dim LigacaoTCP As New TcpClient 'controla a ligação TCP/IP
    Dim Mensagem As NetworkStream 'recebe ou envia array de bytes, usando o TcpClient
    Dim carateres(2000) As Byte
    Dim nCaracteres As Integer
    Dim textorecebido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 1000
        Timer1.Enabled = True
        EsperaLigacaoTCP.Start() 'começa à escuta da port 81
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        Dim i As Integer

        If EsperaLigacaoTCP.Pending() = True Then ' Há ligação à espera de ser aceite ?
            LigacaoTCP = EsperaLigacaoTCP.AcceptTcpClient() ' aceita ligação TCP/IP
            Mensagem = LigacaoTCP.GetStream() ' Le msg recebida no formato NetworkStream
        End If

        If LigacaoTCP.Connected = True Then
            nCaracteres = LigacaoTCP.Available ' há bytes recebidos ?
        End If

        If nCaracteres > 0 Then
            Mensagem.Read(carateres, 0, nCaracteres) 'Le os bytes recebidos
        End If
    End Sub
End Class
```

```

    ' textorecebido = Encoding.Default.GetChars(carateres)
    For i = 0 To nCaracteres - 1
        textorecebido = textorecebido + Chr(carateres(i))
    Next i

    TextBox1.Text = TextBox1.Text + textorecebido + vbCrLf

    ' Envio da pági html para o Browser
    Dim ss As String = "<html>Esta é uma pagina de teste</html>"
    Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss)
    Mensagem.Write(arraybytes, 0, Len(ss))

    LigacaoTCP.Close()      ' O Protocolo HTTP prevê o fim da ligação TCP
    LigacaoTCP = New TcpClient

    nCaracteres = 0
    textorecebido = ""
End If

End Sub

End Class

```

De facto a mensagem HTTP de resposta é mais complexa. Além do documento HTML, começado por <html> e terminado por </html>, a mensagem HTTP começa pelo seu cabeçalho, por exemplo:

*HTTP/1.1 200 OK
Date: Mon, 20 Nov 2017 00:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 20 Jul 2017 19:15:56 GMT
Content-Length: 67
Content-Type: text/html
Connection: Closed*

```

<html>
<body>

Esta é uma pagina de teste

</body>
</html>

```

*HTTP/1.1 200 OK
Date: Mon, 20 Nov 2017 00:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 20 Jul 2017 19:15:56 GMT
Content-Length: 67
Content-Type: text/html
Connection: Closed*

```

<html>
<body>

Esta é uma pagina de teste

</body>
</html>

```

10.2.2. Exemplo – Servidor WEB com duas páginas HTML

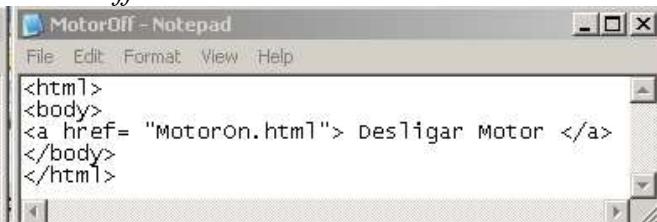
Neste exemplo, o servidor WEB, desenvolvido em VBasic, deve identificar o nome do documento pedido pelo Browser, abrir o ficheiro em disco e enviar o seu conteúdo para o Browser. O documento MotorOn.html tem uma hiperligação para o documento MotorOff.html e vice-versa. Dessa forma, quando o utilizador, no Browser, “clicar” na hiperligação, pede ao servidor WEB o outro documento HTML.

MotorOn.html



```
<html>
<body>
<a href= "Motoroff.html"> Ligar Motor </a>
</body>
</html>
```

MotorOff.html



```
<html>
<body>
<a href= "Motoron.html"> Desligar Motor </a>
</body>
</html>
```

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 70

```
<html>
<body>
<a href= "MotorOff.html"> LigarMotor </a>
</body>
</html>
```

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 72

```
<html>
<body>
<a href= "MotorOn.html"> DesligarMotor </a>
</body>
</html>
```

A partir do Browser, deve pedir ao servidorWEB, os documentos:

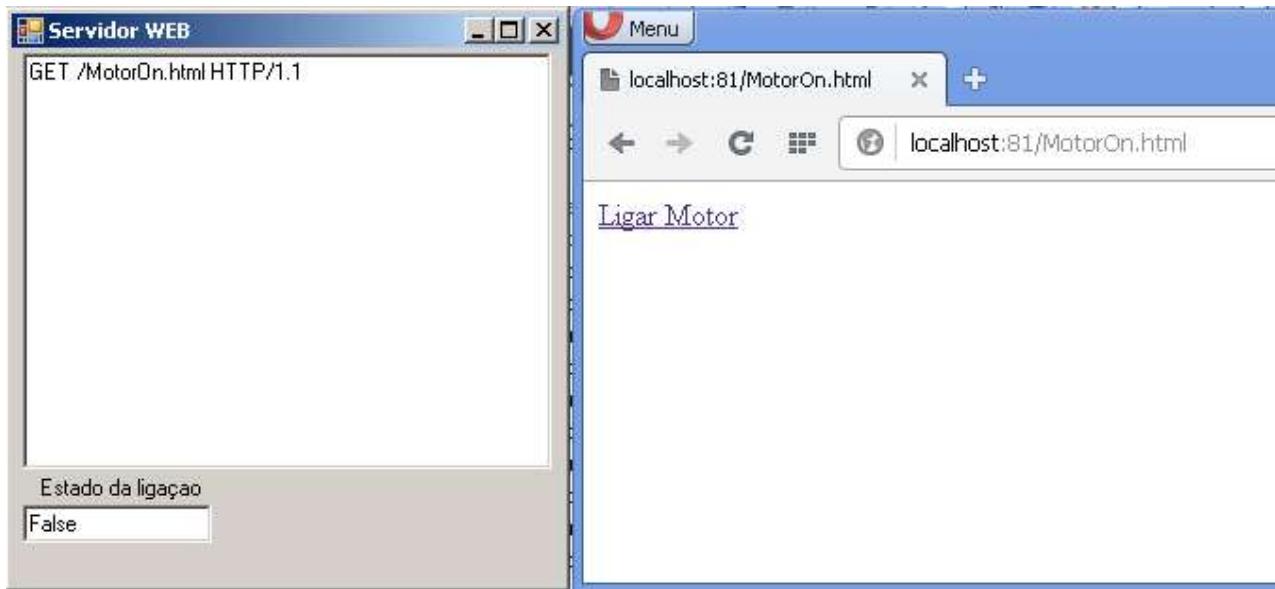
<http://localhost/MotorOn.html>

<http://localhost/MotorOff.html>

O Browser estabelece uma ligação TCPIP com o ServidorWeb e envia-lhe a mensagem HTTP com o nome do documento HTML pretendido:

GET /MotorOn.html ... ou GET /MotorOff.html ...

Como a mensagem HTTP contém o nome do documento HTML pedido pelo utilizador/Browser, o ServidorWEB lê o ficheiro pedido e envia o seu conteúdo para o Browser.



```

Imports System.Net
Imports System.Net.Sockets ' permite usar TcpClient, TcpListener, NetworkStream
Imports System.IO ' manipulacao de ficheiros em disco
Imports System.Text ' encoding converte array bytes em string

Public Class Form1
    Dim EsperaLigacaoTCP As New TcpListener(81) 'espera pedidos de ligacao TCP/IP
    Dim LigacaoTCP As New TcpClient 'controla a ligacao TCP/IP
    Dim Mensagem As NetworkStream 'recebe ou envia array de bytes, usando o TcpClient
    Dim carateres(4000) As Byte
    Dim nCaracteres As Integer
    Dim textorecebido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 1000
        Timer1.Enabled = True
        EsperaLigacaoTCP.Start() ' começa à escuta da port 81
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If EsperaLigacaoTCP.Pending() = True Then ' Há ligacao à espera de ser aceite
            LigacaoTCP = EsperaLigacaoTCP.AcceptTcpClient() ' aceita ligacao TCP/IP
            Mensagem = LigacaoTCP.GetStream() ' Le msg recebida no formato NetworkStream
        End If

        txtEstadoLigacao.Text = LigacaoTCP.Connected ' retorna o estado da ligacao TCP

        If LigacaoTCP.Connected = True Then
            nCaracteres = LigacaoTCP.Available ' há bytes recebidos ?
        End If

        If nCaracteres > 0 Then
            Mensagem.Read(carateres, 0, nCaracteres) 'Le os bytes recebidos
        End If
    End Sub

```

Até aqui o código é igual ao exemplo anterior.

As linhas de código seguintes analisam a mensagem recebida, detetam o nome “MotorOn” ou “MotorOff” e chamam a função “EnviaFicheiro(...)”. Esta função lê o ficheiro e envia o seu conteúdo para o Browser.

```

' Converte o array de bytes "caracteres", numa string "textorecebido"
textorecebido = Encoding.Default.GetChars(caracteres)
TextBox1.Text = TextBox1.Text + textorecebido + vbCrLf

' Se na mensagem recebida constar a a palavra MotorOn ou MotorOff
' deve ser aberto o ficheiro e enviado par a browser
If InStr(textorecebido, "MotorOn.html") Then
    EnviarFicheiro("MotorOn.html")
ElseIf InStr(textorecebido, "MotorOff.html") Then
    EnviarFicheiro("MotorOff.html")
End If

LigacaoTCP.Close()      ' O Protocolo HTTP preve o fim da ligacao TCP
LigacaoTCP = New TcpClient
nCaracteres = 0
textorecebido = ""
End If

End Sub

] Private Sub EnviarFicheiro(ByVal sss As String)
    If LigacaoTCP.Connected = True Then
        Dim ler As StreamReader
        ler = New StreamReader(sss)
        Dim ss As String
        ss = ler.ReadToEnd() ' Le o ficheiro, para a string "ss"
        Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss) ' Converte a string "ss" nu
        Mensagem.Write(arraybytes, 0, Len(ss)) ' Envia o array de bytes para o Browser
        ler.Close()
        ss = ""
    End If
End Sub

End Class

```

10.2.3. Exemplo – Servidor WEB com três páginas HTML

Neste exemplo, o ServidorWEB, desenvolvido em VBasic, deve enviar três documentos HTML para o Browser.

A figura seguinte apresenta os três documentos HTML e o Browser.

Quando o Browser pedir o documento “Reservatorio.html”, mostra duas “frames”. A “frame” de cima apresenta o documento “Controlo.html”, com os botões para ligar ou desligar Y0, Y1, e Y2. A “frame” de baixo apresenta o documento “Supervisao.html”, com o estado dos NívelX0, X1, X2 e X3. A página “Reservatorio.html” define a organização da página que o utilizador verá no Browser, em duas “Frames”, e que documento HTML aparece em cada “Frame”.

Browser



Página Controlo.html

```
Controlo - Notepad
File Edit Format View Help
HTTP/1.1 200 OK
Content-Type: text/html
<html>
<head>
</head>
<body>
<b>Reservatorio</b> <br><br>
<form>
Controlo Motor_Y0<br>
<input type= "submit" Name="Y0" value="Ligar_Y0">
<input type= "submit" Name="Y0" value="Desligar_Y0">
<br>Electrovalvula de entrada Y1<br>
<input type= "submit" Name="Y1" value="Ligar_Y1">
<input type= "submit" Name="Y1" value="Desligar_Y1">
<br>Electrovalvula de saída Y2<br>
<input type= "submit" Name="Y2" value="Ligar_Y2">
<input type= "submit" Name="Y2" value="Desligar_Y2">
</form>
</body>
</html>
```

Página Reservatorio.html

```
Reservatorio - Notepad
File Edit Format View Help
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length:500
<HTML>
<HEAD>
<title>Reservatorio</title>
</HEAD>
<FRAMESET ROWS="50%,50%">
<frame src="Controlo.html" name="cima" id="cima" />
<frame src="Supervisao.html" name="baixo" id="baixo" />
</FRAMESET>
</HTML>
```

Página Supervisao.html

```
Supervisao - Notepad
File Edit Format View Help
HTTP/1.1 200 OK
Content-Type: text/html
<html>
<head>
<meta http-equiv="refresh" content="1">
</head>
<body>
<br><br> Supervisao <br>
Nivel_X0:000 Nivel_X1:111 Nivel_X2:222, Nivel_X3:333
</body>
</html>
```

O documento Reservatorio.html apresenta uma novidade, relativamente ao exemplo anterior.

Usa “Frames”:

```
<FRAMESET ROWS="50%,50%">
    <frame src="Controlo.html" name="cima" id="cima" />
    <frame src="Supervisao.html" name="baixo" id="baixo" />
</FRAMESET>
```

O código do ServidorWEB, é semelhante aos exemplos anteriores. Possui apenas mais algumas linhas de código:

- O textorecebido passa a guardar apenas os 40 caracteres iniciais (GET)

```
' Converte o array de bytes "caracteres", numa string "textorecebido"
textorecebido = Encoding.Default.GetChars(caracteres)
textorecebido = Mid(textorecebido, 1, 40)
```

Nos 40 caracteres iniciais, pode ser encontrado o nome do documento HTML pedido pelo Browser.

```
        ElseIf InStr(textorecebido, "Controlo.html") Then
            EnviarFicheiro("Controlo.html")
        ElseIf InStr(textorecebido, "Supervisao.html") Then
            EnviarFicheiro("Supervisao.html")
        ElseIf InStr(textorecebido, "Reservatorio.html") Then
            EnviarFicheiro("Reservatorio.html")
    End If
```

Quando o Browser pede a página “Reservatorio.html”, e detecta que contém duas “frames”, cada uma com uma página HTML, por isso o Browser volta a estabelecer uma ligação TCPCIP com o servidor e pede-lhe os dois documentos em falta.

Exemplo completo do ServidorWEB

```
Imports System.Net
Imports System.Net.Sockets ' permite usar TcpClient, TcpListener, NetworkStream
Imports System.IO ' manipulacao de ficheiros em disco
Imports System.Text ' encoding converte array bytes em string

Public Class Form1
    Dim EsperaLigacaoTCP As New TcpListener(81) 'espera pedidos de ligacao TCP/IP
    Dim LigacaoTCP As New TcpClient 'controla a ligacao TCP/IP
    Dim Mensagem As NetworkStream 'recebe ou envia array de bytes, usando o TcpClient
    Dim carateres(4000) As Byte
    Dim nCaracteres As Integer
    Dim textorecebido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 1000
        Timer1.Enabled = True
        EsperaLigacaoTCP.Start() ' começa à escuta da port 81
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If EsperaLigacaoTCP.Pending() = True Then ' Há ligacao à espera de ser aceite
            LigacaoTCP = EsperaLigacaoTCP.AcceptTcpClient() ' aceita ligação TCP/IP
            Mensagem = LigacaoTCP.GetStream() ' Le msg recebida no formato NetwokStream
        End If

        txtEstadoLigacao.Text = LigacaoTCP.Connected ' retorna o estado da ligacao TCP

        If LigacaoTCP.Connected = True Then
            nCaracteres = LigacaoTCP.Available ' há bytes recebidos ?
        End If

        If nCaracteres > 0 Then
            Mensagem.Read(carateres, 0, nCaracteres) 'Le os bytes recebidos
        End If
    End Sub
```

```

' Converte o array de bytes "caracteres", numa string "textorecebido"
textorecebido = Encoding.Default.GetChars(caracteres)
textorecebido = Mid(textorecebido, 1, 40)

TextBox1.Text = TextBox1.Text + textorecebido + vbCrLf

' Se na mensagem recebida constar a a palavra MotorOn ou MotorOff
' deve ser aberto o ficheiro e enviado par a browser
If InStr(textorecebido, "MotorOn.html") Then
    EnviarFicheiro("MotorOn.html")
ElseIf InStr(textorecebido, "MotorOff.html") Then
    EnviarFicheiro("MotorOff.html")
ElseIf InStr(textorecebido, "Controlo.html") Then
    EnviarFicheiro("Controlo.html")
ElseIf InStr(textorecebido, "Supervisao.html") Then
    EnviarFicheiro("Supervisao.html")
ElseIf InStr(textorecebido, "Reservatorio.html") Then
    EnviarFicheiro("Reservatorio.html")
End If

LigacaoTCP.Close()      ' O Protocolo HTTP preve o fim da ligacao TCP
LigacaoTCP = New TcpClient
nCaracteres = 0
textorecebido = ""
End If

End Sub

Private Sub EnviarFicheiro(ByVal sss As String)
If LigacaoTCP.Connected = True Then
    Dim ler As StreamReader
    ler = New StreamReader(sss)
    Dim ss As String
    ss = ler.ReadToEnd()  ' Le o ficheiro, para a string "ss"
    Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss) ' Converte a string "ss" num array
    Mensagem.Write(arraybytes, 0, Len(ss))  ' Envia o array de bytes para o Browser
    ler.Close()
    ss = ""
End If
End Sub

```

10.2.4. Exemplo – Servidor WEB específico para controlar um motor

Este exemplo demonstra como é possível controlar o funcionamento de um PLC através de um servidor WEB desenvolvido em VBasic. Este programa permite que um equipamento, neste caso um motor, seja ligado ou desligado através da Internet, mais exatamente a partir de um Browser WEB, portanto a partir de qualquer ponto do globo.

A figura seguinte apresenta os recursos utilizados para alcançar os objectivos propostos: um motor; um PLC para controlar o motor; um computador local para receber as ordens vindas da Internet e comunicar com o PLC através de uma ligação RS232; e um computador remoto (*BrowserWEB*) ligado à Internet de onde serão dadas as ordens para ligar ou desligar o motor.

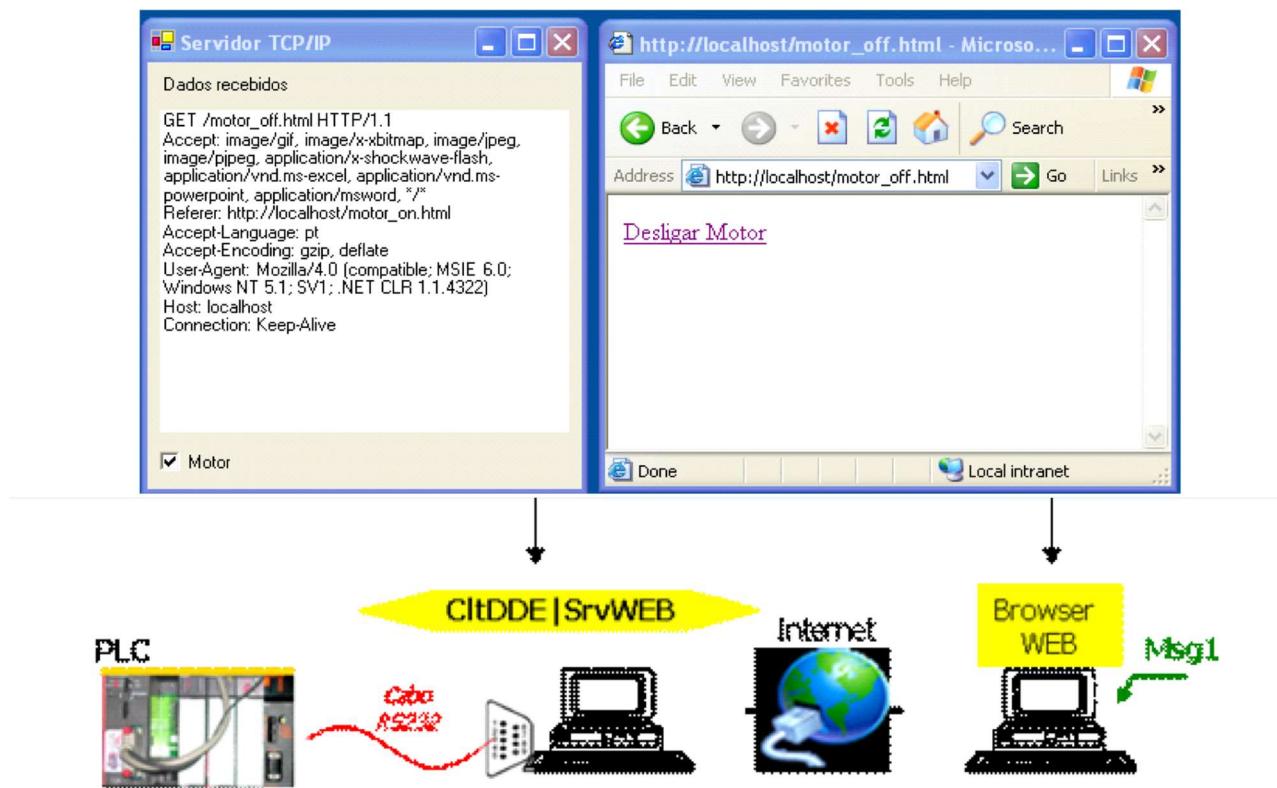
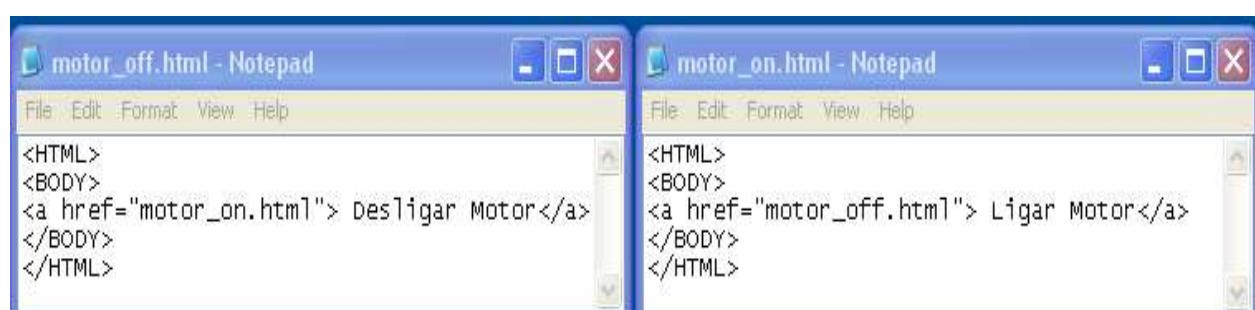


Figura : Controlo remoto de um motor a partir da Internet.

Para alcançar o objectivo proposto é necessário desenvolver um servidor Web que envie uma de duas páginas para o BrowserWeb. Uma das páginas tem a indicação para ligar o motor e a outra para desligar, o texto tem uma hiperligação para a outra página.



10.2.5. Exemplo – Servidor WEB específico para controlar um Motor II

Este exemplo apresenta um programa desenvolvido em VBasic que permite controlar remotamente um motor e também monitorizar no Browser Web o estado de funcionamento do Motor (On/Off). De facto este programa atua como um servidor WEB básico. Sempre que o Browser comunicar com o computador “localhost”, usando o protocolo HTTP, este programa desenvolvido em VBasic recebe a mensagem enviada pelo Browser (“GET ...”).

De acordo com o protocolo HTTP o VBasic deve responder ao Browser com uma mensagem de texto (HTML). Neste exemplo, o VBasic deve alterar dinamicamente a resposta que envia para o Browser, consoante o motor está ligado ou desligado.

O protocolo HTTP também define que apenas o Browser pode tomar a iniciativa de estabelecer uma ligação e fazer pedidos a um servidor WEB. Desta forma, para o servidor ter a oportunidade de enviar dados de segundo a segundo para o Browser é necessário que página HTML instrua o Browser para fazer pedidos de segundo a segundo ao servidor. Por essa razão a página WEB, escrita em HTML, deverá também possuir a instrução “*refresh*” para que o Browser repita automaticamente o pedido “GET ...” e o envie de segundo a segundo para o servidor.

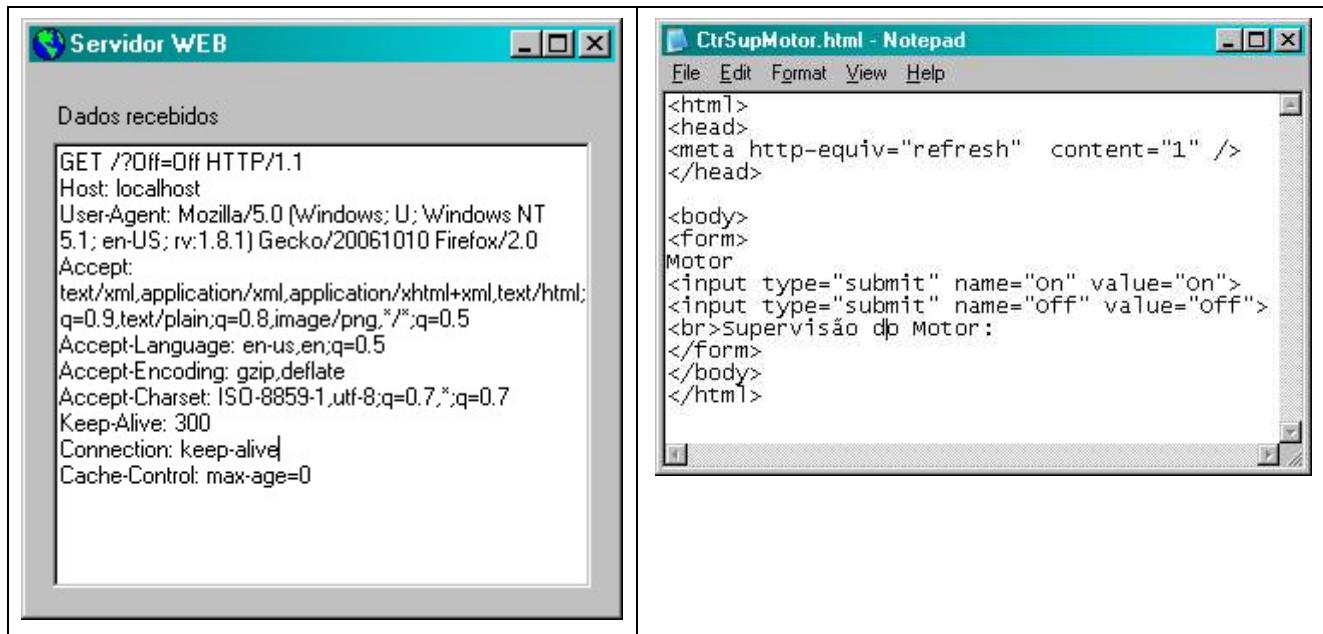


Página MotorCtr.html

```
<html>
<head>
<!meta http-equiv="refresh" content="1">
</head>
<body>
<b>Reservatorio</b> <br><br>
<form>

Motor_Y0<br>
<input type= "submit" Name="Y0" value="Ligar_Y0">
<input type= "submit" Name="Y0" value="desligar_Y0">
<br>Electrovalvula de entrada Y1<br>
<input type= "submit" Name="Y1" value="Ligar_Y1">
<input type= "submit" Name="Y1" value="desligar_Y1">
<br>Electrovalvula de saída Y2<br>
<input type= "submit" Name="Y2" value="Ligar_Y2">
<input type= "submit" Name="Y2" value="desligar_Y2">

<br><br> Supervisao <br>
Nivel_X0:000 Nivel_X1:111 Nivel_X2:222, Nivel_X3:333
</form>
</body>
</html>
```



xemplo4_2 (Running) - Microsoft Visual Studio

Form1.vb [Design]

General] [Declarations]

```
Imports System.IO
Public Class Form1
    Dim WithEvents ws As New MSWinsockLib.Winsock
    Dim WithEvents fs As New FaconSrv.FaconServer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        'Configuração do objecto Winsock
        ws.Protocol = MSWinsockLib.ProtocolConstants.sckTCPProtocol
        ws.LocalPort = 80
        ws.Listen()
        'Configuração da aplicação FaconSrv
        fs.OpenProject("c:\plcgab.fcs")
    End Sub
    'Quando o IExplorer tentar contactar este computador, este procedimento é chamado automaticamente
    Private Sub ws_ConnectionRequest(ByVal requestID As Integer) Handles ws.ConnectionRequest
        If ws.State <> 0 Then ws.Close()
        ws.Accept(requestID)      ' A ligação TCP/IP fica estabelecida
        fs.Connect()              ' A ligação do Faconsrv com o PLC fica estabelecida
    End Sub
    ' Quando o IExplorer envia dados para este computador, este procedimento é chamado
    Private Sub ws_DataArrival(ByVal bytesTotal As Integer) Handles ws.DataArrival
        Dim dados As Object
        Dim posicao, comprimento As Integer
        Dim ficheiro As StreamReader
        Dim respostahtml, a, b, c As String

        ws.GetData(dados, vbString, bytesTotal)
        txtDadosRecebidos.Text = dados
        posicao = InStr(dados, "On")  ' devolve 0 se a substring não for encontrada
        If posicao > 0 Then
            fs.SetItem("channel0.station0.group0", "Y4", "1")
        Else
            fs.SetItem("channel0.station0.group0", "Y4", "0")
        End If
    End Sub
End Class
```

```
ficheiro = New StreamReader("CtrSupMotor.html")
respostahtml = ficheiro.ReadToEnd
posicao = InStr(respostahtml, ":")
comprimento = Len(respostahtml) - posicao
a = Mid(respostahtml, 1, posicao)
b = fs.GetItem("channel0.station0.group0", "Y4")
b = Mid(b, 1, 1)
c = Mid(respostahtml, posicao + 1, comprimento)
txtDadosRecebidos.Text = a & b & c
ws.SendData(a & b & c)
ficheiro.Close()
- End Sub

] Private Sub ws_Error(ByVal Number As Short, ByRef Description As String, ByVal Scode As Integer, ByVal ErrorCode As Long)
    ws.Close()
    ws.Listen()
- End Sub
' Quando a resposta deste programa chega ao IExplorer este procedimento é chamado
] Private Sub ws_SendComplete() Handles ws.SendComplete
    ws.Close()
    ws.Listen()
- End Sub

] Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs)
    ws.Close()
- End Sub
-End Class
```



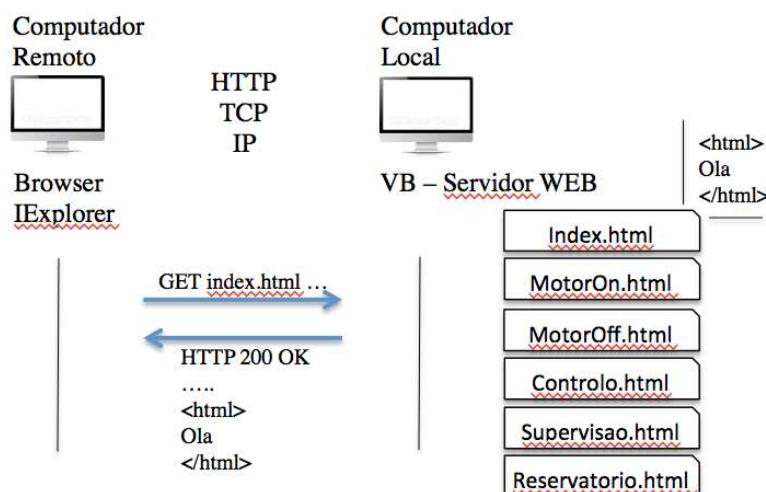
Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

10.3. Trabalho prático nº 9 – Servidor WEB/HTTP

Este trabalho tem por objectivo facilitar a compreensão do protocolo HTTP, lecionado nas aulas teóricas. Para isso, pretende-se desenvolver em VBasic um servidor WEB/HTTP, elementar, que possa responder a pedidos de um browser WEB.

De um forma simplista podemos afirmar que um servidor WEB não é mais que um programa que aceita ligações TCPIP, habitualmente na porta 80, e que responde com o conteúdo dos documentos HTML pedidos pelo Browser. Os pedidos do Browser consistem em mensagens de texto, enviadas por TCPIP para o servidor (ex. GET \index.html). A sintaxe dessas mensagens/pedidos/respostas é definida pelo protocolo HTTP.



Neste trabalho serão utilizados dois computadores (figura):

- Um dos computadores, o **computador local**, tem o **servidorWEB** desenvolvido em VBasic. O ServidorWEB aceita os pedidos de ligação TCPIP e os pedidos de documentos HTML (GET ...) efectuados pelo browser WEB, de acordo com o protocolo HTTP.
Os **documentos HTML** residem no disco duro do computador local. Quando a aplicação ServidorWEB recebe pedidos de documentos HTML, lê o seu conteúdo e envia-o por TCPIP para o BrowserWEB remoto.
- O segundo computador, o **computador remoto**, possui um **browser WEB** que acede ao computador local através do **protocolo HTTP/TCP/IP**.

Trabalho de casa

Crie os documentos HTML que forem necessários e crie um ServidorWEB em VBasic de forma que o Browser apresente as 4 “frames” seguintes:



A frame da esquerda, apresenta um novo documento, a criar em casa, com o nome “Logo.html”. Esse documento tem a referência à imagem:

A frame do centro, apresenta o documento “Controlo.html”

A frame da direita, apresenta o documento “Supervisao.html”

A frame de baixo, apresenta um novo documento, a criar, com nome “Rodape.html”

O documento que define as frames e que o browser pede é o “Reservatorio2.html”

O ServidorWEB deve ser alterado de forma a poder enviar para o Browser os documentos: “Logo.html”, “Rodape.html” e “Reservatorio2.html”.

Conhecimentos a adquirir

Introdução ao protocolo HTTP e ao funcionamento dos Servidores WEB.

Familiarização com páginas HTML

- Sequência de interações Cliente WEB – servidor WEB
- Sintaxe dos pedidos HTTP e respectivas respostas.
- Estrutura e sintaxe dos documentos HTML

Importante:

- O trabalho será avaliado por questionário individual, na semana seguinte à entrega do mesmo.
- Deve submeter no Moodle o programa desenvolvido sob pena da nota obtida no questionário não ser considerada.

I- Desenvolva em VBasic, um servidor WEB/HTTP (ver exemplo no final do documento):

- A **aplicação local**, desenvolvida em VBasic, deverá utilizar objetos do tipo: **TcpListener**, **TcpClient**, e **NetworkStream** para receber ligações TCP/IP e transferir dados; e um objecto do tipo **StreamReader** para ler os documentos disponíveis no disco local. Por exemplo, ler os documentos: MotorOn.html, MotoOff.html, Controlo.html, Supervisao.html, e Reservatorio.html

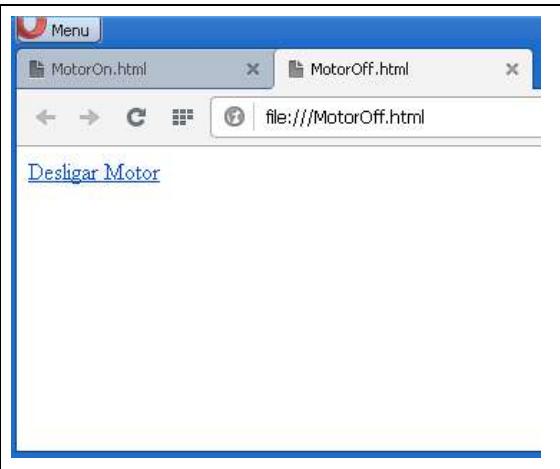
II- Edite os 5 documentos HTML

E guarde-os no diretório:

...\\VisualStudio...\\Projects\\... \\Bin\\Debug

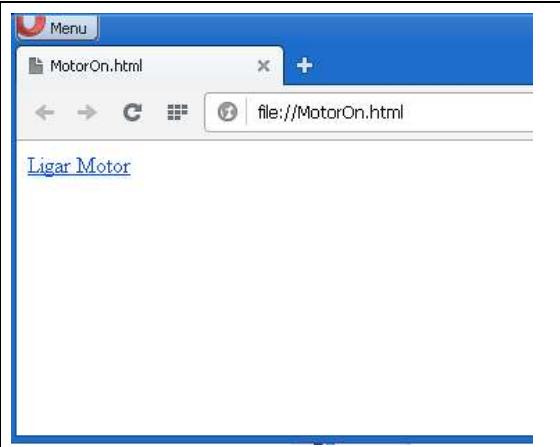
Página *MotorOff.html*

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 72  
  
<html>  
<body>  
<a href="MotorOn.html"> DesligarMotor </a>  
</body>  
</html>
```



Página *MotorOn.html*

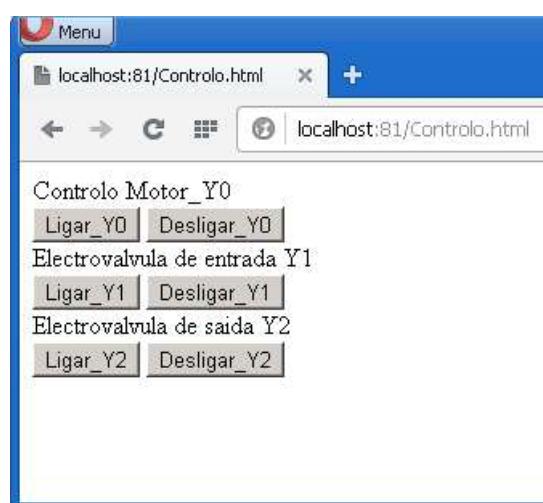
```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 70  
  
<html>  
<body>  
<a href="MotorOff.html"> LigarMotor </a>  
</body>  
</html>
```



Página Controlo.html

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 462

<html>
<head>
</head>
<body>
<form>
Controlo Motor_Y0<br>
<input type= "submit" Name="Y0" Value="Ligar_Y0">
<input type= "submit" Name="Y0" Value="Desligar_Y0">
<br>Electrovalvula de entrada Y1<br>
<input type= "submit" Name="Y1" Value="Ligar_Y1">
<input type= "submit" Name="Y1" Value="Desligar_Y1">
<br>Electrovalvula de saida Y2<br>
<input type= "submit" Name="Y2" Value="Ligar_Y2">
<input type= "submit" Name="Y2" Value="Desligar_Y2">
</form>
</body>
</html>
```



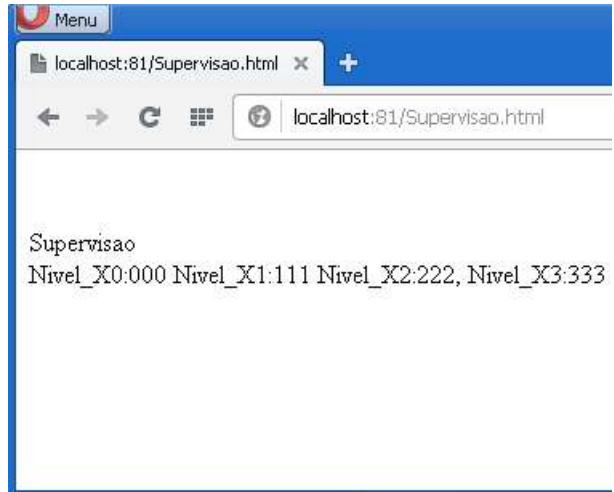
Página Supervisao.html

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 170

<html>
<head>
<meta http-equiv="refresh" content="1">
</head>
<body>
<br><br> Supervisao <br>

Nivel_X0:000
Nivel_X1:111
Nivel_X2:222,
Nivel_X3:333

</body>
</html>
```



Página Reservatorio.html

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length:207
```

```
<HTML>  
<HEAD>  
<title>Reservatorio</title>  
</HEAD>  
  
<FRAMESET ROWS="50%,50%">  
<frame src="Controlo.html" name="cima" id="cima" />  
<frame src="Supervisao.html" name="baixo" id="baixo" />  
</FRAMESET>  
</HTML>
```

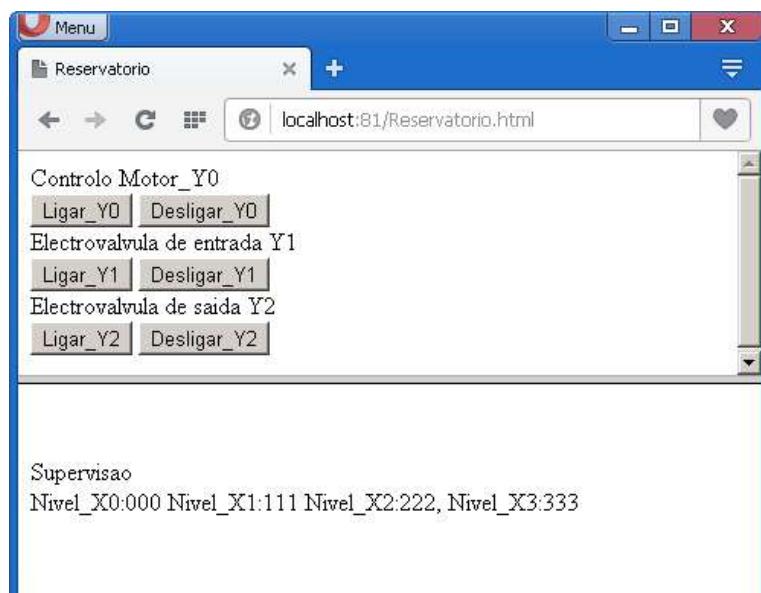


III – Aceda ao ServidorWEB através do Browser

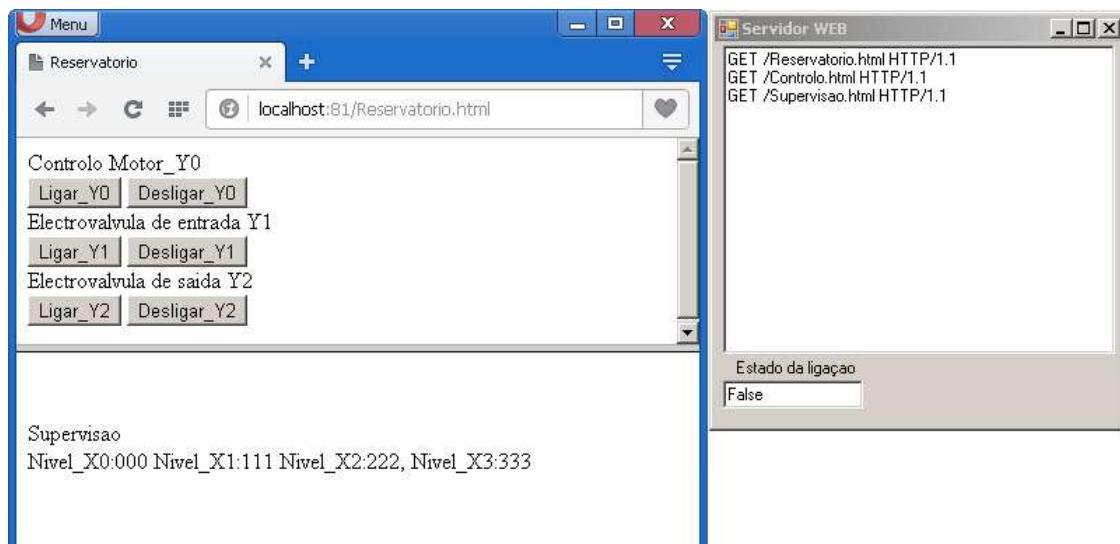
Depois de ter editado o programa exemplo, descrito no final deste documento (ServidorWEB), e depois de ter editado os cinco documentos HTML, poderá, a partir do computador remoto, usando um **browserWEB**, por exemplo o Safari, o IExplorer, o Opera, ou outro, aceder ao servidorWEB (que desenvolveu em VBasic) e pedir os 5 documentos:

<http://localhost/MotorOn.html>
<http://localhost/MotorOff.html>
<http://localhost/Controlo.html>
<http://localhost/Supervisao.html>
<http://localhost/Reservatorio.html>

Quando aceder a <http://localhost/Reservatorio.html> o seu Browser deverá apresentar a imagem seguinte:



Exemplo completo do ServidorWEB, em VBasic, para a aula.



```
Imports System.Net
Imports System.Net.Sockets ' permite usar TcpClient, TcpListener, NetworkStream
Imports System.IO ' manipulacao de ficheiros em disco
Imports System.Text ' encoding converte array bytes em string

Public Class Form1
    Dim EsperaligacaoTCP As New TcpListener(81) 'espera pedidos de ligacao TCPIP
    Dim LigacaoTCP As New TcpClient 'controla a ligacao TCPIP
    Dim Mensagem As NetworkStream 'recebe ou envia array de bytes, usando o TcpClient
    Dim carateres(4000) As Byte
    Dim nCaracteres As Integer
    Dim textorecebido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Interval = 1000
        Timer1.Enabled = True
        EsperaligacaoTCP.Start() 'começa à escuta da port 81
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If EsperaligacaoTCP.Pending() = True Then 'Há ligacao à espera de ser aceite
            LigacaoTCP = EsperaligacaoTCP.AcceptTcpClient() 'aceita ligacao TCPIP
            Mensagem = LigacaoTCP.GetStream() 'Le msg recebida no formato NetwokStream
        End If

        txtEstadoLigacao.Text = LigacaoTCP.Connected 'retorna o estado da ligacao TCP

        If LigacaoTCP.Connected = True Then
            nCaracteres = LigacaoTCP.Available 'há bytes recebidos ?
        End If

        If nCaracteres > 0 Then
            Mensagem.Read(carateres, 0, nCaracteres) 'Le os bytes recebidos
        End If
    End Sub

```

```

' Converte o array de bytes "caracteres", numa string "textorecebido"
textorecebido = Encoding.Default.GetChars(caracteres)
textorecebido = Mid(textorecebido, 1, 40)

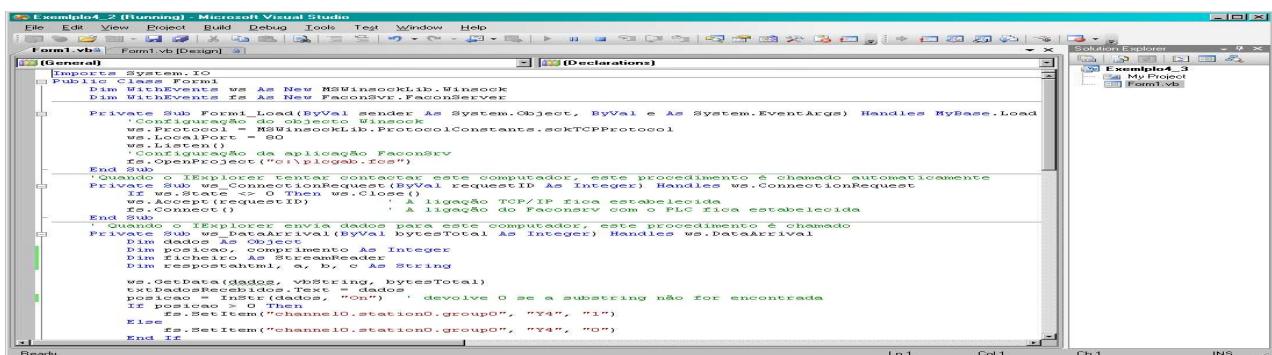
TextBox1.Text = TextBox1.Text + textorecebido + vbCrLf

' Se na mensagem recebida constar a a palavra MotorOn ou MotorOff
' deve ser aberto o ficheiro e enviado par a browser
If InStr(textorecebido, "MotorOn.html") Then
    EnviarFicheiro("MotorOn.html")
ElseIf InStr(textorecebido, "MotorOff.html") Then
    EnviarFicheiro("MotorOff.html")
ElseIf InStr(textorecebido, "Controlo.html") Then
    EnviarFicheiro("Controlo.html")
ElseIf InStr(textorecebido, "Supervisao.html") Then
    EnviarFicheiro("Supervisao.html")
ElseIf InStr(textorecebido, "Reservatorio.html") Then
    EnviarFicheiro("Reservatorio.html")
End If

LigacaoTCP.Close()           ' O Protocolo HTTP preve o fim da ligacao TCP
LigacaoTCP = New TcpClient
nCaracteres = 0
textorecebido = ""
End If

```

End Sub



Bibliografia

- [Rosen 2003] **Web Application Architecture, Principles, Protocols and Practices**, Leon Shklar, Richard Rosen, John Wiley & Sons, 2003
Chapter 3. Birth of the World Wide Web: HTTP
Chapter 4. Web Servers
Chapter 6. HTML and Its Roots

Norma HTTP

<https://www.ietf.org/rfc/rfc2616.txt>

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>

http-protocol.wmv (25 min)

<https://www.youtube.com/watch?v=WGJrLqtX7As>

Introdução ao HTML

<http://www.w3schools.com>



CAPÍTULO

10

HTML

JPSantos
2023

11.HTML – HiperText Markup Language

11.1. Conceitos sobre HTML

HTML significa Hypertext Markup Language. Esta linguagem foi proposta por Tim-Berners e tinha por objectivo permitir escrever e formatar documentos com texto, tabelas e hiperligações a outros documentos, imagens, filmes e ficheiros em geral.

De uma forma simples e incompleta podemos dizer que esta linguagem define um conjunto de palavras chave, etiquetas, <tags> que um browser Web sabe interpretar e em função delas apresentar ao utilizador uma página WEB.

Com base nestas etiquetas é possível escrever documentos de texto com uma estrutura e uma formatação conhecida, não proprietária, podendo por isso ser utilizadas por todos.

As páginas WEB, escritas em HTML, são documentos de texto que têm de ser previamente gravados, por exemplo em disco, antes de poderem ser lidas pelos browsers WEB.

11.1.1. Estrutura de um documento HTML

Um documento HTML está organizado em duas partes principais: cabeçalho “header” e o corpo principal “body”. O documento HTML, como um todo, está contido pelas etiquetas <HTML> ... </HTML>.

```
<HTML>
  <HEAD>
    ...
  </HEAD>
  <BODY>
    ...
  </BODY>
</HTML>
```

11.1.2. Etiquetas HTML

As etiquetas HTML estão definidas como um conjunto de palavras reservadas que aparecem no documento de texto precedidas pelo sinal menor e seguidas pelo sinal maior, envolvendo a palavra chave <etiqueta>. As palavras reservadas <HTML> , <HEAD> e <BODY> são alguns exemplos de etiquetas. Parte destas etiquetas são apresentadas na secção seguinte e são explicadas exemplo a exemplo.

11.2. Exemplos de páginas HTML

Guarde num ficheiro de texto diferente, cada um dos exemplos seguintes. O primeiro exemplo deve ficar gravado com o nome “exemplo1.html” e os exemplos seguintes deverão ter um nome à sua escolha. Depois de editar e guardar em disco cada um dos exemplos seguintes visualize-os com um browser WEB.

Em cada exemplo é apresentado o “texto” HTML e o aspecto da página WEB correspondente, tal como o Browser a interpreta e a visualiza. Estes exemplos são autoexplicativos, as etiquetas aparecem a avermelho nos exemplos.

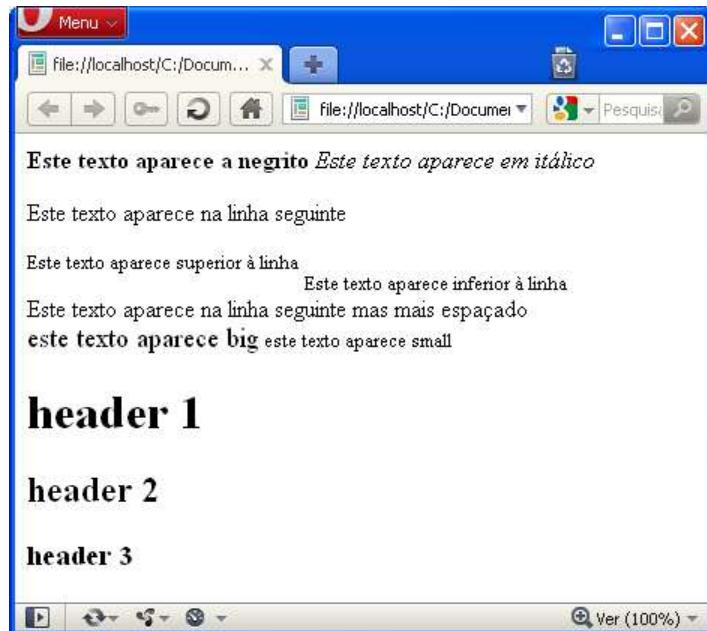
11.2.1. Estrutura de um documento HTML

```
<html>
  <head>
    <title>
      Um titulo para o documento
    </title>
  </head>
  <body>
    Este é o meu primeiro documento em HTML
  </body>
</html>
```



11.2.2. Formatar texto em HTML

```
<html>
  <head>
    <meta http-equiv="refresh" content="1" />
  </head>
  <body>
    <b>Este texto aparece a negrito </b>
    <i>Este texto aparece em itálico </i>
    <sup>Este texto aparece superior à linha </sup>
    <sub>Este texto aparece inferior à linha </sub>
    <p>Este texto aparece na linha seguinte </p>
    <br>Este texto aparece na linha seguinte mas mais espaçado
    <br>
    <big>este texto aparece big</big>
    <small>este texto aparece small</small>
    <h1>header 1 </h1>
    <h2>header 2 </h2>
    <h3>header 3 </h3>
  </body>
</html>
```



11.2.3. Hiperligações

```

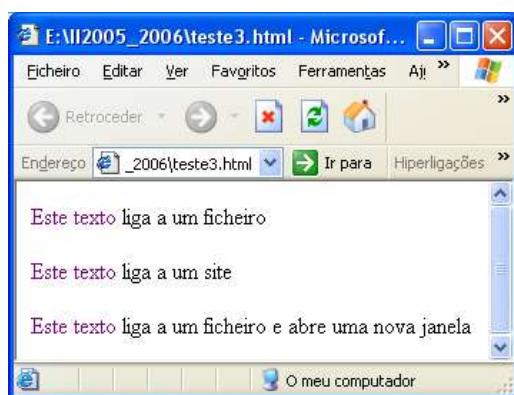
<html>
  <head>
  </head>
  <body>
    <p>
      <a href="teste2.html"> Este texto </a> liga a um ficheiro
    </p>

    <p>
      <a href="http://www.clix.pt"> Este texto </a> liga a um site
    </p>

    <p>
      <a href="teste2.html target = "_blank"> Este texto </a>
      liga a um ficheiro e abre uma nova janela
    </p>

  </body>
</html>

```



11.2.4. Inserir imagens

```
<html>
<head>
</head>
<body>

<p>Inserir uma imagem em movimento:

</p>

<p>Inserir uma imagem da WWW:

</p>

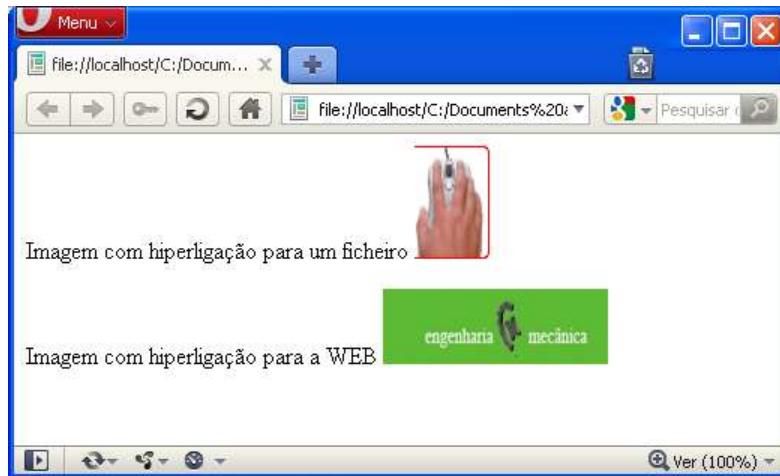
</body>
</html>
```



11.2.5. Inserir imagens com hiperligação

```
<html>
<head>
</head>
<body>
    <p> Imagem com hiperligação para um ficheiro
    <a href=teste2.html > <img border=0 src=hand1.gif width=50 Height=75 >
    </a>
    </p>

    <p> Imagem com hiperligação para a WEB
    <a href=http://cim3.mec.ua.pt > <img border=0 src=logo_2.jpg width=200 Height=50 >
    </a>
    </p>
</body>
</html>
```



11.2.6. Inserir imagens com áreas de hiperligação

```
<html>
  <body>
    <map name="mapa" id="NavMapa">
      <area shape="rect" coords="0,0,5,5" href="teste.html" alt="1Quad" />
      <area shape="circle" coords="50,50,30" href="teste2.html" alt="2Quad" />
    </map>
    
  </body>
</html>
```



11.2.7. Criar tabelas

```
<html>
  <head></head>
  <body>
    <p> Aqui está uma tabela </p>
    <table border=1>
      <tr> <td>Verde </td> <td>azul</td> <td>branco</td> </tr>
      <tr><td>Amarelo</td><td>Branco</td><td></td></tr>
    </table>
  </body>
</html>
```

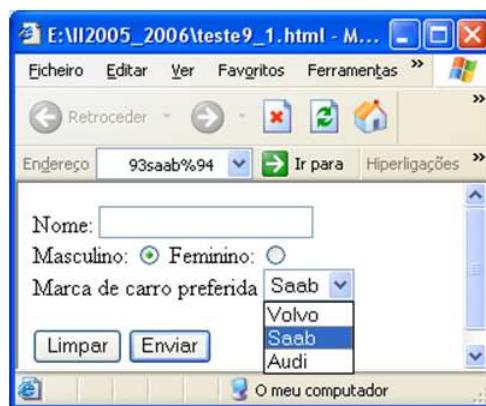


11.2.8. Criar formulários

```

<html>
<body>
    <form>
        Nome: <input type="text" name="nome" /> <br>
        Masculino: <input type="radio" checked="" name="sexo" value="M" />
        Feminino: <input type="radio" name="sexo" value="F" />
        <br> Marca de carro preferida
        <select name="auto">
            <option value="v"> Volvo
            <option value="s" selected> Saab
            <option value="a"> Audi
        </select> <br> <br>
        <input type="reset" value="Limpar" />
        <input type="submit" value="Enviar" />
    </form>
</body>
</html>

```



11.2.9. Criar frames

Frames horizontais

```

<html>
<FRAMESET ROWS="5%,*,50%">
    <frame src="teste1.html" name="cima" id="idcima" />
    <frame src="teste2.html" name="meio" id="idmeio" />
    <frame src="teste3.html" name="baixo" id="idbaixo" />

```

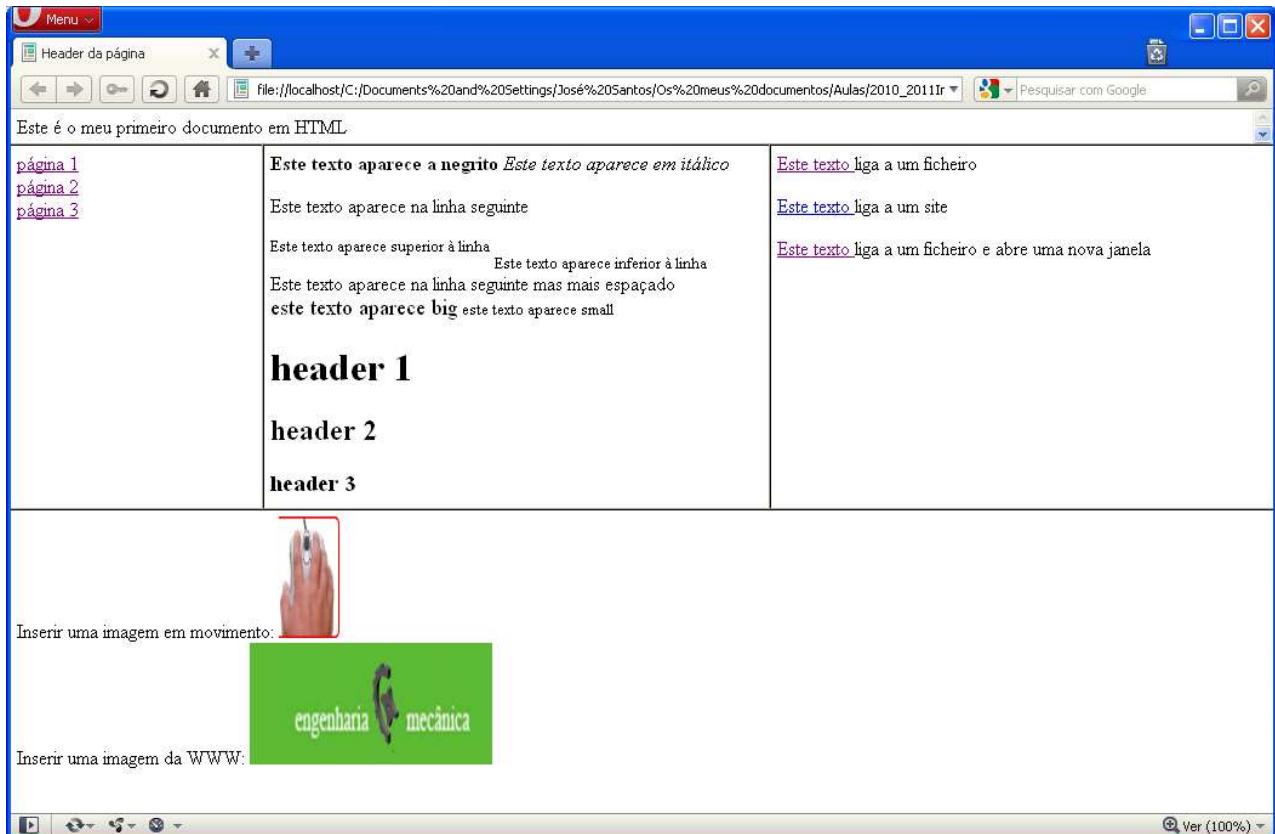
```
</FRAMESET>  
</html>
```

Frames verticais

```
<html>  
<FRAMESET COLS="30%,*,50%">  
    <frame src="teste1.html" name="esquerda" id="idesquerda" />  
    <frame src="teste2.html" name="meio" id="idmeio" />  
    <frame src="teste3.html" name="direita" id="iddireita" />  
</FRAMESET>  
</html>
```

11.2.10. Criar frames com targets

```
<html>  
    <FRAMESET ROWS="5%,*,50%">  
        <frame src="teste.html" name="cima" id="cima" />  
        <frameset cols="20%,*,40%">  
            <frame src="indice.html" name="esq" id="esq" />  
            <frame src="teste2.html" name="centro" id="centro" />  
            <frame src="teste3.html" name="dir" id="dir" />  
        </frameset>  
        <frame src="teste4.html" name="baixo" id="baixo" />  
    </FRAMESET>  
</html>  
  
<html>  
    <body>  
        <a href="teste.html" target="centro" >página 1 </a><br>  
        <a href="teste1.html" target="centro" >página 2 </a><br>  
        <a href="teste2.html" target="centro" >página 3 </a><br>  
    </body>  
</html>
```



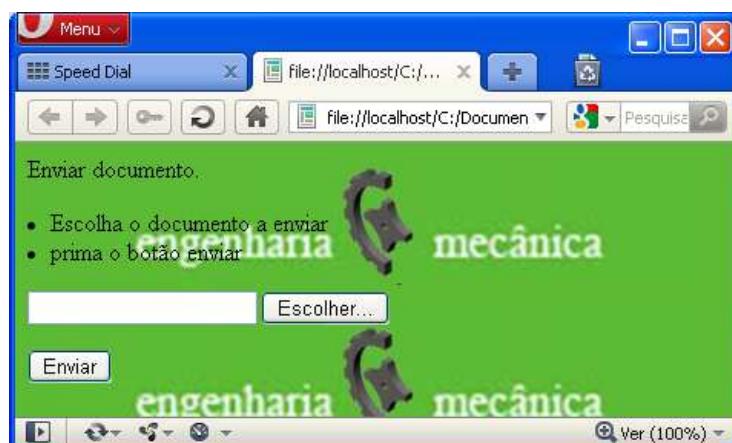
11.2.11. Enviar um ficheiro para o servidor

```

<html>
<body background="logo_2.jpg">
<form action="teste15FileSend.html" method="post">
  <p>Send me your resume.</p>

  <li>Escolha o documento a enviar</li><li>prima o botão enviar</li>
  <p><input type="file" enctype="multipart/form-data" name="resume" id="resume" /></p>
  <p><input type="submit" value="Enviar" name="submit" id="submit" /></p>
</body>
</html>

```



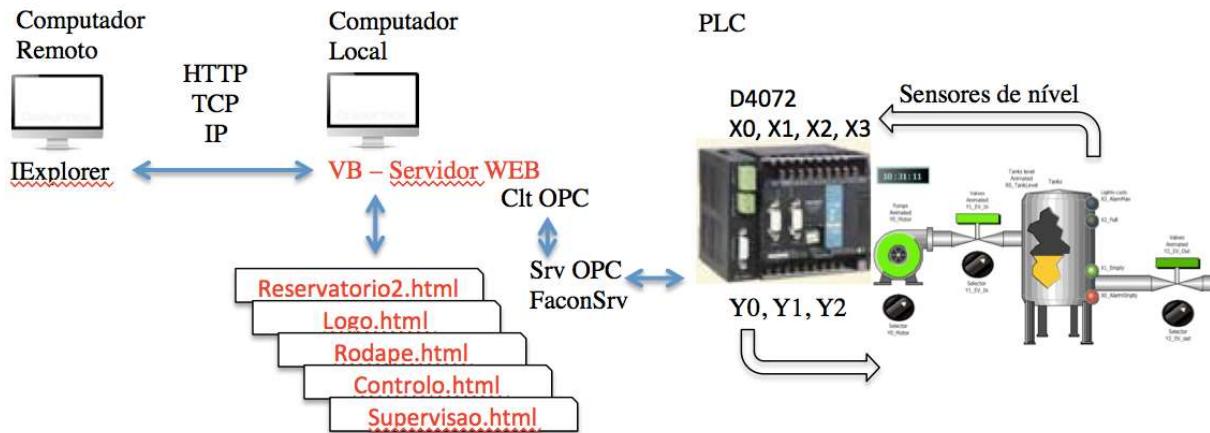


Informática Industrial 2022/2023

Universidade de Aveiro
Departamento de
Engenharia Mecânica

Trabalho prático nº 10 – SCADA remoto do PLC via Browser WEB (HTML/HTTP/TCP/IP/OPC)

Neste trabalho (uma semana), pretende-se que um utilizador em qualquer ponto do mundo possa usar um **Browser WEB** para comunicar com um PLC e, dessa forma, controlar e monitorizar remotamente um processo (ex. reservatório de água). Como o PLC não possui uma carta WEB é utilizado um computador local para fazer a interface entre o Browser WEB e o recurso fabril (ex. PLC).

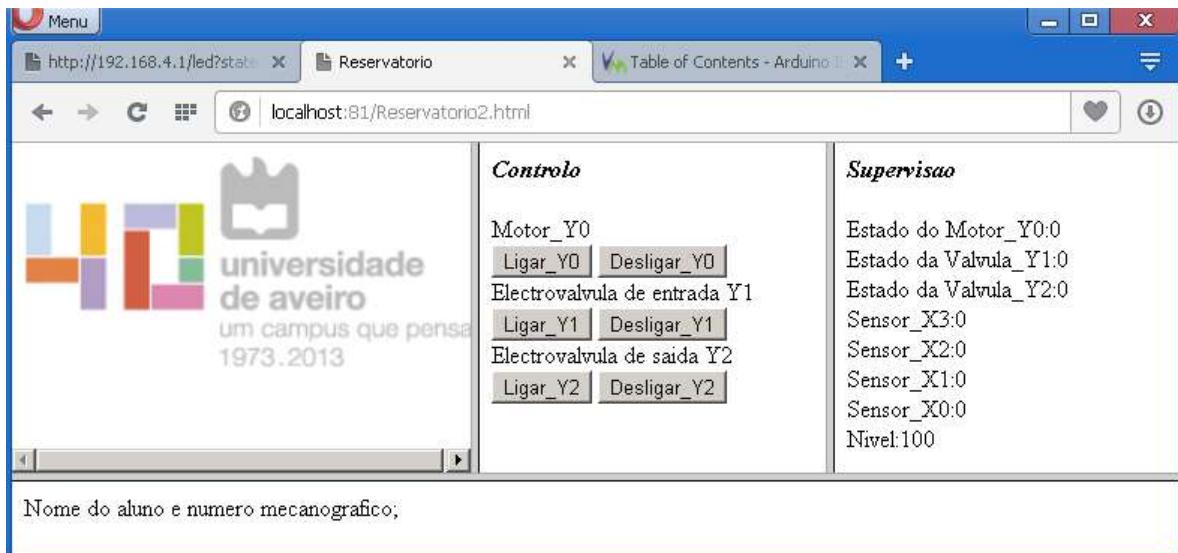


Serão utilizados dois computadores e um PLC (figura):

- O **computador local**, encontra-se **próximo do PLC** tem uma aplicação desenvolvida em Visual Basic que comunica com o PLC através de um servidor OPC (**FaconSrv**). Simultaneamente, a aplicação VB atua como um servidor WEB, capaz de receber os pedidos de um Browser WEB e de responder com páginas HTML que mostrem o estado das entradas e saídas do PLC.
- O **computador remoto**, possui apenas um Browser WEB para **visualizar o estado dos sensores de nível** do reservatório e o **estado das saídas**: o estado do motor e das electroválvulas de entrada e de saída.

Objectivos

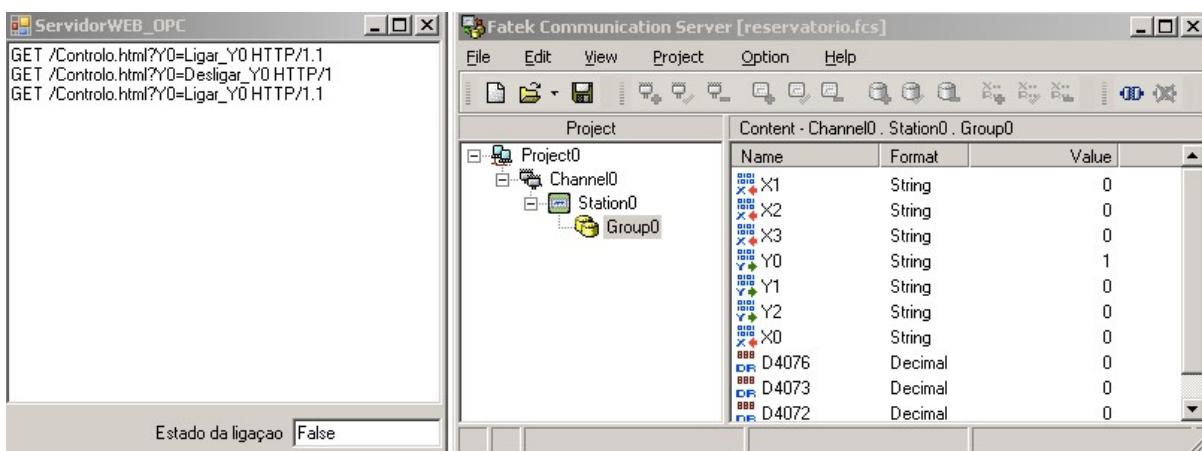
Conforme a figura seguinte ilustra, no computador remoto, o Browser WEB deve permitir ao utilizador **controlar e monitorizar o estado do motor e das electroválvulas (saídas Y0,Y1,Y2); dos sensores de nível (entradas X0,X1,X2,X3); e do nível de água (D4072)**.



Quando o utilizador, no browser, premir um dos botões da frame “controlo.html”, por exemplo o botão “Ligar_Y0” é enviado para o servidor WEB a mensagem HTTP:

“**GET /Controlo.html?Y0=Ligar_Y0**” (figura seguinte). Nessa altura, no computador local, **o servidorWEB**, através do FaconSrv, deve ativar o item “Y0”.

Na figura seguinte, (lado esquerdo), pode observar os primeiros 40 caracteres da mensagem HTTP recebida pelo servidor WEB sempre que o utilizador premir um dos botões no Browser (ex. Ligar_Y0).



Em síntese, para poder monitorizar e controlar remotamente o funcionamento do PLC/Reservatório a partir de um Browser WEB, é necessário:

- **Desenvolver em Visual Basic um servidor WEB**, semelhante ao do trabalho anterior, mas agora o Servidor WEB deve **alterar dinamicamente as páginas HTML** em função dos itens do PLC (X,Y,D), antes de as enviar para o Browser,
- **Utilizar um servidor OPC para aceder ao PLC** (como no trabalho nº 7).
- **Desenvolver as páginas HTML** para visualizar e controlar as entradas e saídas do PLC a partir do Browser.

Para realizar o trabalho pode utilizar o trabalho anterior:

- **Alterar o documento HTML** “Supervisao.html”
- **Alterar o código do Servidor WEB** que desenvolveu em Visual Basic.

Veja o exemplo seguinte, permite a supervisão e o controlo da saída Y0 do PLC. Adapte esse exemplo.

Conhecimentos a adquirir, ou a aprofundar:

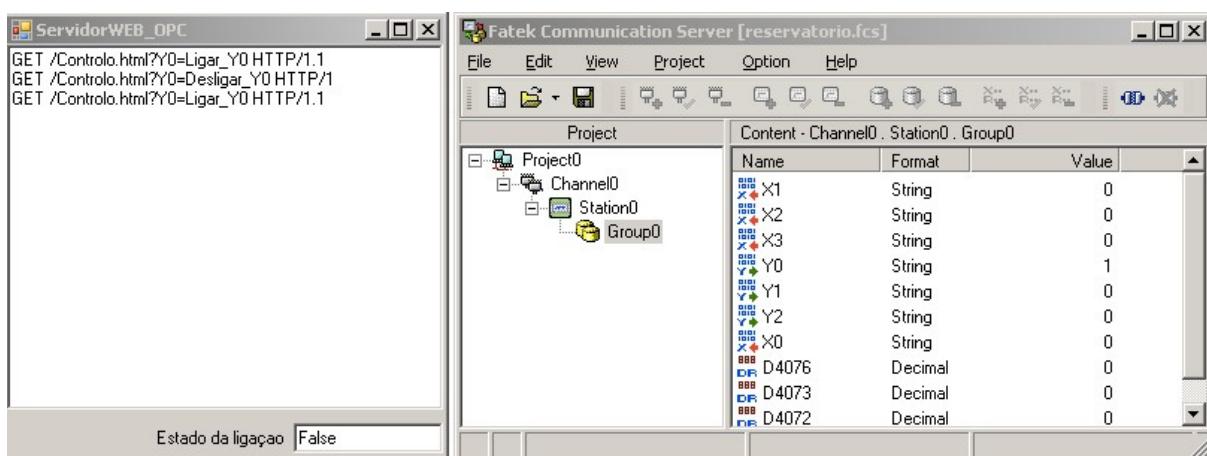
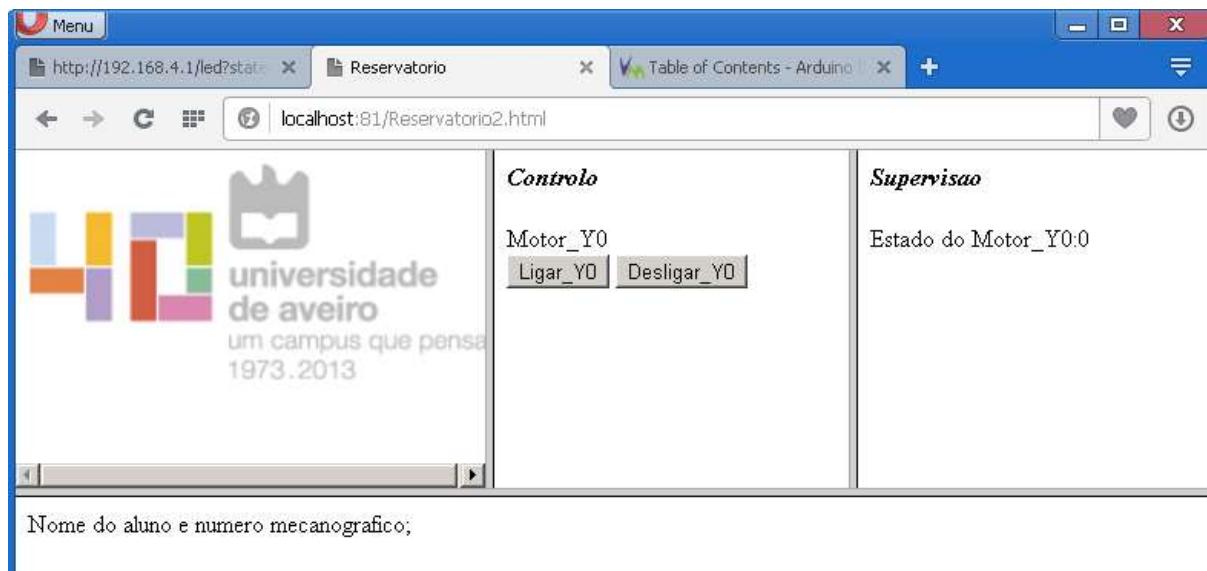
- HTML
- HTTP
- TcpClient
- TcpListener
- NetworkStream
- FaconSrv

Importante:

- Neste trabalho o PLC não tem programa, todo o controlo e supervisão é efectuado a partir do computador remoto/Browser WEB.
- O trabalho será avaliado por questionário individual, no 3º momento de avaliação.

Exemplo a realizar na sala de aula

Pretende-se que o utilizador a partir do Bowser WEB, ao premir o botão “Ligar_Y0”, possa ativar o item Y0 do FaconSrv (e do PLC).



I.1- Altere o código do trabalho anterior, para que seja enviado para o Brower o estado da saída Y0 do PLC.

No trabalho anterior, o subprocedimento “EnviarFicheiro”, além de ler o conteúdo de um ficheiro HTML, convertia todos os caracteres num array de bytes, e enviava esses bytes para o Brower.

Neste exemplo, o novo subprocedimento “EnviarFicheiro” além de fazer tudo o que fazia no trabalho anterior, troca alguns dos caracteres lidos no documento HTML por “0” ou por “1” conforme o estado da saída Y0 do PLC, e só depois os converte num array de bytes e os envia para o Brower.

Neste exemplo de código, é lido o estado da saída Y0 do PLC (`fs.GetItem...`) o seu valor é guardado na variável “estado”.

```
' Le os Itens do PLC
estado = fs.GetItem("Channel0.Station0.Group0", "Y0")
```

Relativamente ao trabalho anterior foram acrescentadas duas linhas de código na função “EnviarFicheiro”:

```
estado = fs.GetItem("Channel0.Station0.Group0", "Y0")
ss=Replace(ss,"YYY0",estado)
```

```

Private Sub EnviarFicheiro(ByVal sss As String)
    If LigacaoTCP.Connected = True Then
        Dim estado As String
        Dim ss As String
        Dim ler As StreamReader
        ler = New StreamReader(sss)

        ' Le ficheiro/documento HTML e guarda o texto na string "ss"
        ss = ler.ReadToEnd()

        ' Le os Itens do PLC
        estado = fs.GetItem("Channel0.Station0.Group0", "Y0")

        ' Altera string "ss" , troca os caracteres "YY0" que estiverem no documento HTML
        ' pelo estado da saída Y0 do PL ("0 ou "1")
        ss = Replace(ss, "YY0", estado)

        ' Converte a string "ss" num array de bytes, de acordo com tabela ASCII
        Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss)

        ' Envia o array de bytes para o Browser
        Mensagem.Write(arraybytes, 0, Len(ss))

        ler.Close()
        ss = ""
    End If
End Sub

```

I.2 - Altere o código do trabalho anterior para poder ligar ou desligar a saída Y0 do PLC , premindo o botão “Ligar_Y0” e o botão “Desligar_Y0”, presentes no Browser.

Repare que em relação ao trabalho anterior foram acrescentados dois
`fs.SetItem("Channel0.Station0.Group0", "Y0", "0")`

```

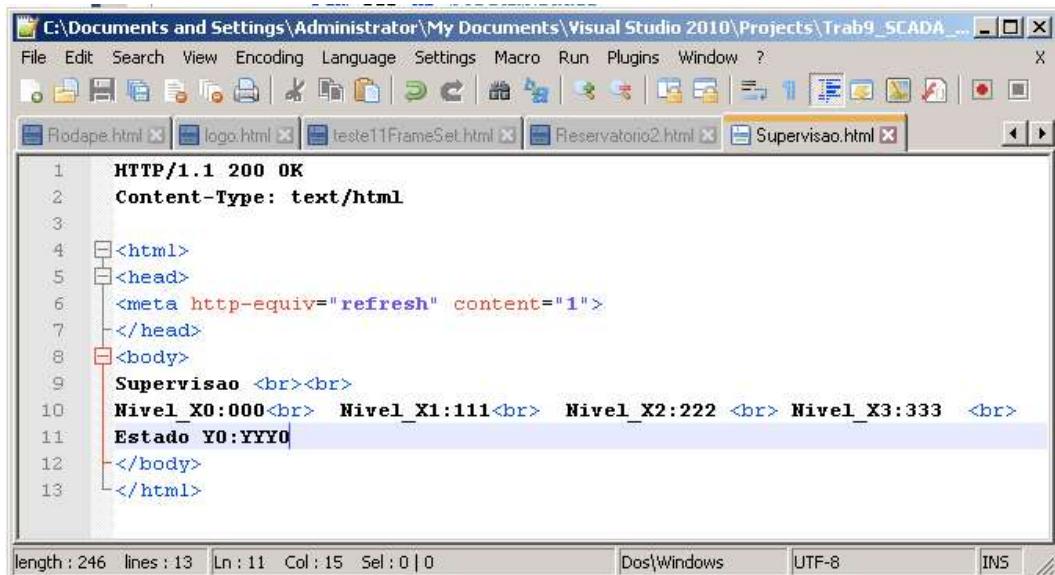
If nCaracteres > 0 Then
    Mensagem.Read(carateres, 0, nCaracteres) 'Le os bytes recebidos

    ' Converte o array de bytes "carateres", numa string "textorecebido"
    textorecebido = Encoding.Default.GetChars(carateres)
    textorecebido = Mid(textorecebido, 1, 40)

    'Atualizar Itens do PLC
    If InStr(textorecebido, "Ligar_Y0") Then
        fs.SetItem("Channel0.Station0.Group0", "Y0", "1")
    End If
    If InStr(textorecebido, "Desligar_Y0") Then
        fs.SetItem("Channel0.Station0.Group0", "Y0", "0")
    End If

```

II- Altere o ficheiro “Supervisao.html” do trabalho anterior, para conter a palavra “YYY0”



```
1 HTTP/1.1 200 OK
2 Content-Type: text/html
3
4 <html>
5 <head>
6 <meta http-equiv="refresh" content="1">
7 </head>
8 <body>
9 Supervisao <br><br>
10 Nivel_X0:000<br> Nivel_X1:111<br> Nivel_X2:222 <br> Nivel_X3:333 <br>
11 Estado Y0:YYY0
12 </body>
13 </html>
```

length : 246 lines : 13 Ln : 11 Col : 15 Sel : 0 | 0 Dos\Windows UTF-8 INS

O Visual Basic, depois de ler o ficheiro usando um objeto do tipo StreamReader, e guardar o seu conteúdo na variável “ss”, troca/replace a palavra “YYY0” por “0” ou “1”, conforme o estado da saída Y0 do PLC e guarda a novo texto na variável “ss”

```
' Altera string "ss" , troca os caracteres "YYY0" que estiverem no documento HTML
' pelo estado da saída Y0 do PL ("0" ou "1")
ss = Replace(ss, "YYY0", estado)
```

Posteriormente, converte a string “ss” num array de bytes e envia-os para o Browser.

Desta forma, a página visualizada no Browser muda dinamicamente a informação que disponibiliza para o utilizador, em função da saída Y0 do PLC.

Apêndices

Listagem Programa VB

```
Imports System.Net
Imports System.Net.Sockets      ' Utilização de TCPClient/TCPListener
Imports System.IO                ' Manipulação de ficheiros
Imports System.Text              ' Manipulação de strings: encoding

Public Class Form1
    ' Declaração de variáveis globais
    Dim Tcp_Servidor As New TcpListener(81)          ' TCP Listener
    Dim Tcp_Client As New TcpClient                 ' Cliente TCP
    Dim msg As NetworkStream                         ' leitura e envio de array de bytes através de TCP/IP
    Dim caracteres(4000) As Byte
    Dim ncar As Integer
    Dim read_aux As String

    'Faconsrv
    Dim fs As New FaconSvr.FaconServer

    Dim Y0, Y1, Y2, Y3, X0, X1, X2, X3, Nivel As String

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Definição valores iniciais
        tmr_com.Interval = 1000
        tmr_com.Enabled = True
        Tcp_Servidor.Start()

        ' Abertura dos dados do FaconSvr
        fs.OpenProject("E:\Aula7_2016.fcs")
    End Sub

    Private Sub PLC()
        ' Actualiza a visualização das entradas e saídas
        If Y0 = "1" Then chk_Y0.Checked = True Else chk_Y0.Checked = False
        If Y1 = "1" Then chk_Y1.Checked = True Else chk_Y1.Checked = False
        If Y2 = "1" Then chk_Y2.Checked = True Else chk_Y0.Checked = False

        If X0 = "1" Then chk_X0.Checked = True Else chk_X0.Checked = False
        If X1 = "1" Then chk_X1.Checked = True Else chk_X1.Checked = False
        If X2 = "1" Then chk_X2.Checked = True Else chk_X2.Checked = False
        If X3 = "1" Then chk_X2.Checked = True Else chk_X3.Checked = False

        txt_nivel.Text = Nivel
    End Sub

    Private Sub tmr_com_Tick(sender As Object, e As EventArgs) Handles tmr_com.Tick
        ' Leitura do estado das variáveis do PLC
        Y0 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "Y0")))
        Y1 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "Y1")))
        Y2 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "Y2")))

        X0 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "X0")))
        X1 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "X1")))
        X2 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "X2")))
        X3 = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "X3")))

        Nivel = CStr(CInt(fs.GetItem("Channel0.Station0.Group0", "D4072")))

        ' Visualiza estado das entradas e saídas
        PLC()
    End Sub

    ' Aguarda pedido de ligação
    If Tcp_Servidor.Pending = True Then
        ' Aceita ligação
        Tcp_Client = Tcp_Servidor.AcceptTcpClient
        msg = Tcp_Client.GetStream           ' Leitura da msg recebida em format networkstream
    End If

    txt_Estado.Text = Tcp_Client.Connected      ' Leitura do estado de ligação

```

```

If Tcp_Client.Connected Then
    ncar = Tcp_Client.Available
    If ncar > 0 Then
        msg.Read(caracteres, 0, ncar)      ' Leitura dos dados recebidos

        ' conversão do array de bytes "caracteres", numa string "texto recebido"
        read_aux = Encoding.Default.GetChars(caracteres)
        read_aux = Mid(read_aux, 1, 40)

        txt_recebido.Text = txt_recebido.Text + read_aux + vbCrLf

        ' Verifica a existência de comandos
        If InStr(read_aux, "Ligar_Y0") Then
            fs.SetItem("Channel0.Station0.Group0", "Y0", "1")
        End If
        If InStr(read_aux, "Desligar_Y0") Then
            fs.SetItem("Channel0.Station0.Group0", "Y0", "0")
        End If

        If InStr(read_aux, "Ligar_Y1") Then
            fs.SetItem("Channel0.Station0.Group0", "Y1", "1")
        End If
        If InStr(read_aux, "Desligar_Y1") Then
            fs.SetItem("Channel0.Station0.Group0", "Y1", "0")
        End If

        If InStr(read_aux, "Ligar_Y2") Then
            fs.SetItem("Channel0.Station0.Group0", "Y2", "1")
        End If
        If InStr(read_aux, "Desligar_Y2") Then
            fs.SetItem("Channel0.Station0.Group0", "Y2", "0")
        End If

        ' Identificação de palavras chave
        If InStr(read_aux, "MotorOn.html") Then
            EnviarFicheiro("MotorOn.html")
        ElseIf InStr(read_aux, "MotorOff.html") Then
            EnviarFicheiro("MotorOff.html")
        ElseIf InStr(read_aux, "Controlo.html") Then
            EnviarFicheiro("Controlo.html")
        ElseIf InStr(read_aux, "Supervisao.html") Then
            EnviarFicheiro("Supervisao.html")
        ElseIf InStr(read_aux, "Reservatorio.html") Then
            EnviarFicheiro("Reservatorio.html")
        ElseIf InStr(read_aux, "Logo.html") Then
            EnviarFicheiro("Logo.html")
        ElseIf InStr(read_aux, "Rodape.html") Then
            EnviarFicheiro("Rodape.html")
        End If

        Tcp_Client.Close()
        Tcp_Client = New TcpClient
        ncar = 0
        read_aux = ""
    End If
End If
End Sub

Private Sub EnviarFicheiro(ByVal sss As String)
    If Tcp_Client.Connected Then
        Dim ler_ficheiro As StreamReader
        ler_ficheiro = New StreamReader(sss)

        Dim ss As String
        ss = ler_ficheiro.ReadToEnd()

        ' Actualiza o valor das entradas e saídas digitais e do nível se o ficheiro
        ' contém essas variáveis
        ss = Replace(ss, "XXX0", X0)
        ss = Replace(ss, "XXX1", X1)
        ss = Replace(ss, "XXX2", X2)
        ss = Replace(ss, "XXX3", X3)

        ss = Replace(ss, "YYY0", Y0)
        ss = Replace(ss, "YYY1", Y1)
        ss = Replace(ss, "YYY2", Y2)
        ss = Replace(ss, "NNN0", Nivel)

        Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss)
        msg.Write(arraybytes, 0, Len(ss))
    End If
End Sub

```

```

        ler_ficheiro.Close()
        ss = ""
    End If
End Sub

End Class

```

Reservatório.html

```

<html>
<head>
<title>
Reservatorio
</title>
</head>
<frameset rows="*,50%">
<!frame superior />
    <frameset cols="30%,*,30%">
        <frame src="Logo.html" name="Esq" id="Esq" />
        <frame src="Controlo.html" name="Centro" id="Centro" />
        <frame src="Supervisao.html" name="Dir" id="Dir" />
    </frameset>
<frame src="Rodape.html" name="baixo" id="baixo" />
</frameset>
</html>

```

```

1  <html>
2  <head>
3  <title>
4  Reservatorio
5  </title>
6  </head>
7  <frameset rows="*,50%">
8      <!frame superior />
9      <frameset cols="30%,*,30%">
10         <frame src="Logo.html" name="Esq" id="Esq" />
11         <frame src="Controlo.html" name="Centro" id="Centro" />
12         <frame src="Supervisao.html" name="Dir" id="Dir" />
13     </frameset>
14     <frame src="Rodape.html" name="baixo" id="baixo" />
15 </frameset>
16 </html>

```

Logo.html

```

<html>
<head>
</head>
<body>

</body>
</html>

```

```

1  <html>
2  <head>
3  </head>
4  <body>
5      
6  </body>
7  </html>

```

Controlo.html

```

<html>
<head>
</head>
<body>
<b> Reservatório</b> <br> <br>

```

```

<form>
Controlo Motor (Y0) <br>
<input type="submit" Name="Y0" Value="Ligar_Y0">
<input type="submit" Name="Y0" Value="Desligar_Y0">
<br>
Válvula Entrada Água (Y1) <br>
<input type="submit" Name="Y1" Value="Ligar_Y1">
<input type="submit" Name="Y1" Value="Desligar_Y1">
<br>
Válvula Saída Água (Y1) <br>
<input type="submit" Name="Y2" Value="Ligar_Y2">
<input type="submit" Name="Y2" Value="Desligar_Y2">
<br>
</form>
<a href="MotorOn.html"> * Desligar Motor </a>
</body>
</html>

```

```

1   1 <html>
2   2   <head>
3   3     </head>
4   4   <body>
5   5     <b> Reservatório</b> <br> <br>
6   6   <form>
7   7     Controlo Motor (Y0) <br>
8   8       <input type="submit" Name="Y0" Value="Ligar_Y0">
9   9       <input type="submit" Name="Y0" Value="Desligar_Y0">
10 10      <br>
11 11     Válvula Entrada Água (Y1) <br>
12 12       <input type="submit" Name="Y1" Value="Ligar_Y1">
13 13       <input type="submit" Name="Y1" Value="Desligar_Y1">
14 14      <br>
15 15     Válvula Saída Água (Y1) <br>
16 16       <input type="submit" Name="Y2" Value="Ligar_Y2">
17 17       <input type="submit" Name="Y2" Value="Desligar_Y2">
18 18      <br>
19 19    </form>
20 20    <a href="MotorOn.html"> * Desligar Motor </a>
21 21  </body>
22 22  </html>
23 23

```

Supervisão.html

```
<html>
<head>
<meta http-equiv="refresh" content="1">
<meta charset="ISO-8859-1">
</head>
<body>
<br>
<br>
Supervisão <br>
Motor (Y0)
<span style="margin-left:3em">: YYY0 </span><br>
Val IN (Y1)
<span style="margin-left:3em">: YYY1 </span><br>
Val Out (Y2)
<span style="margin-left:3em">: YYY2 </span><br>
Nível Min Al(X0)
<span style="margin-left:3em">: XXX0 </span><br>
Nível Min (X1)
<span style="margin-left:3em">: XXX1 </span><br>
Nível Max (X2)
<span style="margin-left:3em">: XXX2 </span><br>
Nível Max Al (X3)
<span style="margin-left:3em">: XXX3 </span><br>
Nível (D4072)
<span style="margin-left:3em">: NNN0 </span><br>
</body>
</html>
```

```
1  □<html>
2  □<head>
3  <meta http-equiv="refresh" content="1">
4  <meta charset="ISO-8859-1">
5  </head>
6  □<body>
7  <br>
8  <br>
9  Supervisão <br>
10 Motor (Y0)
11 <span style="margin-left:3em">: YYY0 </span><br>
12 Val IN (Y1)
13 <span style="margin-left:3em">: YYY1 </span><br>
14 Val Out (Y2)
15 <span style="margin-left:3em">: YYY2 </span><br>
16 Nível Min Al(X0)
17 <span style="margin-left:3em">: XXX0 </span><br>
18 Nível Min (X1)
19 <span style="margin-left:3em">: XXX1 </span><br>
20 Nível Max (X2)
21 <span style="margin-left:3em">: XXX2 </span><br>
22 Nível Max Al (X3)
23 <span style="margin-left:3em">: XXX3 </span><br>
24 Nível (D4072)
25 <span style="margin-left:3em">: NNN0 </span><br>
26 </body>
27 </html>
28
```

Rodape.html

```
<html>
<body>
Inf Industrial 2016 - A Borges;
</body>
</html>
```

```
1  □<html>
2  □<body>
3  Inf Industrial 2016 - A Borges;
4  </body>
5  </html>
```



CAPÍTULO

11

BASES DE DADOS

JPSantos
2023

12. BASES de DADOS

12.1. Conceitos

De uma forma simplista poderíamos dizer que uma base de dados mais não é que um repositório de dados. Estes dados podem ser organizados num conjunto de tabelas, e as tabelas podem estar relacionadas entre si. Para gerir e armazenar estas tabelas e em disco podem ser utilizados programas conhecidos por “gestores de base de dados”.

Sistema de gestão de bases de dados - SGBD

Existem vários gestores de bases de dados comerciais, ou em inglês *DBMS - Database Management System*, cada um deles armazena os dados em disco de uma forma específica, mas em geral todos eles conhecem e aceitam pedidos realizados por outros programas, desde que estes pedidos sejam escritos de acordo com a linguagem *SQL (Structured Query Language)*.

Os programas que iremos desenvolver em VBASIC também podem fazer pedidos a estes gestores de bases de dados e enviar-lhes mensagens de texto escritas de acordo com a linguagem “SQL”.

Structured Query Language - SQL

A linguagem *SQL* permite consultar uma base de dados, inserir dados ou alterar os dados nela armazenados. Esta linguagem foi desenvolvida nos anos 70 pela IBM para aceder e gerir bases de dados relacionais. Foi posteriormente normalizada em 1986 pelo instituto *ANSI (American National Standards Institute)* sob a designação ANSI-92 SQL.

Um exemplo de uma pergunta/comando SQL é “*Select * from Motor*”. O gestor de bases de dados que receber este comando devolve todos os registos/linhas e campos/colunas da tabela *Motor*. Em Visual Basic existem diversos objetos, com diversos métodos que permitem enviar os comandos SQL para os gestores de bases de dados e guardar as respostas a essas perguntas.

Learn SQL in 1 Hour - SQL Basics for Beginners
<https://www.youtube.com/watch?v=9Pzj7Aj25lw>

Modelo relacional

Uma base de dados relacional não é mais que um conjunto de relações (tabelas) relacionadas entre si. Cada relação (tabela, entidade) pode ser armazenada em disco num ficheiro. Não confundir relação com relacionamento entre relações. Cada relacionamento entre relações pode dar origem a uma nova tabela, um novo ficheiro, com atributos de ambas as tabelas que se pretende relacionar.

Cada linha da tabela pode ser designada por *registro, record ou tuplo*.

Cada coluna da tabela contém *atributos, campos ou fields* dos registos.

Ao número de atributos de uma relação dá-se o nome de *grau da relação* e

Ao número de registo de uma relação dá-se o nome de *cardinalidade* da relação.

A *chave primária* de cada relação de uma base de dados relacional consiste num atributo ou num conjunto de atributos dessa relação que permite identificar de forma biunívoca um só registo. Por exemplo o número do B.I., ou o número mecanográfico, são atributos únicos, não existem dois registos, relativos a dois alunos que tenham o mesmo número mecanográfico. O número mecanográfico é por isso uma boa chave primária de uma relação.

Projeto de uma base de dados normalizada

Uma base de dados deve ser projetada de forma a satisfazer um conjunto de requisitos, a base de dados:

- 1- Deve ter a capacidade necessária para armazenar todos os dados, atributos, necessários à empresa;
- 2- Pode ter dados duplicados mas não dados redundantes;
- 3- Deve conter o menor número possível de relações;
- 4- Deve ter todas as relações normalizadas para evitar problemas futuros na atualização, remoção ou atualização de dados.

Um dado pode estar duplicado em duas relações mas ainda assim não ser redundante. Por exemplo, se esse dado, atributo, permitir o relacionamento entre duas relações ele pode estar presente em duas relações mas ainda assim ser indispensável. Contudo, os dados duplicados e redundantes têm de ser evitados.

Dados duplicados mas não redundantes

Os nomes João e Silva aparecem duplicados na tabela seguinte, mas esses dados são necessários.

NºEmplegado	NomeSupervisor
125	João
138	Silva
195	Silva
200	João

Tabela com dados duplicados mas não redundantes

Pois se fossem apagados (tabela seguinte) perdia-se informação útil e deixava de se saber quem eram os supervisores dos empregados nº 195 e nº 200

NºEmplegado	NomeSupervisor
125	João
138	Silva
195	-
200	-

Falta de informação nesta tabela

Dados redundantes

Os telefones dos supervisores João e Silva estão duplicados na tabela seguinte e são também redundantes.

NºEmplegado	NomeSupervisor	TelSup
125	João	3051
138	Silva	2222
195	Silva	2222
200	João	3051

Tabela com dados redundantes

Podemos observar na tabela seguinte que se os números de telefones duplicados na tabela fossem eliminados continuava a ser possível saber quais eram os números de telefone dos supervisores. Contudo, a tabela passava a ter campos vazios o que levantaria problemas na consulta e tratamento dos dados.

NºEmplegado	NomeSupervisor	TelSup
125	João	3051
138	Silva	2222
195	Silva	-
200	João	-

Tabela sem dados redundantes mas com campos nulos

Dados duplicados mas não redundantes

A solução para não existirem campos redundantes na tabela passa pela sua subdivisão em duas novas tabelas. As duas tabelas seguintes contêm toda a informação da anterior sem dados redundantes ou campos nulos.

NºEmplegado	NomeSupervisor
125	João
138	Silva
195	Silva
200	João

NomeSupervisor	TelSup
João	3051
Silva	2222

Ao invés de usar uma só relação com todos os dados relativos a uma empresa é preferível subdividir essa *relação universal* em várias relações, cada uma com menor cardinalidade e grau pois é mais fácil de gerir e evita problemas futuros na gestão da base de dados. A esse processo de subdivisão uma tabela universal em várias tabelas, de acordo com um conjunto de regras que iremos abordar nas secções seguintes, dá- se o nome de *normalização da base de dados*.

Uma relação não Universal

NºAluno	NomeAluno	NºQuarto	TelQuarto	Cadeira	Semestre	Nota
3215	João	120	2136	MAT122	SET84	6.4
				CIE120	SET84	9.6
				FIS230	JUN85	8.4
				MAT122	JUN85	9.2
3462	Silva	238	2344	MAT122	JUN84	9.2
				MAT123	JUN85	14
				PSI220	JUN85	14.8
3567	Oscar	120	2136	CIE239	JUN84	13.2
				EGR171	SET84	14
				FIS141	SET84	7.2
4756	Alex	345	3321	MUS389	SET83	16

Uma relação Universal

NºAluno	NomeAluno	NºQuarto	TelQuarto	Cadeira	Semestre	Nota
3215	João	120	2136	MAT122	SET84	6.4
3215	João	120	2136	CIE120	SET84	9.6
3215	João	120	2136	FIS230	JUN85	8.4
3215	João	120	2136	MAT122	JUN85	9.2
3462	Silva	238	2344	MAT122	JUN84	9.2
3462	Silva	238	2344	MAT123	JUN85	14
3462	Silva	238	2344	PSI220	JUN85	14.8
3567	Oscar	120	2136	CIE239	JUN84	13.2
3567	Oscar	120	2136	EGR171	SET84	14
3567	Oscar	120	2136	FIS141	SET84	7.2
4756	Alex	345	3321	MUS389	SET83	16

Normalização da base de dados

As tabelas/relações de uma base de dados podem estar na primeira forma normal (1FN), segunda forma normal (2FN), na terceira forma normal (3FN) ou na Forma Normal de Boyce Codd.

1FN: Atomicidade, cada atributo tem valores indivisíveis/atómicos. Não há grupos de valores repetidos.

2FN: cada tabela da BD, além de estar na 1FN, sem dependências parciais. Todos os atributos de cada tabela devem estar dependentes de parte dos atributos da chave primária. Elimina dependências parciais dos atributos a partes da chave primária.

3FN: Todos os atributos de uma tabela dependem inequivocamente da chave primária, sem dependências transitivas. Podem ter duas ou mais chaves candidatas, atributos em comum nas chaves candidatas.

Dependência transitiva: quando um atributo da tabela não depende da chave da tabela mas de outro da mesma tabela que não é parte da chave primária.

Dependência multivalorada:

FNBC : os únicos determinantes da tabela são a chave primária. Todos os atributos da tabela dependem da sua chave primária.

Diagramas para representar a base de dados e auxiliar na sua normalização

Existem vários diagramas, símbolos e regras para representar uma base de dados. Por exemplo, [Diagramas de Dependências funcionais \(DFD\)](#), [Diagramas de Entidade Relacionamento \(DER\)](#), e mais recentemente os [diagramas de classes da Unified Modelling Language \(UML - Diagrama de classes\)](#)

Nas secções seguintes serão apresentados os três tipos de diagramas e regras

12.2. Diagramas de Dependências Funcionais

Na secção anterior descreveram-se alguns conceitos relativos a bases de dados relacionais, nomeadamente a importância de projetar uma base de dados que esteja normalizada. Põe-se no entanto a dificuldade de saber subdividir uma relação universal em várias relações que estejam devidamente normalizadas. O método dos *DDF - Diagrama de Dependências Funcionais* entre atributos de uma relação permite ajudar a normalizar essas relações.

Para explicar o que é uma *dependência funcional - DF*, um diagrama de dependências funcionais e de que forma um DDF pode ser usado para normalizar uma relação universal, gerando um conjunto de sub-relações normalizadas, vamos apresentar um exemplo concreto e passo a passo apresentar esses conceitos e definições.

12.2.1. Exemplo - Normalização da BD Alunos

Pretende-se projetar uma base de dados normalizada para guardar dados relativos aos alunos de engenharia Mecânica, nomeadamente as cadeiras que frequentaram em cada semestre e a nota que obtiveram. Pretende-se também saber o quarto e a extensão telefónica, dos alunos que estão alojados nas residências universitárias. A relação universal foi apresentada na secção anterior e contém os atributos:

R(*NºAluno*, *Nome Aluno*, *Quarto*, *TelQuarto*, *Cadeira*, *Semestre*, *Nota*)

Como assumimos que em cada quarto “existe um e só um” telefone, se soubermos o número do quarto podemos saber o número de telefone de forma inequívoca. Existe por isso uma *dependência funcional* entre estes dois atributos. Quando existe uma dependência funcional entre atributos podemos desenhar no DDF uma “seta”, neste caso, entre o atributo *NºQuarto* e o atributo *TelQuarto*.

Cada “seta” num DDF pode ser lida como “existe um e só um”. Como um aluno só pode estar alojado “num e apenas num” quarto, podemos também desenhar uma seta, uma DF, entre os atributos *NºAluno* e *NºQuarto*. Pelo contrário, como num quarto podem estar alojados vários alunos não podemos afirmar que num quarto existe “um e apenas um aluno”, por essa razão a seta que relaciona os atributos *NºQuarto* e *NºAluno* tem apenas um sentido, o sentido $\langle N^{\circ}Aluno \rangle \rightarrow N^{\circ}Quarto$.

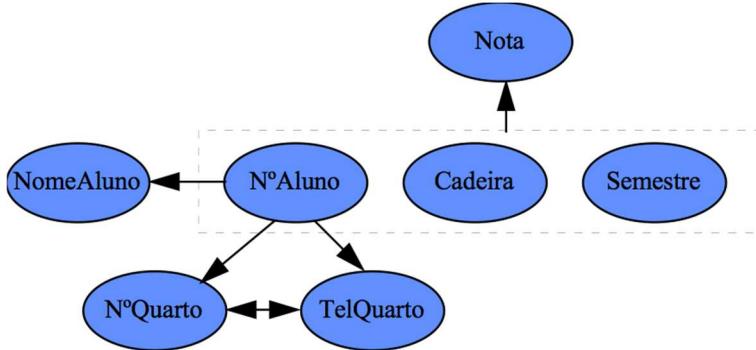
Assumimos também que cada aluno tem “um e só um” *NºAluno*, e por isso podemos desenhar uma “seta” com origem no atributo $\langle N^{\circ}Aluno \rangle$ e a terminar no atributo *NomeAluno*. Não podemos desenhar uma seta com o sentido oposto porque podem existir alunos diferentes com nomes iguais e nesse caso não podíamos afirmar que para um *NomeAluno* “existe um e só um” *NºAluno*.

Pretende-se também registrar na BD a(s) nota(s) que os alunos obtiveram a cada cadeira. Considere que um aluno pode levar vários semestres/anos para fazer uma disciplina ou pode voltar a inscrever-se no ano seguinte à mesma cadeira para melhorar a nota.

Por esta razão, **não** podemos afirmar que para um aluno “existe uma e apenas uma” nota. Também **não** podemos afirmar que um aluno “tem uma e apenas uma” nota a uma cadeira, mas podemos afirmar que um *Aluno* teve “uma e só uma” *Nota* a uma *Cadeira* numa determinada data “*Semestre*”. Quando é necessários usar um conjunto de atributos $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$, para determinar de forma inequívoca outro atributo, neste exemplo o atributo *Nota*, existe uma dependência funcional composta. Este tipo de DF pode ser representado num DDF através de um rectângulo a tracejado que envolve neste caso os atributos $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$ de onde tem origem uma seta com direção ao atributo *Nota*.

Podemos concluir que esta relação universal possui vários *determinantes*:
 $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$, $\langle N^{\circ}Aluno \rangle$, $\langle N^{\circ}Quarto \rangle$, $\langle TelQuarto \rangle$

E apenas um destes determinantes permite identificar de forma inequívoca todo e qualquer atributo, sendo por isso a *chave candidata* desta tabela/relação universal:
 $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$



R(NºAluno, NomeAluno, NºQuarto, TelQuarto, Cadeira, Semestre, Nota)

Determinantes:

1. $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$
2. $\langle N^{\circ}Aluno \rangle$
3. $\langle N^{\circ}Quarto \rangle$
4. $\langle TelQuarto \rangle$

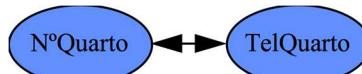
Chaves Candidatas:

1. $\langle N^{\circ}Aluno, Cadeira, Semestre \rangle$

A relação universal “R” não está na *Forma Normal de Boyce Codd – FNBC* dado que nem todos os determinantes da relação são chaves candidatas desta mesma relação. Por isso é necessário decompor a relação.

Numa primeira iteração vamos dividir a tabela universal nas tabelas R2 e R1.

R2



R2(NºQuarto, TelQuarto)

Determinantes:

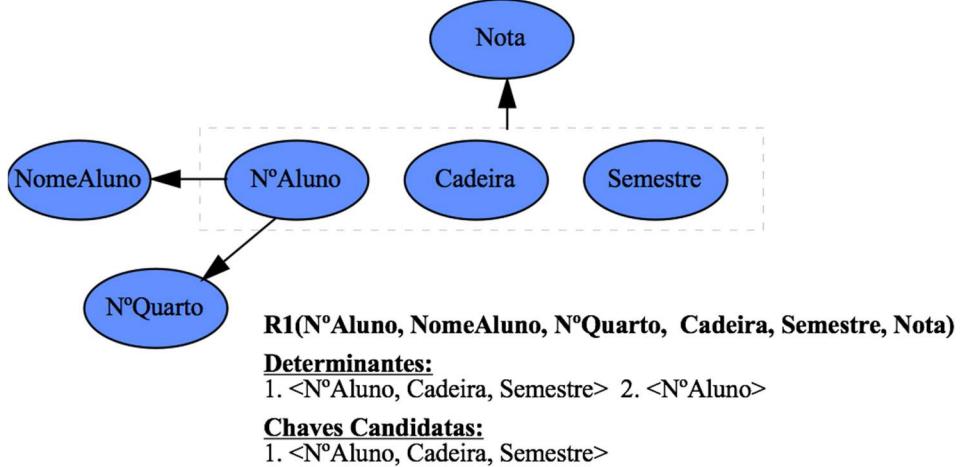
1. $\langle N^{\circ}Quarto \rangle$
2. $\langle TelQuarto \rangle$

Chaves Candidatas:

1. $\langle N^{\circ}Quarto \rangle$
2. $\langle TelQuarto \rangle$

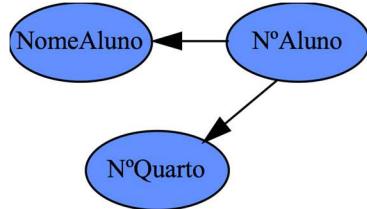
Podemos concluir que a sub relação R2 está normalizada pois todos os seus determinantes são também chaves candidatas.

R1



Pelo contrário, a relação R1 não está na FNBC pois tem mais determinantes que chaves candidatas. Por isso é necessário decompor esta relação nas relações R1.1 e R1.2

R1.1

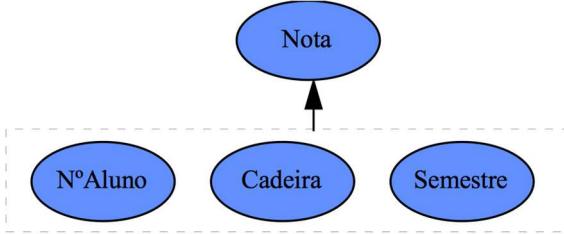


R1.1(NºAluno, NomeAluno, N°Quarto)

Determinantes:
1. <NºAluno>

Chaves Candidatas:
1. <NºAluno>

R1.2



R1.2(NºAluno, Cadeira, Semestre, Nota)

Determinantes:
1. <NºAluno, Cadeira, Semestre>

Chaves Candidatas:
1. <NºAluno, Cadeira, Semestre>

Verifica-se que as relações R1.1 e R1.2 estão normalizadas, todos os determinantes são chaves candidatas. Depois do processo de normalização concluído foram criadas 3 tabelas normalizadas R1.1, R1.2 e R2:

R1.1

NºAluno	NomeAluno	NºQuarto
3215	João	120
3462	Silva	238
3567	Oscar	120
4756	Alex	345

R1.2

NºAluno	Cadeira	Semestre	Nota
3215	MAT122	SET84	6.4
3215	CIE120	SET84	9.6
3215	FIS230	JUN85	8.4
3215	MAT122	JUN85	9.2
3462	MAT122	JUN84	9.2
3462	MAT123	JUN85	14
3462	PSI220	JUN85	14.8
3567	CIE239	JUN84	13.2
3567	EGR171	SET84	14
3567	FIS141	SET84	7.2
4756	MUS389	SET83	16

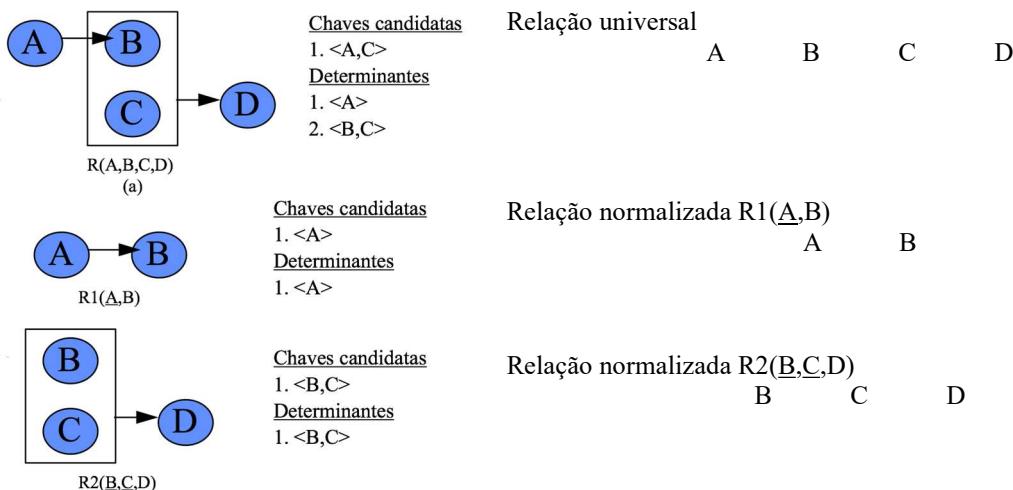
R2

NºQuarto	TelQuarto
120	2136
238	2344
345	3321

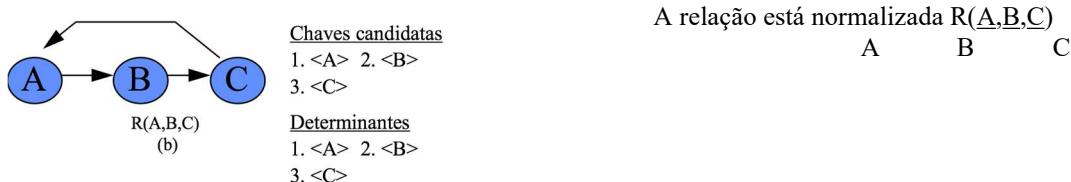
12.2.2. Exemplo - Normalização de algumas relações

Este exemplo apresenta as dependências funcionais de várias relações: a, b, c, d, e. Para cada relação identifique todos os determinantes e chaves candidatas. Determine quais as relações que se encontram na forma BCNF. Se uma relação não se encontrar na forma normal use o método de decomposição.

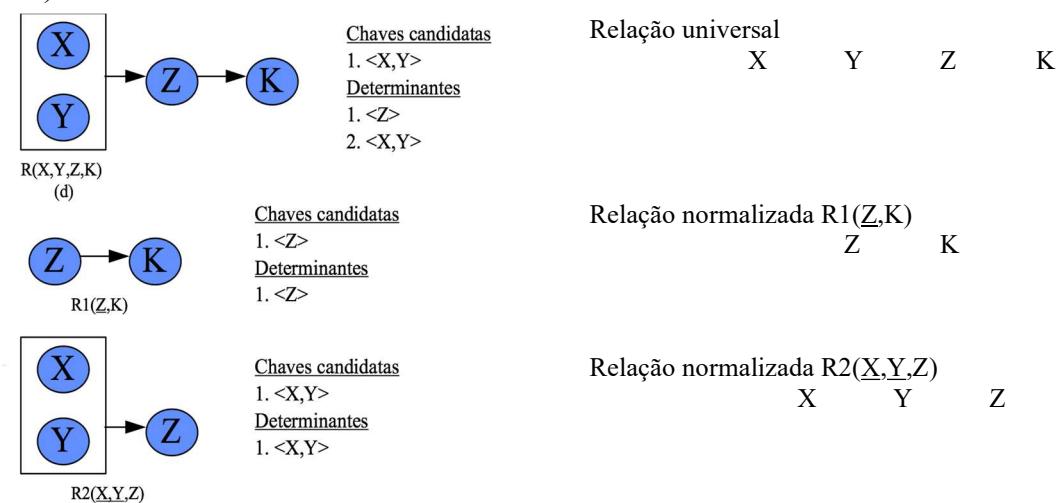
Relação a)



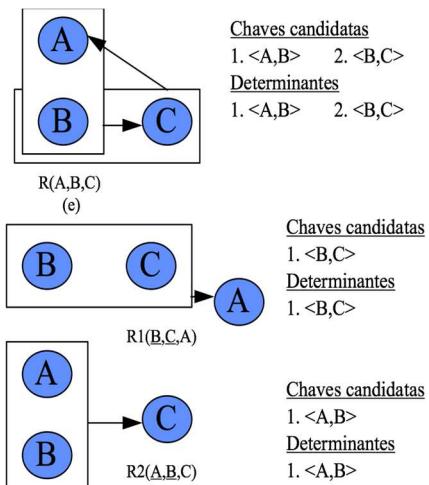
Relação b)



Relação c)

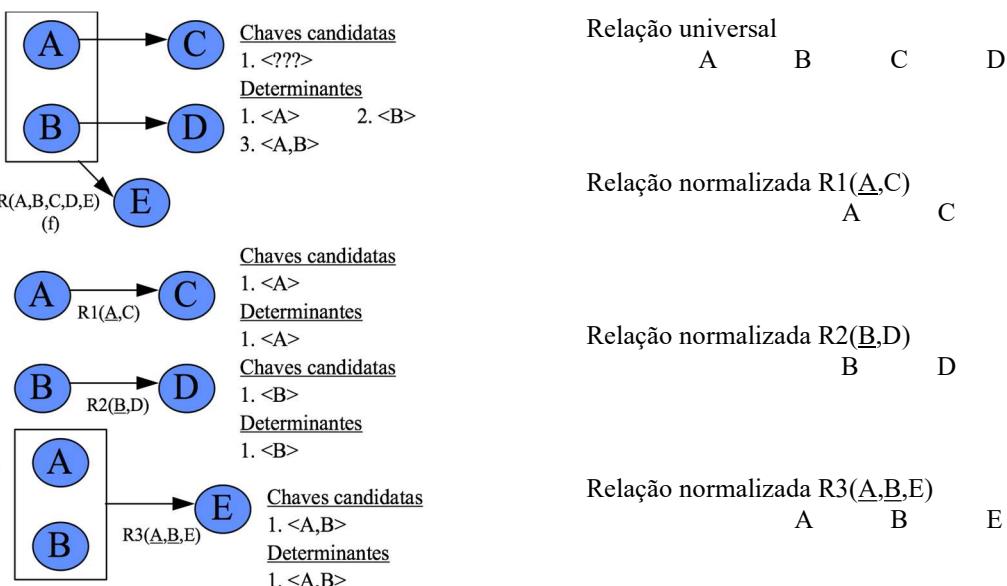


Relação d)



A relação inicial $R(A,B,C)$ já se encontra normalizada uma vez que todos os determinantes da relação são também chaves candidatas da relação. As supostas relações normalizadas da figura $R(B,C,A)$ e $R(A,B,C)$ são na realidade iguais à relação inicial.

Relação f)



12.2.3. Exemplo – Normalização de uma agenda telefónica

Neste exemplo pretende-se projetar uma BD normalizada para conter os números telefónicos de casa e do trabalho de cada empregado de uma empresa.

Assume-se que:

- cada empregado tem um e só um telefone em casa.
- cada empregado pode ser contactado no emprego em várias extensões.
- cada extensão serve vários empregados.

Atributos

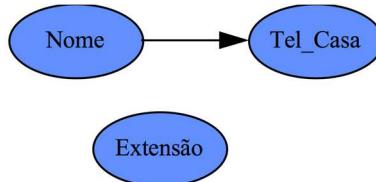
Nome: nome do empregado da empresa.

Tel_Casa: número de telefone de casa do empregado

Extensão: Extensão telefónica do empregado na empresa.

Relação Universal

R(Nome, Tel_Casa, Extensão)



R(Nome, Tel_Casa, Extensão)

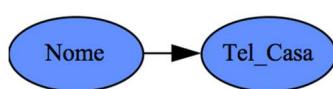
Determinantes:

1. <Nome>

Chaves Candidatas:

1. <Nome, Extensão>

Verifica-se que a relação universal não está normalizada porque os determinantes da relação não são chaves candidatas da relação.



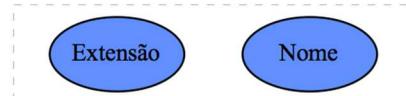
R1(Nome, Tel_Casa)

Determinantes:

1. <Nome>

Chaves Candidatas:

1. <Nome>



R2(Extensão, Nome)

Determinantes:

1. <Extensão>

Chaves Candidatas:

1. <Extensão>

A sub-relação R(Nome, Tel_Casa) encontra-se normalizada.

A segunda sub-relação R(Extensão, Nome) não tem verdadeiramente um “Determinante”, nem uma “Chave Candidata”, mas é a solução possível para relacionar esses atributos..

12.2.4. Exemplo – Normalização de BD de Clientes com conta aberta numa loja

Uma loja deseja desenvolver uma base de dados para armazenar informação sobre as contas dos seus clientes. Os dados a serem armazenados incluem para cada cliente: número da conta, nome, endereço, número de telefone, crédito (excelente, bom, fraco, mau), e o saldo. Desenhe o diagrama das dependências funcionais para os atributos envolvidos, apresente as pressuposições que assumir. Apresente as relações na forma BCNF para esta base de dados.

Assume-se que:

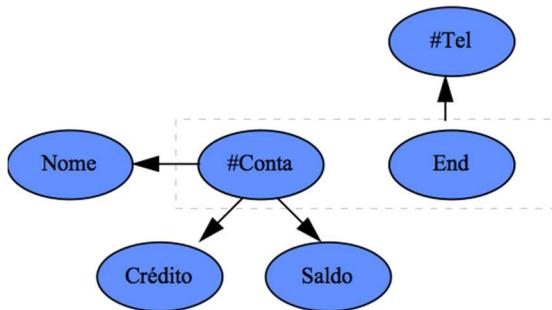
- Cada cliente pode ter várias moradas, mas em cada morada existe um e um só número de telefone.
- Cada cliente é identificado univocamente pelo número de conta.

Atributos:

- #Conta: número da conta do cliente na loja.
 Nome: nome do cliente, podem existir nomes iguais para clientes diferentes.
 End: morada do cliente, vários clientes podem ter a mesma morada.
 #Tel: número de telefone da morada do cliente.
 Credito: Crédito do cliente (excelente, bom, fraco, mau).
 Saldo: Valor da dívida do cliente para com a loja.

Relação Universal

$R(\#Conta, \text{Nome}, \text{End}, \#Tel, \text{Crédito}, \text{Saldo})$



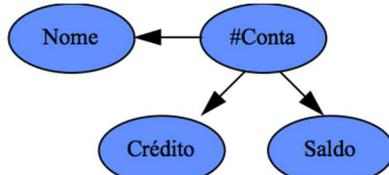
$R(\#Conta, \text{Nome}, \text{Crédito}, \text{Saldo}, \text{End}, \#Tel)$

Determinantes:

1. <#Conta>
2. <#Conta, End>

Chaves Candidatas:

1. <#Conta, End>



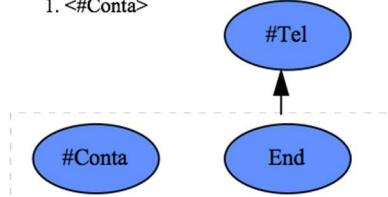
$R1(\#Conta, \text{Nome}, \text{Crédito}, \text{Saldo})$

Determinantes:

1. <#Conta>

Chaves Candidatas:

1. <#Conta>



$R2(\#Conta, \text{End}, \#Tel)$

Determinantes:

1. <#Conta, End>

Chaves Candidatas:

1. <#Conta, End>

12.2.5. Exemplo – Normalização de uma BD para uma biblioteca

Pretende-se desenvolver uma base de dados para ser usada numa biblioteca. A biblioteca pretende saber que livros estão emprestados e a quem.

Sabendo-se que um utilizador só tem uma categoria sabe-se o tempo máximo do empréstimo. Um utilizador não pode requisitar duas vezes no mesmo dia o mesmo livro. Interessa manter um registo ao longo do tempo dos empréstimos de cada livro. Essa base de dados deve armazenar a informação seguinte:

- Código do utilizador
- categoría
- Código do livro
- situação do livro (emprestado/devolvido)
- tempo máximo do empréstimo (nº dias)
- data de saída do livro
- data de devolução do livro
- data máxima de devolução do livro

Assume-se que:

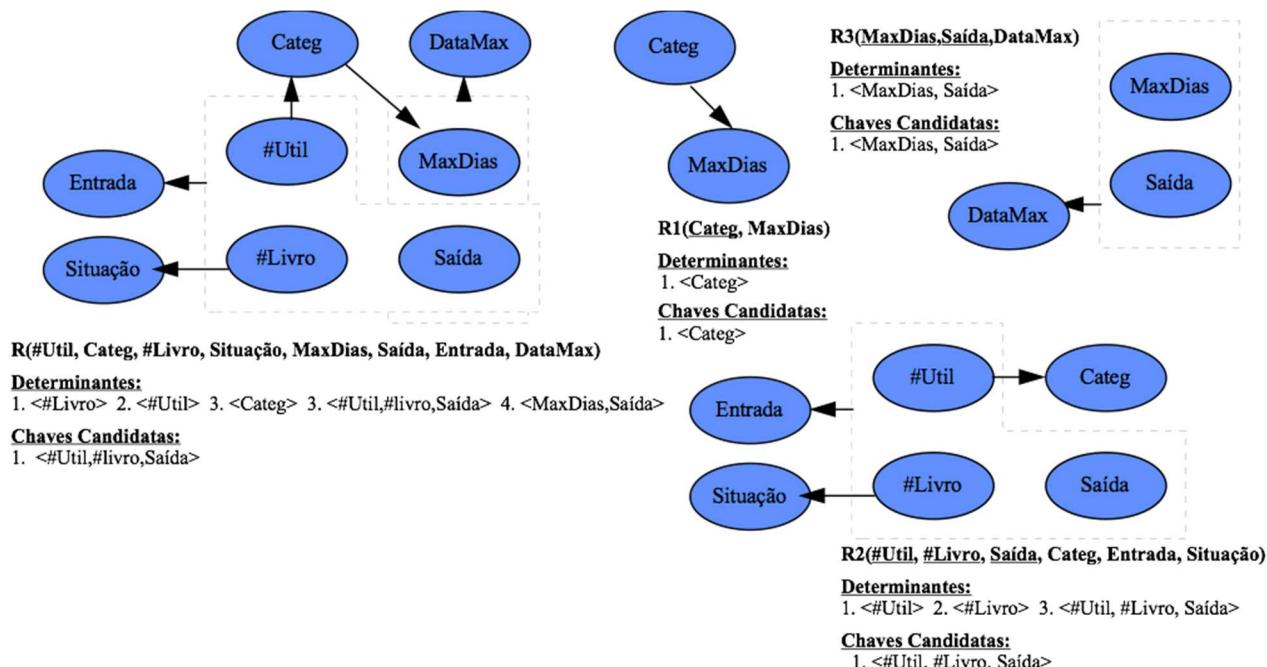
- O mesmo utilizador não pode requisitar duas vezes no mesmo dia o mesmo livro, mas utilizadores diferentes podem.

Atributos:

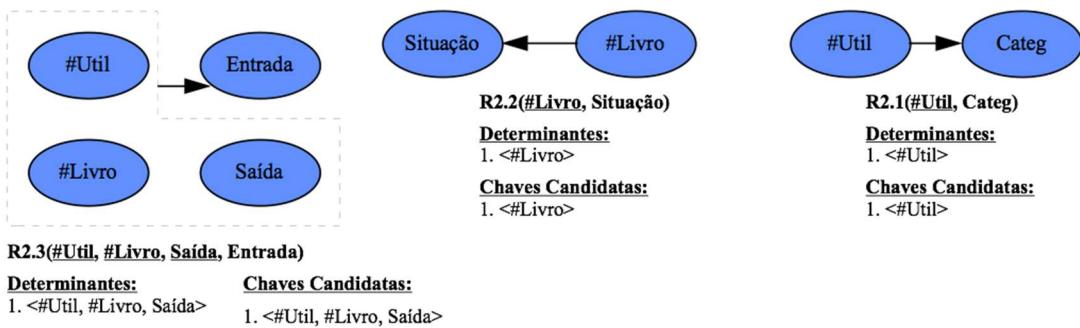
- #Util: código do utilizador
- Categ: categoria
- #Livro: código do livro
- Situação: situação do livro
- MaxDias: tempo máximo
- Saída: data saída do livro
- Entrada: data de devolução do livro
- DataMax: data máxima de devolução do livro

Relação Universal

$R(\#Util, Categ, \#Livro, Situação, MaxDias, Saída, Entrada, DataMax)$



A relação R2 não se encontra ainda na BCNF (3FN) por isso deve ser decomposta em R2.1, R2.2, R2.3.



A relação **R3(MaxDias, Saída, DataMax)** não é mais que uma tabela de somas. O programa que utilizar as outras relações pode calcular a **DataMax** com base na data de saída e na categoria do utilizador, não sendo por isso necessário implementar esta relação na base de dados.

Categ - R2.1(#Util, Categ,...)

Empréstimos-R2.3(#Util, #Livro, Saída, Entrada,...)

MaxDias- R1(Categ, MaxDias,...)

Livros- R2.2(#Livro, Situação,...)

12.2.6. Exemplo – Normalização de BD para um Vídeo Clube

Um clube de vídeo tem um conjunto de sócios. Cada um deles é identificado por um código (único). Nos ficheiros do clube encontra-se ainda registado o nome do sócio e a respectiva morada e telefone. Cada sócio pode autorizar até um máximo de cinco pessoas a utilizar os serviços do clube de vídeo. Para tal, cada uma destas pessoas recebe um código de identificação único. Não há repetições entre estes códigos e os códigos de sócio. O sócio é responsável por todos os vídeos que sejam emprestados às pessoas por ele autorizadas. Uma pessoa não pode ser autorizada a utilizar o clube por mais que um sócio.

O clube possui um número elevado de vídeos, estando atribuído um código único a cada um deles. Podem existir várias cópias do mesmo vídeo mas cada uma corresponderá a um código diferente. Qualquer vídeo deverá ser devolvido ao fim de três dias a seguir à data em que foi emprestado.

Resumindo, a informação que o clube terá sobre cada sócio é a seguinte:

Código de sócio, nome, morada, telefone, código das pessoas autorizadas por ele, detalhe dos vídeos emprestados e pelos quais este sócio é responsável.

Da informação que é armazenada no clube de vídeo emprestado destaca-se a seguinte:

Código do vídeo, código da pessoa que o tem, título do vídeo, realizador, data do empréstimo, e data limite para a devolução.

O clube também possui informação sobre todos os vídeos não emprestados.

- a) Estabeleça o diagrama de dependências funcionais DDF entre os vários dados. Apresente as premissas que assumir.
- b) Apresente a tabela na primeira forma normal 1FN correspondente ao DDF anterior. Indique também a chave primária associada a esta tabela.
- c) Apresente as tabelas na terceira forma normal 3FN correspondente ao DDF anterior.

Assume-se que:

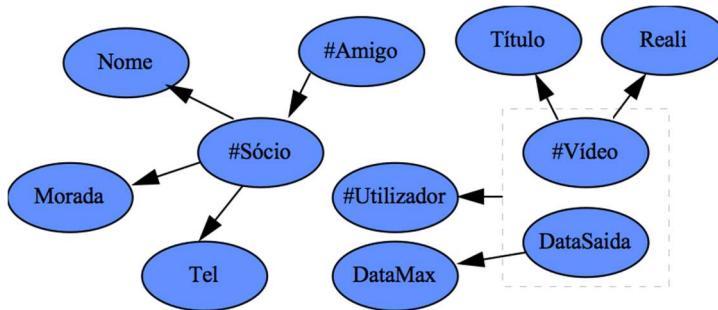
- Cada vídeo tem um e só um realizador

Atributos:

#Sócio: código sócio
Nome: nome
Morada: morada
Tel: telefone
#Amigo: código das pessoas autorizadas por ele
#Vídeo: código do vídeo
Título: título do vídeo
Reali: realizador
DataSaida: data do empréstimo
DataMax: data limite de devolução

Relação Universal

R(#Util, Categ, #Livro, Situação, MaxDias, Saída, Entrada, DataMax)



$R(\#Sócio, Nome, Morada, Tel, \#Amigo, \#Utilizador, \#Vídeo, Título, Reali, DataSaida, DataMax)$

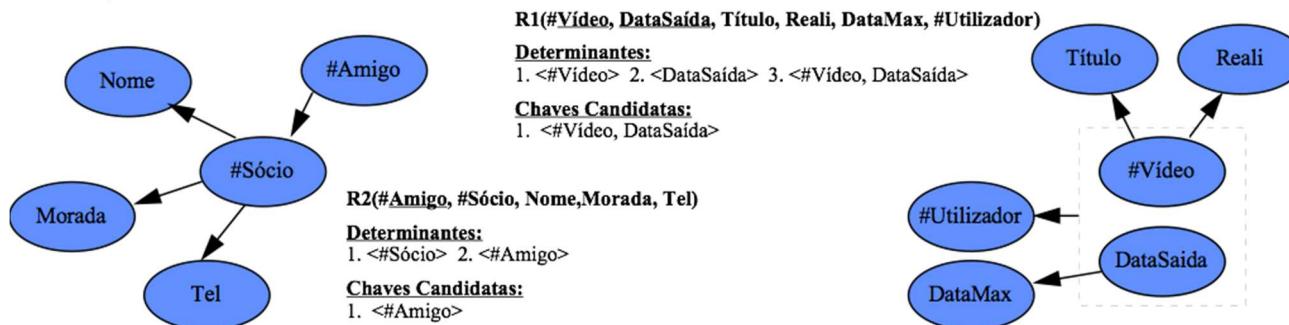
Determinantes:

1. <\#Sócio>
2. <\#Amigo>
3. <\#Vídeo, DataSaida>
4. <\#Vídeo>
5. <DataSaida>

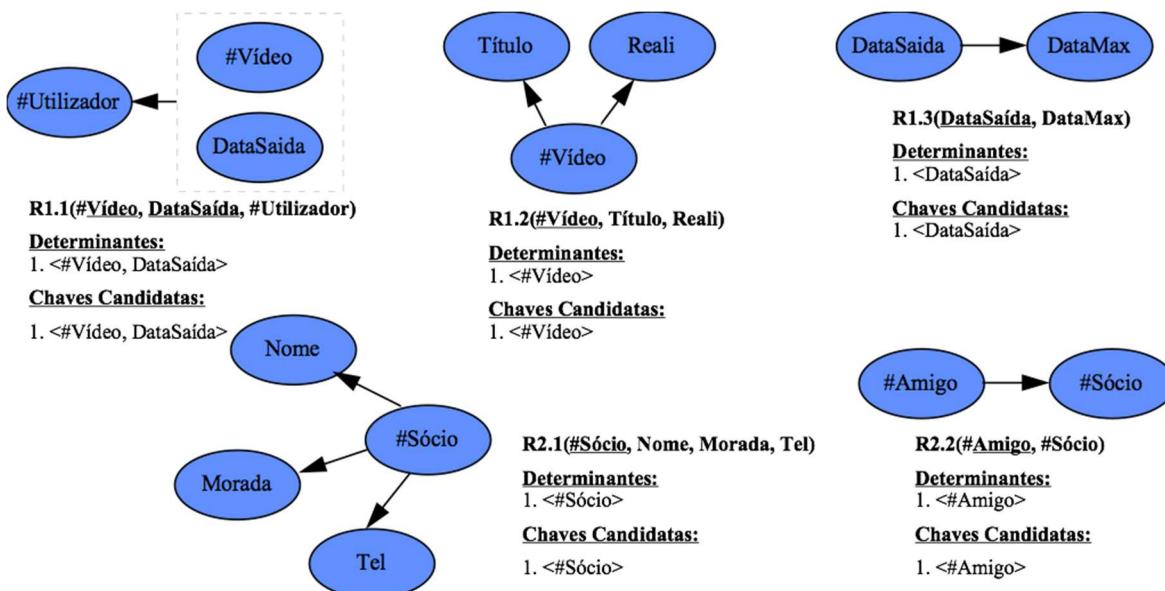
Chaves Candidatas:

1. <>

alínea b)

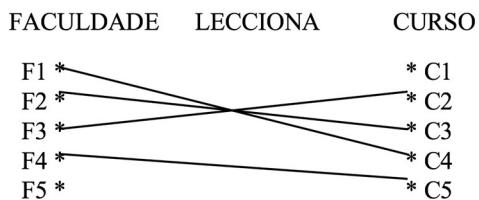


alínea c)



12.3. Diagramas de Entidade Relacionamento

12.3.1. Diagrama de Ocorrências

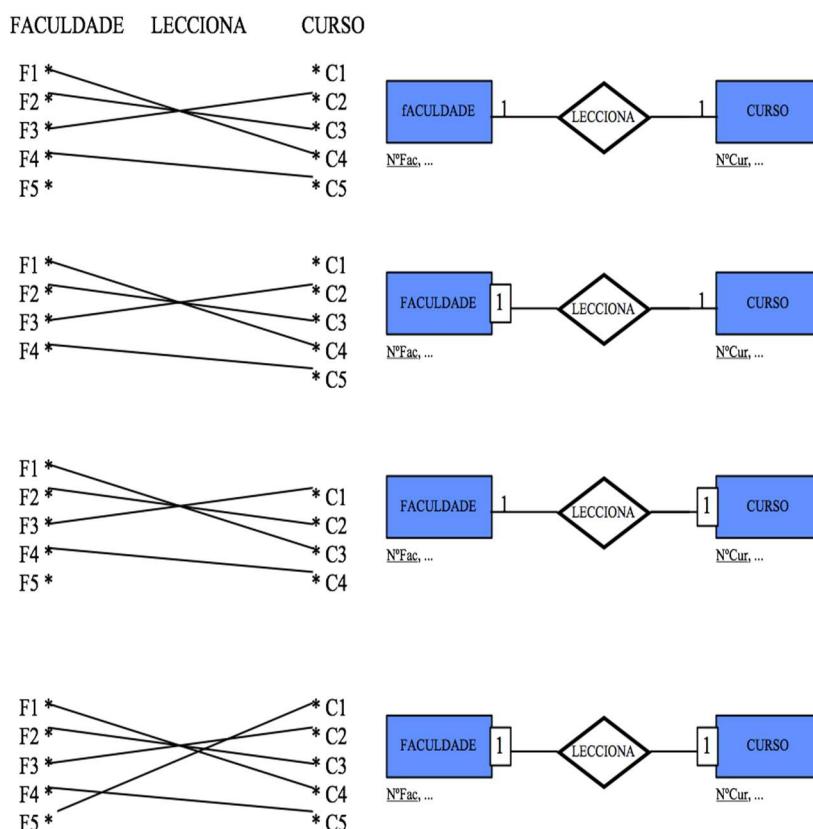


12.3.2. Diagrama de Entidade - Relacionamento



12.3.3. Relacionamentos binários 1:1

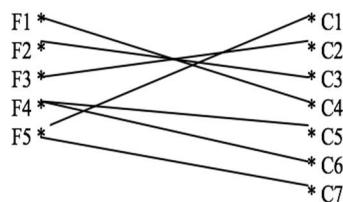
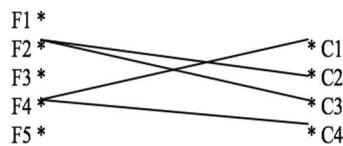
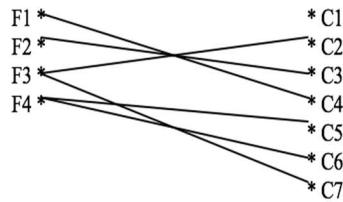
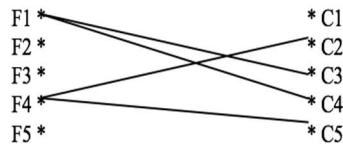
Com participação obrigatória ou não de cada Entidade



12.3.4. Relacionamentos binários 1:N

Com participação obrigatória ou não de cada Entidade

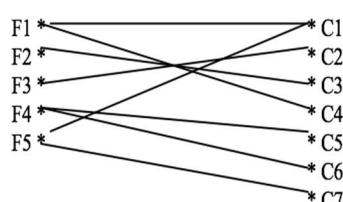
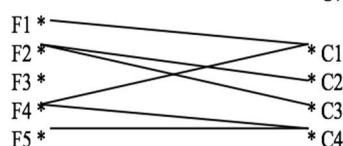
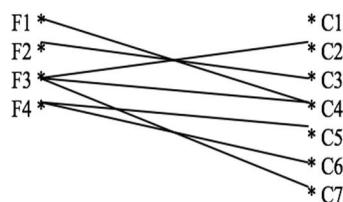
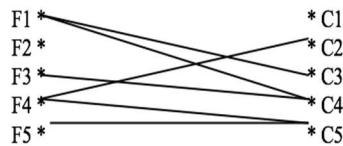
FACULDADE LECCIONA CURSO



12.3.5. Relacionamentos binários M:N

Com participação obrigatória ou não de cada Entidade

FACULDADE LECCIONA CURSO



Nas secções seguintes são apresentadas as relações necessárias para guardar os dados relativos a relacionamentos binários (entre duas relações apenas):

Grau 1:1

Participação de ambas as entidades obrigatória no relacionamento

Participação de uma só entidade obrigatória no relacionamento

Participação de nenhuma entidade obrigatória no relacionamento

Grau 1:N

Participação da entidade do lado N obrigatória no relacionamento

Participação de nenhuma entidade obrigatória no relacionamento

Grau N:M

Independentemente da participação de cada entidade no relacionamento

Serão também apresentadas as relações necessárias para relacionamentos de ordem superior, entre três ou mais entidades

Papéis desempenhados pela mesma entidade num dado relacionamento

12.3.6. Relações necessárias para relacionamentos binários de Grau 1:1

Participação de ambas as entidades obrigatória no relacionamento



Regra 1: Quando o grau de um relacionamento binário é de 1:1 com participação obrigatória das duas entidades, uma só relação é suficiente para armazenar todos os dados. Sendo a chave primária dessa relação a chave de qualquer das entidades.

Relações preliminares:

FACULDADE(NºFac, ..., NºCur, ...)

Relações:

FACULDADE(NºFac, NomeFac, EndFac, NºCur, NomeCurso)

NºFac	NomeFac	EndFac	NºCur	NomeCur
F1	Ciências	Rua de Santiago	C1	Electrónica
F2	Línguas	Quinta da Boavista	C2	Francês
F3	Medicina	Rua da Boa saúde	C3	Pediatria

Participação de uma só entidade obrigatória no relacionamento



Regra 2: Quando o grau de um relacionamento binário é de 1:1 com participação obrigatória de uma só entidade, duas relações são necessárias. Deve existir uma relação por cada entidade, com a chave de cada entidade a servir de chave primária da sua relação. Além disso a chave da entidade de participação não obrigatória tem que ser um **atributo da relação** correspondente à entidade de participação obrigatória.

Relações preliminares:

FACULDADE(NºFac, ...)

CURSO(NºCur, NºFac, ...)

Relações:

FACULDADE (NºFac, NomeFac, EndFac)

CURSO (NºCur, NomeCur, NºFac)

FACULDADE

NºFac	NomeFac	EndFac
F1	Ciências	Rua de Santiago
F2	Línguas	Quinta da Boavista
F3	Medicina	Rua da Boa Saúde

CURSO

NºCur	NomeCur	NºFac
C1	Electrónica	F1
C2	Francês	F2
C3	Pediatria	F3

Participação de nenhuma entidade obrigatória no relacionamento



Regra 3: Quando o grau de um relacionamento binário é de 1:1 sem participação obrigatória de nenhuma entidade, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da sua relação e uma para o relacionamento, fazendo parte dos seus atributos as chaves de cada uma das entidades.

Relações preliminares:

FACULDADE(NºFac, ...)

CURSO(NºCur, ...)

LECCIONA(NºFac,NºCur, ...)

12.3.7. Relações necessárias para relacionamentos binários de Grau 1:N

Participação da entidade do lado N obrigatória no relacionamento



Regra 4: Quando o grau de um relacionamento binário é de 1:N com participação obrigatória da entidade do lado N, duas relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente. Além disso a chave da entidade do lado 1 tem de ser um atributo da relação correspondente ao lado N.

Relações preliminares:

FACULDADE(NºFac, ...)

CURSO(NºCur, NºFac, ...)

Participação de nenhuma entidade obrigatória no relacionamento



Regra 5: Quando o grau de um relacionamento binário é de 1:N com participação não obrigatória da entidade do lado N, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento, tendo esta relação entre os seus atributos as chaves de cada uma das entidades.

Relações preliminares:

FACULDADE(NºFac, ...)

CURSO(NºCur, ...)

LECCIONA(NºFac,NºCur, ...)

12.3.8. Relações necessárias para relacionamentos binários de Grau N:M

Independentemente da participação de cada entidade no relacionamento



Regra 6: Quando o grau de um relacionamento binário é de N:M independentemente da participação obrigatória ou não das entidades, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento, tendo esta relação entre os seus atributos as chaves de cada uma das entidades.

Relações preliminares:

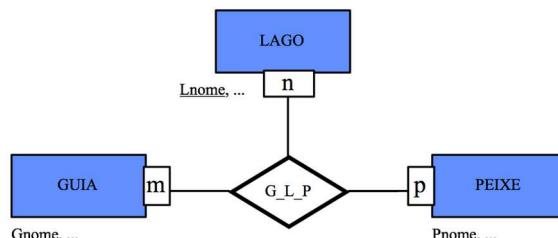
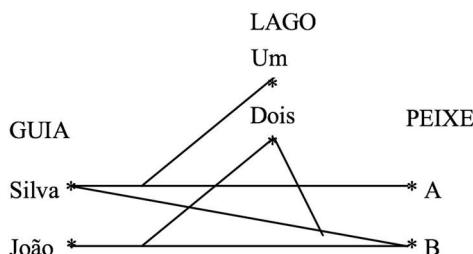
FACULDADE(NºFac, ...)

CURSO(NºCur, ...)

LECCIONA(NºFac,NºCur, ...)

12.3.9. Relações necessárias para relacionamentos de ordem superior

Relacionamentos entre três ou mais entidades



Regra 7: Quando o relacionamento é ternário, quatro relações preliminares são necessárias: uma para cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento. A relação gerada pelo relacionamento terá entre os seus atributos a chave de cada entidade.

Da mesma forma num relacionamento n-ário são necessárias n+1 relações.

Relações Preliminares:

GUIA(Gnome, ...)

PEIXE(Pname, ...)

LAGO(Lname, ...)

G_L_P(Gnome, Lname, Pname, ...)

Relações:

GUIA(Gnome, Tel, Folga, Max)

PEIXE(Pname, Peso, Eng)

LAGO(Lname, Rec)

G_L_P(Gnome, Lname, Pname)

12.3.10. Relações necessárias quando uma entidade tem vários papéis

Vários papéis desempenhados pela mesma entidade num dado relacionamento

Uma pequena empresa produtora de componentes automóvel deseja armazenar informação sobre os seus empregados. Existem duas classes de empregados: supervisores e montadores. Os supervisores pertencem ao quadro e recebem um salário mensal, enquanto os montadores são pagos à hora.

Numa primeira tentativa podíamos considerar o diagrama de ER seguinte em que a chave de cada entidade é o seu número da segurança social.

Assume-se:

- todos os empregados têm telefone em casa
- cada montador têm sempre um e só um supervisor
- cada supervisor supervisiona sempre um ou mais montadores

Atributos:

#SupSS:	número da segurança social do supervisor
#Mont:	número da segurança social do montador
Nome:	nome do empregado
TelTrab:	telefone no trabalho do supervisor
TelEmp:	telefone de casa do empregado
Morada:	morada do empregado
Hora:	valor pago à hora ao montador
Sal:	salário mensal do supervisor
Cóg:	código da tarefa do montador
Área:	área de competência do supervisor



Relações preliminares:

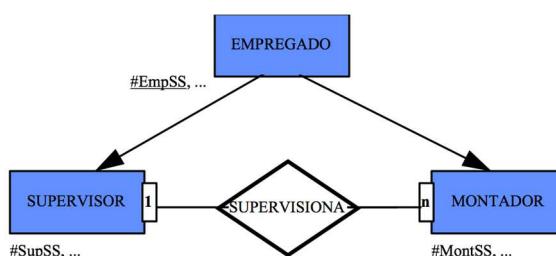
SUPERVISOR(#SupSS, ...)
MONTADOR(#MontSS, ... , #SupSS)

Relações:

SUPERVISOR(#SupSS, ...)
MONTADOR(#MontSS, ... , #SupSS)

Como atribuir atributos Nome, Morada, TelEmp, às relações ?

Criar novos atributos NomeSup, NomeMont, MoradaSup, MoradaEmp, TelSup, TelMont ?



Regra 8: a entidade fonte irá gerar uma relação, com a sua chave a servir como chave primária da relação. Cada papel assumido por esta entidade no relacionamento irá gerar o número de relações descritas nas regras anteriores onde cada papel assumido por esta entidade é tratado como uma entidade regular.

Assim três relações preliminares são necessárias:

EMPREGADO(#EmpSS, ...)
SUPERVISOR(#SupSS, ...)
MONTADOR(#MontSS, , #SupSS)

Relações:

EMPREGADO(#EmpSS, Nome, TelEmp, Morada)
SUPERVISOR(#SupSS, Salário, Área, TelTrab)
MONTADOR(#MontSS, Hora, Cód, #SupSS)

EXERCÍCIOS (DER)

12.3.11. Exemplo Casas e Pintores

Desenhe os diagramas de entidade ocorrência para cada uma das situações seguintes. Apresente os pressupostos que assumir para cada uma das situações:

Uma empresa de pinturas deseja relacionar os pintores e as casas que eles estiverem presentemente a pintar. Cada pintor pinta só uma casa de cada vez mas vários pintores podem simultaneamente pintar uma só casa. Alguns pintores podem estar desempregados. Só as casas que estiverem presentemente a serem pintadas devem ser consideradas.

Assume-se que:

- podem existir na base de dados pintores que não estejam presentemente a pintar nenhuma casa.
- todas as casas presentes na base de dados estão a ser pintadas por um ou por vários pintores.
- cada pintor só pinta uma casa de cada vez..

Atributos:

- #Pintor: código do pintor
MP: morada do pintor
TelP: telefone do pintor
#Casa: código da casa
MC: morada da casa

Diagrama de entidade - relacionamento



Para um relacionamento binário de grau 1:n em que a classe de participação do lado n não é obrigatória aplica-se a regra 5 para determinar as relações correspondentes.

PINTOR(#Pintor, MP, TelP)
CASA(#Casa, MC)
PINTA(#Pintor,#Casa)

12.3.12. Exemplo Garagem e Mecânicos

Uma garagem tem mecânicos a reparar automóveis. Cada mecânico trabalha em vários carros, mas cada carro é reparado por um só mecânico.

Desenhe os diagramas de Entidade Relacionamento para cada um dos diagramas de entidade ocorrência definidos no problema anterior e ainda as relações correspondentes.

Assume-se que:

- cada mecânico pode reparar vários carros.
- cada carro é reparado por um só mecânico.
- podem existir mecânicos que não tenham carros para reparar ou noutras alturas podem existir carros à espera de serem reparados.

Atributos:

- #Mec: código do mecânico.
NM: nome do mecânico.
Ext: extensão telefónica do mecânico na garagem.
#Entrada: código de entrada do carro.
ND: nome do dono do carro.
TelD: telefone do dono do carro.

Diagrama de entidade - relacionamento



Para um relacionamento binário de grau 1:n em que a classe de participação do lado n não obrigatória aplica-se a regra 5 para determinar as relações correspondentes.

MECÂNICO(#Mec, NM, Ext)
CARRO(#Carro, ND, TelD)
REPARA(#Mec,#Carro)

12.3.13. Exemplo Distribuidor , Lojas e Fornecedores

Uma empresa de distribuição deseja armazenar informação sobre lojas individuais e sobre os fornecedores de cada uma delas. Cada loja compra produtos fornecidos por vários fornecedores e cada fornecedor vende produtos para várias lojas.

Aos fornecedores é atribuído um código que os identifica univocamente e idêntico procedimento é usado para as peças. Os fornecedores são ainda caracterizados por designação e cor. As cores possíveis são preto, branco, cinzento e vermelho. Os fornecimentos são feitos a um dado preço e a uma dada data.

- Apresente o diagrama de entidade relacionamento correspondente.
- Apresente as relações correspondentes.

Solução:

Alínea a)

Assume-se que:

- todas as lojas que constam da base de dados têm pelo menos um fornecedor.
- todos os fornecedores que constam da base de dados vendem pelo menos para uma loja.

Atributos:

- | | |
|----------|------------------------|
| #Fornec: | código do fornecedor |
| MF: | morada do fornecedor |
| TelF: | telefone do fornecedor |
| #Loja: | código da loja |
| ML: | morada da loja |
| TelL: | telefone da loja |
| #Peça: | código da peça |

Diagrama de entidade - relacionamento



Para um relacionamento binário de grau n:m aplica-se a regra 6 para determinar as relações correspondentes. É de notar que essas relações não dependem da participação obrigatória ou não, destas duas classes no relacionamento.

FORNECEDORES(#Fornec, MF, TelF)
LOJA(#Loja, ML, TelL)
FORNECE(#Fornec, # Loja, #Peça)

12.3.14. Exemplo Empresa de Investigação

Uma empresa de investigação e desenvolvimento possui diversos tipos de empregados. Cada empregado possui um número de identificação único. Existem várias funções a desempenhar na empresa: investigador, técnico, contratado, estagiário e administrativo. Com excepção do pessoal administrativo todos os empregados estão envolvidos em projectos. Cada projeto visa desenvolver obras ou cursos (remunerados de forma diferente). Interessa saber em que projetos, obras e cursos os empregados se encontram envolvidos. Existem sempre um investigador responsável para cada projeto, para cada obra ou para cada curso respetivamente.

- a) Apresente o diagrama de entidade relacionamento correspondente, criando os atributos que considerar relevantes.
- b) Apresente as relações correspondentes.

Solução:

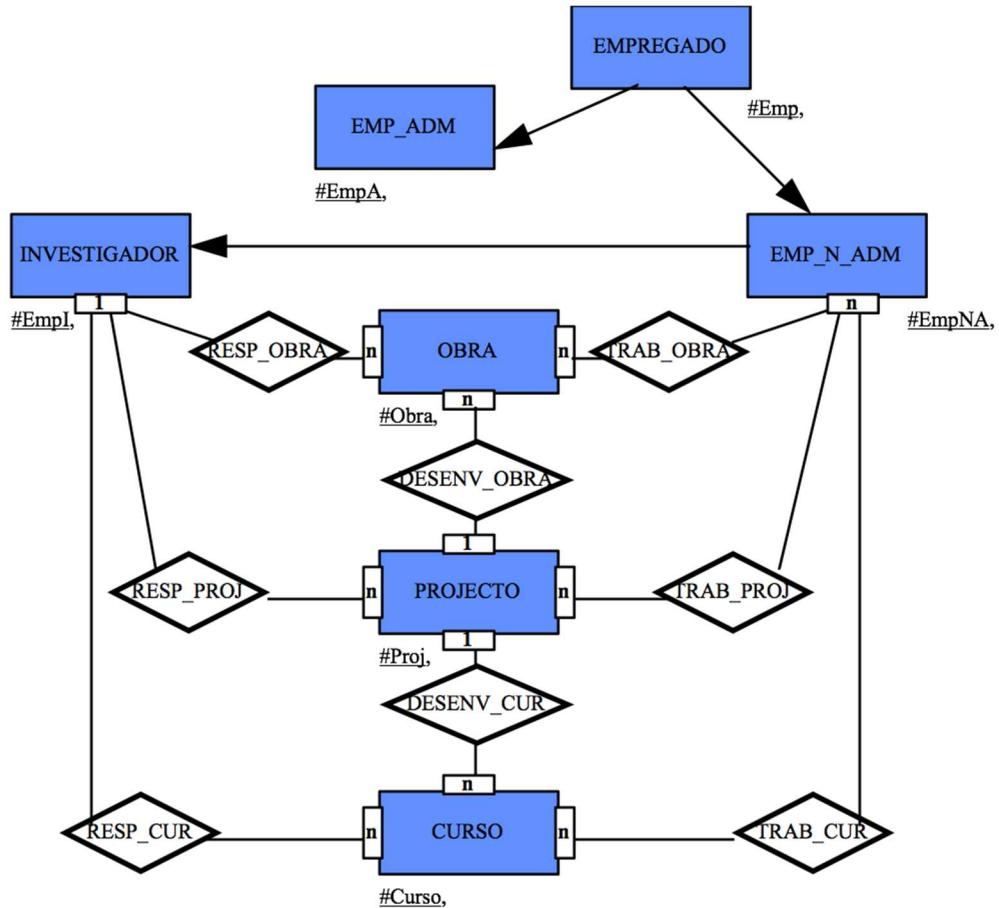
Assume-se que:

- Existe um e um só responsável por cada obra ou por cada curso.
- O mesmo investigador pode ser responsável por mais que um curso ou obra.
- Cada obra ou curso não é partilhado por mais que um projeto.

Atributos:

- #Emp: código do empregado
#EmpA: código do empregado (Administrativo)
#EmpNA: código do empregado (Investigador, técnico, contratado, ou estagiário)
#EmpI: código do empregado (Investigador)
N: nome do empregado.
M: morada do empregado
tel: telefone do empregado
#Obra: código da obra
OO: orçamento da obra
#Proj: código do projecto
OP: orçamento do projecto
#Curso: código do curso
OC: orçamento do curso

Diagrama de entidade - relacionamento



EMPREGADO(#Emp, N, M, Tel)
 EMP_ADMIN(#EmpA, ...)
 EMP_N_ADMIN(#EmpNA, ...)
 INVESTIGADOR(#EmpI, ...)
 OBRA(#Obra, OO)
 PROYECTO(#Proj, #EmpI, OP)
 CURSO(#Curso, #EmpI, OC)
 TRAB_OBRA(#EmpNA, #Obra)
 TRAB_PROJ(#EmpNA, #Proj)
 TRAB_CUR(#EmpNA, #Curso)
 DESENV_OBRA(#Obra, #Proj)
 DESENV_CUR(#Curso, #Proj)

12.3.15. Exemplo - Competições desportivas

É necessário projectar uma base de dados para as Competições Desportivas Inter Universitárias (CDIU). A base de dados será usada pelos membros da equipa no escritório dos comissários da CDIU. O escritório dos comissários controla todas as actividades da liga nomeadamente o agendamento de todos os desportos da liga, dos *hiring* oficiais para todas as competições da liga, validam a elegibilidade de todos os atletas da liga, mais a lista de todos os atletas, administradores, e treinadores para todas as universidades da liga.

O comissário identificou os seguintes aspectos como os mais importantes:

a. Para cada universidade:

Nome oficial da Universidade
Número de estudantes
Todos os desportos em que a Universidade participa
Logotipo
Nome do campo e a sua capacidade
Nome do campo de futebol e a sua capacidade
Nome, morada, telefone de casa e do trabalho das seguintes pessoas:

Reitor
Director desportivo
Director de informações desportivas
Representante dos atletas da universidade
Treinador principal de cada desporto

b. Lista de todos os Fiscais aprovados pela liga

Nome e número da segurança social
Morada de casa
Número de telefone de casa
Desportos que fiscalizam
Nível do treinador no ano passado
Competições específicas para as quais o Fiscal tenha sido designado para a próxima época.

c. Uma lista de todos os estudantes-atletas

Nome e número da segurança social
Morada de casa
Morada da Universidade
Número de telefone na Universidade
Current major
Grade point average
Hours toward graduation
Number of seasons of competition in each sport participated in Date of the first entry in college
Number of credit hours currently being taken
Number of credit hours completed in the last two terms

d. A agenda da liga para o próximo ano:

Equipa da casa para cada competição
Equipa de fora para cada competição
Data e hora de cada competição
Fiscais atribuídos
Desportos em que se irão realizar competições

e. Cada desporto na liga tem, uma comissão que define as regras, e um treinador principal que é designado pela liga como sendo o presidente dessa comissão.

Podem-se fazer as seguintes pressuposições: O agendamento de actividades diz respeito ao ano seguinte, o treinador principal treina apenas um desporto, algumas universidades não participam nas actividades da liga, algumas pessoas partilham o mesmo número de telefone, a liga tem desportos femininos e masculinos. Considere as seguintes entidades:



Usando os DER determine as relações necessárias para implementar a base de dados definida no enunciado do problema.

Solução:

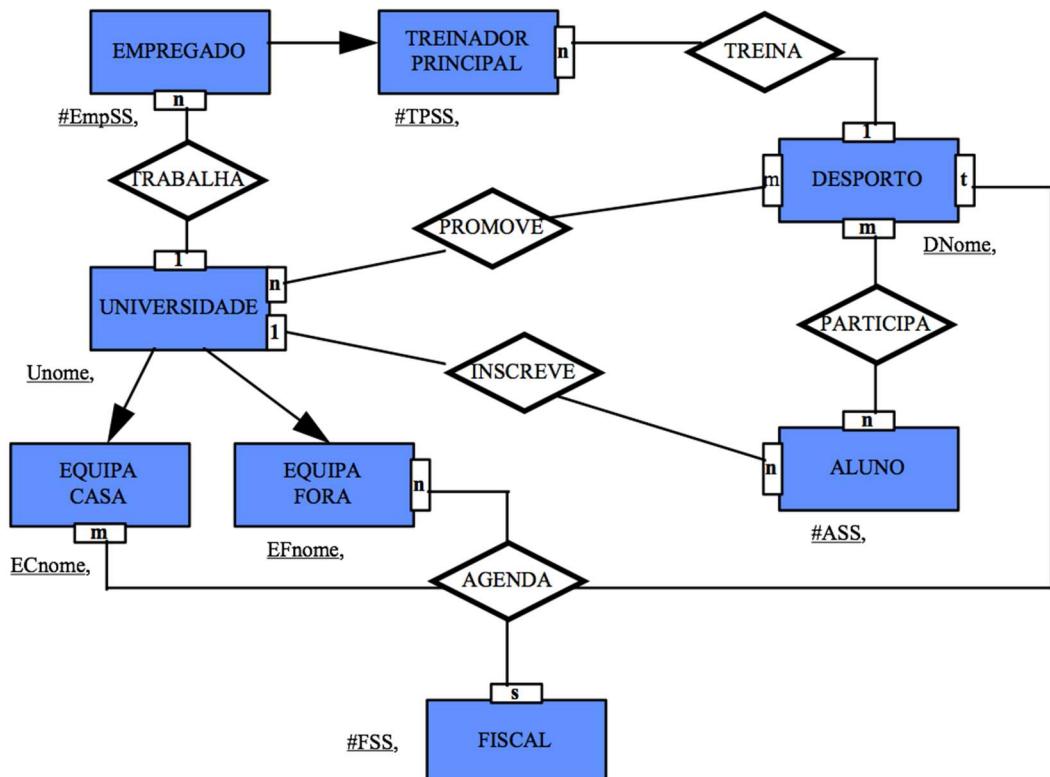
Assume-se que:

- a agenda refere-se ao ano que começa.
- cada treinador principal treina apenas um desporto.
- algumas universidades não participam e não participam em todas as modalidades desportivas
- algumas pessoas partilham o mesmo telefone do escritório.
- a liga tem desportos femininos e masculinos.

Atributos:

- #EmpSS: número da segurança social do empregado da universidade
- #TPSS: número da segurança social do treinador principal
- Dnome: nome da modalidade desportiva
- #ASS: número da segurança social do aluno
- Unome: nome da universidade
- ECnome: nome da equipa da casa
- EFnome: nome da equipa de fora
- #FSS: número da segurança social do fiscal

Diagrama de entidade - relacionamento



Relações preliminares:

UNIVERSIDADE(Unome,
)
 EMPREGADO(#EmpSS, , Unome)
 EQUIPACASA(ECnome,)
 EQUIPAFORA(EFnome,)
 FISCAL(#FSS,)
 TREINADORPRINCIPAL(#TPSS, Dnome)
 DESPORTO(Dnome)
 ALUNO(#ASS, , Unome)
 PROMOVE(Unome, Dnome,)
 PARTICIPA(Dnome, #ASS,)
 AGENDA(ECnome, EFnome, #FSS, Dnome,)

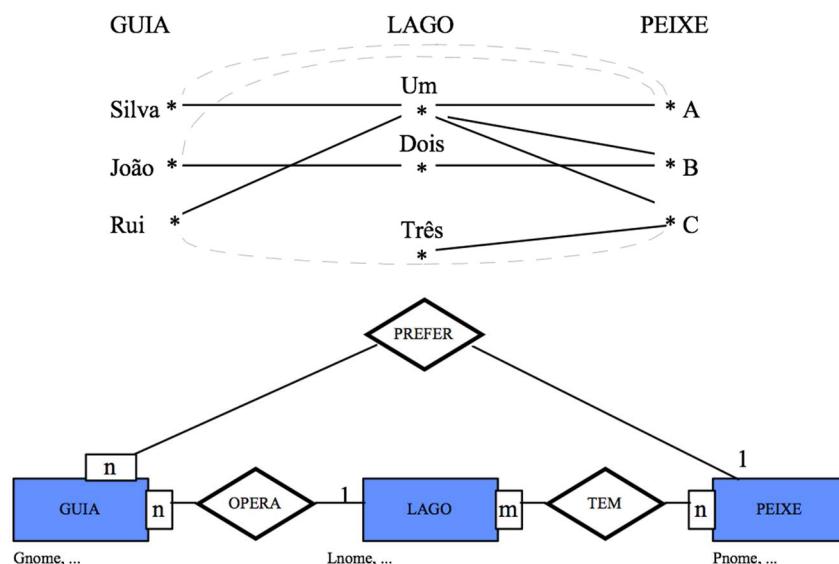
12.3.16. Exemplo - Pesca nos Lagos com três associações binárias

Assume-se:

- Silva opera apenas no lago Um
- João opera apenas no lago Dois
- Rui opera apenas no lago Um
- Silva prefere pescar o peixe A
- João prefere pescar o peixe A
- Rui prefere pescar o peixe C
- lago Um tem peixe A,B,C
- lago Dois tem peixe B
- lago Três tem peixe C

Atributos:

- | | |
|--------|-----------------------------------------------------|
| Gnome: | nome do guia |
| Tel: | número de telefone do guia |
| Folga: | dia de folga do guia |
| Lnome: | nome do lago onde o guia opera |
| Max: | número máximo de pessoas no grupo de um guia |
| Rec: | recorde de peixe pescado num lago |
| Pnome: | tipo de peixe existente num lago |
| Peso: | peso do maior peixe de cada tipo apanhado na região |
| Eng: | o melhor engodo para cada tipo de peixe na região |



Relações Preliminares:

- GUIA(Gnome, ... ,Lnome, Pnome)
- PEIXE(Pnome, ...)
- LAGO(Lnome, ...)
- LAGO _ TEM _ PEIXE(Lnome, Pnome, ...)

Relações:

- GUIA(Gnome, Tel, Folga, Max ,Lnome, Pnome)
- PEIXE(Pnome, Peso, Eng)
- LAGO(Lnome, Rec)
- LAGO _ TEM _ PEIXE(Lnome, Pnome)

12.3.17. Exemplo - Pesca nos Lagos com uma associação ternária

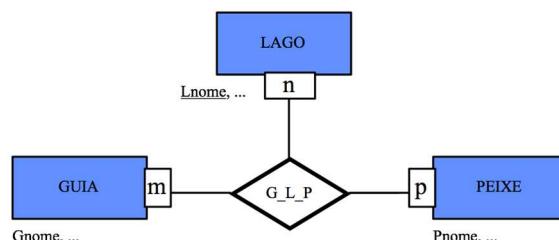
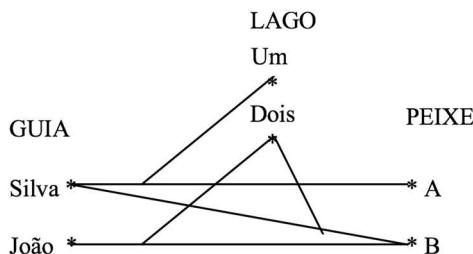
Relações necessárias para relacionamentos de ordem superior (entre três ou mais entidades)

Assume-se:

- Silva opera no lago Um e Dois
- João opera apenas no lago Dois
- Silva prefer pescar o peixe A no lago Um
- Silva prefer pescar o peixe B no lago Dois
- João prefer pescar o peixe A no lago Dois
- lago Um tem peixe A,B
- lago Dois tem peixe B

Atributos:

Gnome:	nome do guia
Tel:	número de telefone do guia
Folga:	dia de folga do guia
Lnome:	nome do lago
Max:	número máximo de pessoas no grupo do guia
Rec:	recorde de peixe pescado num lago
Pnome:	tipo de peixe existente num lago
Peso:	peso do maior peixe de cada tipo apanhado na região
Eng:	o melhor engodo para cada tipo de peixe na região



Regra 7: Quando o relacionamento é ternário, quatro relações preliminares são necessárias: uma para cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento. A relação gerada pelo relacionamento terá entre os seus atributos a chave de cada entidade.
Da mesma forma num relacionamento n-ário são necessárias n+1 relações.

Relações Preliminares:

GUIA(Gnome, ...))
PEIXE(Pnome, ...))
LAGO(Lnome, ...))
G_L_P(Gnome, Lnome, Pnome, ...)

Relações:

GUIA(Gnome, Tel, Folga, Max)
PEIXE(Pnome, Peso, Eng)
LAGO(Lnome, Rec)
G_L_P(Gnome, Lnome, Pnome)

Houve perda de informação, a partir das relações obtidas não é possível determinar que :

- o Silva prefere o peixe A no lago Um
- o Silva prefere o peixe B no lago Dois

12.4. Diagramas de Entidade-Relacionamento (Não Peter Chen)

Os diagramas de entidade Relacionamento, como o nome indica são compostos pela representação gráfica de **entidades** e dos **relacionamentos** entre si.

Ao longo dos anos, diversos autores têm alterado essa simbologia, mas as entidades representam por exemplo: Universidades, Estudantes, Departamentos, Clientes, Produtos, Ordens, etc.

Cada entidade possui um conjunto de atributos, por exemplo: o nome da entidade, o preço da entidade, etc.

Depois de normalizada uma base de dados, a cada entidade do DER costuma corresponder uma tabela com diversas colunas, campo, propriedades (**atributos**), e diversas linhas, registos (**objectos** desse tipo de entidade).

Por exemplo, se quisermos guardar numa base de dados os produtos existentes na loja, e os produtos encomendados pelos clientes, podemos começar o projecto dessa base de dados, criando um DER com três entidades “Cliente”, “Encomendas” e “Produtos”. A entidade Cliente, tem as suas propriedades (atributos): Código do Cliente, nome, morada , etc. A entidade “Produtos” tem os atributos: código do produto, nome, preço, etc. A entidade “Encomendas” tem os atributos: Ref do Cliente, Quantidade, data da encomenda, morada de destino, etc.

Um desses atributos, ou um conjunto de atributos, pode ter a designação de “**chave primária**” dessa tabela, se for suficiente conhecer esse(s) atributo(s) para identificar inequivocamente uma linha (um objecto) dessa tabela.

No caso da entidade “Cliente” o atributo “Código do Cliente” é único para cada cliente, se o soubermos identificamos inequivocamente todos os outros atributos desse cliente (objecto/linha).

O atributo que for a chave primária de uma tabela:

- tem de ter um valor que nunca se repita para outras linhas da tabela.
- nunca pode ser nulo numa das linhas da tabela.

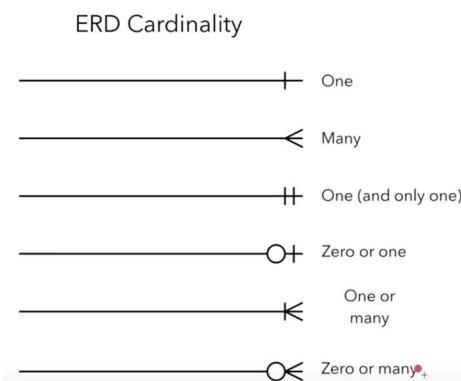
Numa tabela podem existir atributos que são a chave primária de outra tabela, nesse caso diz-se que esse(s) atributos são a “**chave estrangeira**”.

A cada tabela também se pode chamar **relação**, não confundir com **relacionamentos** entre entidades.

Os relacionamentos entre entidades podem ter um grau: unário, binário, ternário, etc.

Ao grau de um relacionamento pode-se chamar de **cardinalidade de um relacionamento**:

1 para 1, muitos-muitos, 0, 1:m, 0:m



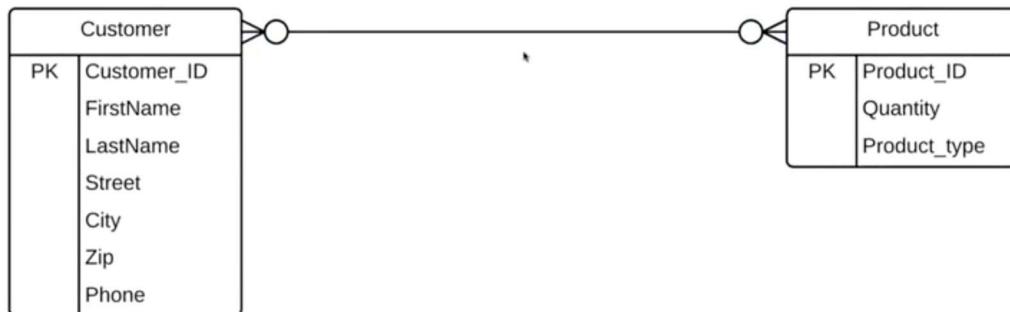
Relacionamento Recursivo: relacionamento da entidade consigo própria. (é representado por um losangulo).

12.4.1. Exemplo - Cliente, Ordens e Produtos

No caso do relacionamento entre a entidade “Cliente” e a entidade “Produto”, assumindo que:

- uma encomenda só pode ser efectuada por um, e apenas um, cliente.
- um cliente pode já estar registado na base de dados mas ainda não ter feito nenhuma encomenda. Ou pelo contrário, já ter feito inúmeras encomendas “m”.
- os produto já podem ter sido encomendados ou permanecerem na loja
- na encomenda podem constar vários tipos de produtos, no mínimo um tipo (c/ Quantity).

A forma de representar essas entidades e relacionamentos é ilustrado na figura seguinte:



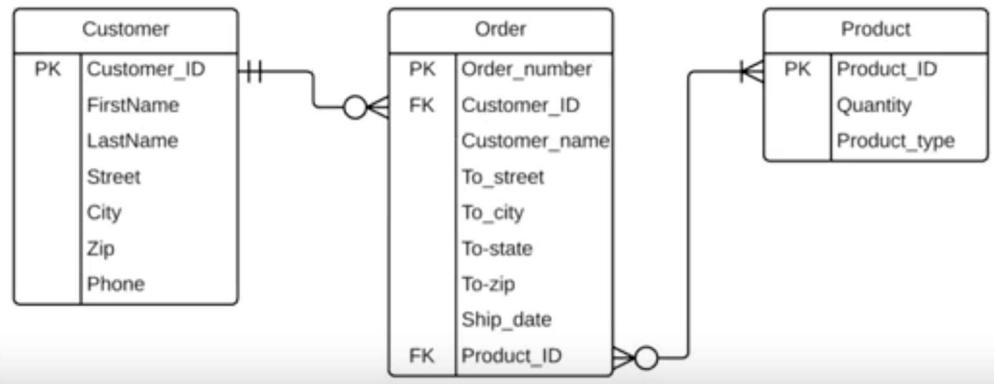
Depois de normalizada a base de dados, teríamos 3 tabelas

Regra 6: Quando o grau de um relacionamento binário é de N:M independentemente da participação obrigatória ou não das entidades, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento, tendo esta relação entre os seus atributos as chaves de cada uma das entidades.

Relações preliminares:

CUSTOMER(Customer_ID, ...)
PRODUCT(Product_ID, ...)
ORDER(Order_number, Customer_ID, Product_ID,...)

Outra forma seria:



Ao relacionamento da esquerda podemos aplicar a Regra nº5.

Regra 5: Quando o grau de um relacionamento binário é de 1:N com participação não obrigatória da entidade do lado N, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento, tendo esta relação entre os seus atributos as chaves de cada uma das entidades.

Relações preliminares:

CUSTOMER(Customer_ID, ...)

ORDER(Order_number, ..., Customer_ID, Product_ID)

CUST_ORD (Customer_ID, Order_number, ...)

Ao relacionamento da direita podemos aplicar a regra nº 6.

Regra 6: Quando o grau de um relacionamento binário é de N:M independentemente da participação obrigatória ou não das entidades, três relações são necessárias: uma por cada entidade, com a chave de cada entidade a servir como chave primária da relação correspondente e uma para o relacionamento, tendo esta relação entre os seus atributos as chaves de cada uma das entidades.

Relações preliminares:

ORDER(Order_number, ..., Customer_ID, Product_ID)

PRODUCT (Product_ID, ...)

PROD_ORD(Order_number, Product_ID, ...)

Em síntese, teríamos 5 tabelas:

PRODUCT (Product_ID, ...)

CUSTOMER(Customer_ID, ...)

ORDER(Order_number, ..., Customer_ID, Product_ID)

CUST_ORD (Customer_ID, Order_number, ...)

PROD_ORD(Order_number, Product_ID, ...)

Reparam que a tabela “ORDER” reune/contém os atributos das duas tabelas “CUST_ORD” e “PROD_ORD”. Podemos optar uma das duas hipóteses: apenas a tabela “ORDER”, ou as duas tabelas “CUST_ORD” e “PROD_ORD”.

Se optarmos pela primeira hipóteses, apenas a tabela “ORDER”, se uma encomenda incluir vários tipos de produtos, aparecerão na tabela tantas linhas quantos os tipos de produtos encomendados, duplicando os atributos “Order_ID” e “Customer_ID” nessas linhas, relativas à mesma encomenda.

Se usarmos o LucidChart para desenhar o DER e gerar as tabelas normalizadas, obtemos:

```
CREATE TABLE `Order` (
    `Order_number` int,
    `Customer_ID` varchar(50),
    `Product_ID` int,
    `Ship_date` varchar(50),
    PRIMARY KEY (`Order_number`),
    KEY `FK` (`Customer_ID`, `Product_ID`)
);
```

```
CREATE TABLE `Customer` (
    `Customer_ID` int,
    `First_name` varchar(50),
    `Last_name` varchar(50),
    PRIMARY KEY (`Customer_ID`)
);
```

```
CREATE TABLE `Product` (
    `Product_ID` int,
    `Quantity` int,
    `Product_type` varchar(40),
    PRIMARY KEY (`Product_ID`)
);
```

Licença de estudante para usar o LucidChart,
Free Education (Pro) Accounts for Teachers and Students
Para criar uma conta/user de estudante:
<https://www.lucidchart.com/pages/usecase/education>

Para usar remotamente a aplicação Lucidchart e criar os gráficos:
<https://www.lucidchart.com/users/login>

Bibliografia:

Entity Relationship Diagram (ERD) Tutorial - Part 1
<https://www.youtube.com/watch?v=QpdhBUYk7Kk> (7 min - Lucid Chart)
<https://www.youtube.com/watch?v=-CuY5ADwn24> (14 min - Lucid Chart)

12.5. Linguagem SQL

Nas linhas seguintes são apresentados cinco tipos de comandos SQL:

“**CREATE TABLE** customer(CustID INTEGER PRIMARY KEY, LastName CHARACTER VARYING (25), FirstName CHARACTER VARYING (20), Address addr_typ, Phone CHARACTER VARYING (15) ARRAY [3]) “

“**SELECT** OrderID, CustomerID **FROM** Orders”

“**SELECT** * **FROM** Table1 **ORDER BY** field1 ASC or **SELECT** * **FROM** Table1 **ORDER BY** field1 DESC”

“**INSERT INTO** table_name(column_1, column_2, ..., column_n) **VALUES** (value_1, value_2, ..., value_n)”

“**INSERT INTO** Customers(CustomerID, CompanyName) **Values**('NWIND', 'Northwind Traders') **WHERE** id=1”

“**INSERT INTO** Motor(VelMotorOrdem, VelMotorLeitura) **VALUES** (?, ?)”

“**UPDATE** table_name **SET** column_1 = expression_1, column_2 = expression_2,..., column_n = expression_n [**WHERE** predicates]”

“**UPDATE** members **SET** first_name='Jose' **WHERE** id='1'”

“**UPDATE** COMP **SET** Pay = Pay + 100”

“**DELETE FROM** Customer **WHERE** FirstName = ‘David’ AND LastName = ‘Taylor’”

“**DELETE FROM** members **WHERE** first_name='Jose' ”

Poderá aprofundar os conhecimentos em SQL no sitio: <https://www.w3schools.com/sql/>

12.6. VBasic – ODBC

12.6.1. Configuração de uma ligação ODBC para aceder às bases de dados

Neste exemplo vamos aceder à base de dados Access através de uma ligação *ODBC (Open DataBase Connectivity)*. A Figura 1216 apresenta uma base de dados com o nome “Produtos.mdb” possui uma só tabela chamada “Produtos” onde estão definidos vários campos (colunas): “ID, NomeDoProduto, PreçoUnitário” e três registos (linhas).



Figura: Base de dados Access

No entanto, antes de podermos utilizar uma ligação ODBC para aceder às bases de dados presentes no sistema operativo Windows é necessário defini-la. Para isso é necessário utilizar o gestor ODBC “*Iniciar > Ferramentas Administrativas > Origem de Dados (ODBC)*”. Utilizando o gestor ODBC podemos definir o nome da ligação ODBC e qual a base de dados que deverá ser acedida através dessa ligação. É através do nome da ligação ODBC que as aplicações Windows poderão aceder à base de dados respectiva.



Figura: Adicionar uma ligação ODBC no ambiente Windows

Podemos definir um novo *DSN (Data Source Name)* no sistema se utilizarmos a aplicação “Administração da origem de dados de ODBC” e selecionarmos o botão “Adicionar”. Neste caso vamos criar uma ligação ODBC como nome (*DSN=“ODBC”*).



Figura: Configurar a ligação ODBC para aceder ao ficheiro “Produtos.mdb”

Através do botão “Selecionar” podemos associar esta ligação ODBC a uma base de dados específica. Neste caso associámos a ligação “ODBC” à base de dados Access, mais exatamente ao ficheiro “Produtos.mdb”

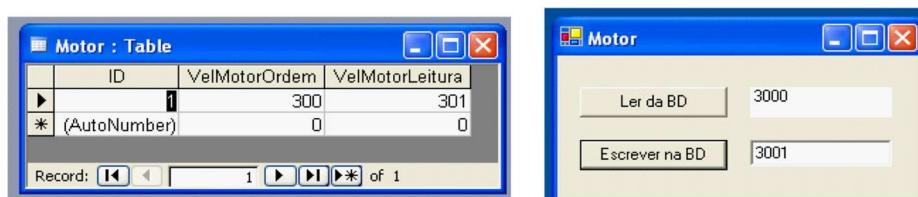
12.6.2. Acesso à base de dados através de uma ligação ODBC

Este exemplo demonstra como é possível desenvolver um programa em VisualBasic capaz de escrever e ler num ficheiro de dados “VelMotor.mdb”. Este ficheiro foi criado em Access e contém uma só tabela chamada “Motor”. A tabela “Motor” só possui um registo (linha) e três campos (colunas): “ID”, “VelMotorOrdem” e “VelMotorLeitura”. O primeiro campo identifica o registo, o segundo campo contém o nº de rpm a que o motor deveria rodar, e o terceiro campo contém o nº de rpm reais a que o motor efectivamente roda. Pode existir uma diferença entre as rpm pedidas e as reais devido à inércia da carga que o motor tem de accionar e aos atritos a vencer.

Quando o utilizador “clica” no botão “Le da BD”, esta aplicação lê o campo “VelMotorOrdem” existente na tabela “Motor” e visualiza esse valor no label “lblVelMotorOrdem”.

Quando o utilizador “clica” no botão “Escreve na BD” esta aplicação lê a caixa de texto “txtVelMotorLeitura” e escreve o seu valor no ficheiro “VelMotor.mdb”, mais exactamente no seu campo “VelMotorLeitura”.

Deverá ser criada previamente uma ligação ODBC (Open DataBase Connectivity) do tipo “File DSN” com o nome “ODBCMotor”, com o driver para Microsoft Access e tendo como database o ficheiro “VelMotor.mdb”.



```

Imports System.Data.Odbc

Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Dim cn As New OdbcConnection("FileDSN=ODBCMotor")

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
        cn.Open()
    End Sub

    Private Sub btnLer_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLer.Click
        ' Lê o primeiro registo do campo VelMotorOrdem da tabela Motor
        Dim cmd As New OdbcCommand("select VelMotorOrdem from Motor where ID=1", cn)
        Dim rdr As OdbcDataReader = cmd.ExecuteReader
        If rdr.Read Then lblVelMotorOrdem.Text = rdr("VelMotorOrdem")
        rdr.Close()
    End Sub

    Private Sub btnEscrever_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEsc.Click
        ' Escreve o valor da caixa de texto no primeiro registo do campo VelMotorLeitura da tabela Motor
        Dim cmd As OdbcCommand = cn.CreateCommand()
        cmd.CommandText = ("update Motor Set VelMotorLeitura=" & txtVelMotorLeitura.Text & " where ID=1")
        cmd.ExecuteNonQuery()
    End Sub

    Private Sub Form1_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
        cn.Close()
    End Sub

End Class

```

Ou em alternativa ...

```

Imports System.Data.Odbc

Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim Cn As New OdbcConnection
    Dim Cmd As New OdbcCommand
    Dim Rdr As OdbcDataReader

    Windows Form Designer generated code
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Cn.ConnectionString = "FileDSN=ODBC_EIB_NET"
        Cn.Open()
        Cmd.Connection = Cn
    End Sub

    Private Sub BtLer_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtLer.Click
        Cmd.CommandText = "Select * from Motor Where ID=1"
        Rdr = Cmd.ExecuteReader
        If Rdr.Read Then TxtVelMotorLeitura.Text = Rdr("VelMotorOrdem")
        Rdr.Close()
    End Sub

    Private Sub BtEscrever_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtEscrever.Click
        Cmd.CommandText = "Update Motor Set VelMotorLeitura=" & TxtVelMotorOrdem.Text & " Where ID=1"
        Cmd.ExecuteNonQuery()
    End Sub
End Class

```

Figura: VBasic – Acesso a uma base de dados (ODBC)

Vamos agora analisar o código em VBasic deste exemplo.

Ligaçāo

O objeto *cn* é do tipo *OdbcConnection* que estabelece a ligação ODBC na aplicação VBasic, o parāmetro *ConnectionString* define qual das ligações ODBC previamente estabelecidas será utilizada.

```
Dim cn As New OdbcConnection("FileDSN=ODBCMotor")
```

A ligação é aberta com o método *cn.Open* e finalizada com o método *cn.Close*.

Leitura

Para a leitura de um valor da base de dados é necessário executar um comando com o correspondente código SQL (Structured Query Language). A linha de código seguinte cria o objeto “*cmd*” do tipo *OdbcCommand*, com o código SQL e com o nome do objeto de ligação ODBC “*cn*” anteriormente definido:

```
Dim cmd As New OdbcCommand("select VelMotorOrdem from Motor where ID=1", cn)
```

Ao executar este comando é devolvida a informação da base de dados para o objeto “*rdr*” que é do tipo *OdbcDataReader*:

```
Dim rdr As OdbcDataReader = cmd.ExecuteReader
```

O objeto *rdr* tem todos os registos resultantes do comando SQL (neste caso é apenas um), no entanto estes registos (linhas na tabela) só podem ser acedidos um de cada vez.

O método *read* avança para o próximo registo dos resultados, e além disso retorna um valor booleano, Verdadeiro no caso de haver mais registos e Falso caso contrário.

A linha de código seguinte, avança uma linha no registo, ou seja passa para a primeira linha, e atribui o valor do campo “*VelMotorOrdem*” ao label “*lblVelMotorOrdem*”.

```
If rdr.Read Then lblVelMotorOrdem.Text = rdr("VelMotorOrdem")
```

É ainda necessário fechar o “*datareader*” com o método *Close*.

Escrita

Para a escrita de um valor na base de dados, também é necessário executar um comando com código SQL, mas neste caso não são devolvidos dados, este tipo de comando é designado por *Non Query Command*.

As linhas de código seguintes permitem criar e executar um comando ODBC, de forma a passar o valor de caixa de texto “*txtVelMotorLeitura*”, para o primeiro registo do campo *VelMotorLeitura*

```
Dim cmd As OdbcCommand = cn.CreateCommand()
cmd.CommandText = ("update Motor Set VelMotorLeitura=''" & txtVelMotorLeitura.Text & "' where ID=1")
cmd.ExecuteNonQuery()
```

Neste exemplo são utilizados os objetos do tipo *DataReader* e *Command*. Vamos analisá-los agora com maior detalhe.

Command

Os objetos do tipo *Command* podem enviar perguntas SQL/comandos para os “drivers” das bases de dados através de ligações previamente estabelecida. Para criar um objeto do tipo *Command*, tem de definir a ligação aos “drives” das bases de dados e de definir o comando SQL. Para isso escreva as três linhas seguintes:

```
Dim cmd as new OdbcCommand  
cmd.ConnectionString = "DSN=Motor"  
cmd.CommandText= "SELECT * FROM Motor"
```

Os objetos do tipo *Command* possuem diversos métodos e propriedades, nomeadamente:

cmd.ExecuteNonQuery: este método permite enviar um comando escrito em SQL para o “driver” que gere a base de dados. Retorna o número de registos que foram afectados pelo comando. Neste exemplo retorna o valor 1 pois só o primeiro registo foi afectado pelo comando.

```
cmd.CommandText = "UPDATE Motor SET VelMotorOrdem = 1000 WHERE ID=1"  
TextBox1.text = cmd.executeNonQuery()
```

cmd.ExecuteScalar: este método permite também enviar um comando escrito em SQL para o “driver” que gere a base de dados e retorna o dado pedido pelo comando, um só valor (escalar) e não um array ou outro tipo de dados. Neste exemplo apenas é retornado um valor, por exemplo: 1000

```
cmd.CommandText = "SELECT VelMotorOrdem FROM Motor WHERE ID=1"  
textBox1.text = cmd.executeScalar()
```

cmd.ExecuteReader: este método envia também um comando escrito em SQL para o “driver” que gere a base de dados e devolve o resultado da consulta sob a forma de um *DataReader*. Neste caso o resultado da consulta pode consistir num conjunto de registos com vários campos.

```
Dim cmd As OdbcCommand  
Dim rdr As OdbcDataReader = cmd.ExecuteReader  
cmd.CommandText = "SELECT VelMotorOrdem FROM Motor"  
Rdr = cmd.ExecuteReader  
If rdr.Read Then lblVelMotorOrdem.Text = rdr("VelMotorOrdem")  
rdr.Close()
```

DataReader

Os objetos do tipo *DataReader* permitem guardar vários registos, cada registo com vários campos.

```
Dim reader As OdbcDataReader  
Reader = cmd.ExecuteReader()  
While reader.Read()  
    txtResults.Text = txtResults.Text & Reader("VelMotorOrdem") & Reader("VelMotorLeitura")  
End While  
reader.Close()
```

Os campos de cada registo podem ser acedidos através dos métodos seguintes:

X = Reader("VelMotorOrdem")	‘ obtém o calor do campo “VelMotorOrdem”
X = Reader.GetInt16(0)	‘ obtém o valor inteiro do campo/coluna indicado
Str = Reader.GetString(2)	‘ obtém o conteúdo da terceira coluna

Cada objeto do tipo *DataReader* pode conter vários registos/linhas, mas os métodos anteriores aplicam-se apenas ao registo activo. Para mudar de registo activo usa-se a instrução *Reader.Read()*, esta instrução copia para o registo activo o registo seguinte. Quando esta função retornar um valor “falso” significa que já não existem mais registos para serem lidos

12.7. VBasic – MySQL

Para aceder ao gestor de bases de dados “MySQL server”, a partir do VBasic, tem de instalar o “MySQL Connector/NET” no seu computador.

12.7.1. Instalação do MySql

Para instalar o “MySQL Server”, o “MySQL Workbench”, e o “MySQL Connector/NET”, aceda a:

<https://dev.mysql.com/downloads/windows/installer/5.7.html>

- MySQL

- **MySQL Server** : Este programa não tem interface, é responsável por gerir as bases de dados, aceita ligações TCP/IP e quando recebe mensagens SQL executa-as. De acordo com as mensagens SQL recebidas, cria bases de dados e tabelas, insere registos nas tabelas, altera ou apaga registos, etc.

- **MySQL Workbench** : Este programa é um cliente TCPIP. Através da interface deste programa o utilizador fazer com sejam enviadas mensagens SQL para o “MySQL Server”.

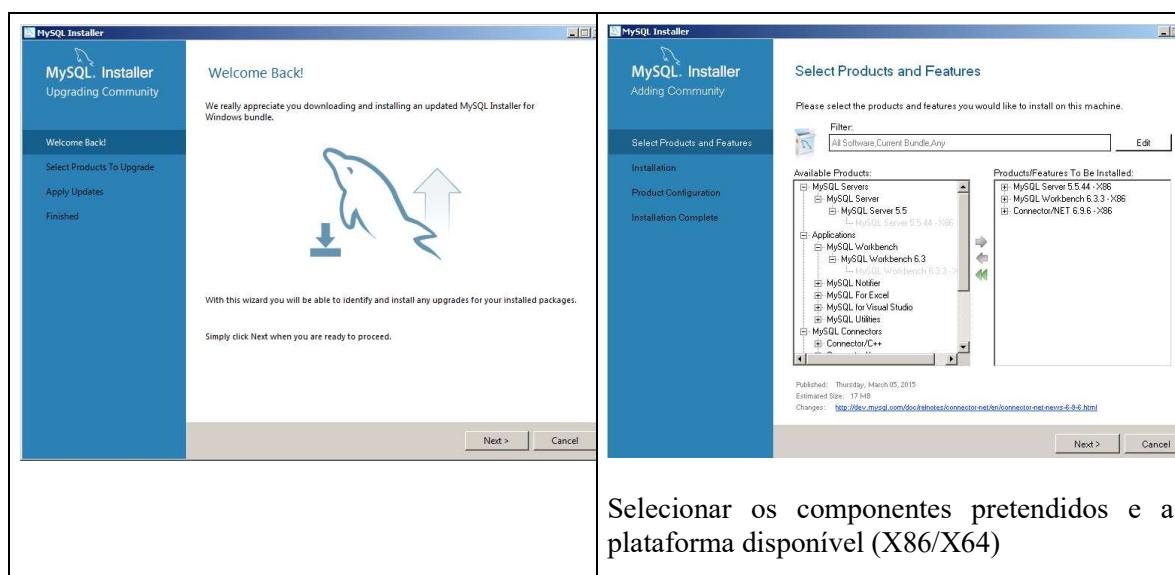
Tutorial: <http://dev.mysql.com/doc/workbench/en/>

- **MySQL Connector/NET** : Esta biblioteca (MySQL.Data.Dll) permite ao VBasic atuar como o “MySQL Workbench” e também acede ao “MySQL server” enviando-lhe mensagens SQL.

Tutorial: <http://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-intro.html>

Depois de o instalar o “MySQL connector” no seu computador, fica disponível um ficheiro .DLL que pode incluir nos seus projetos em VBasic. Esta DLL poderá ficar instalada no diretório: *C:\Programas\MySQL\MySQL Connector Net 6.1.6\Assemblies\MySQL.Data.Dll*

Deve criar um utilizador, caso ainda não o tenha feito.



Installation

Press Execute to upgrade the following products.

Product	Status	Progress	Notes
MySQL Server 5.7.16	Downloaded		
MySQL Workbench 8.3.8	Downloaded		
MySQL Notifier 1.1.7	Complete		
MySQL For Excel 13.6	Complete		
MySQL for Visual Studio 12.6	Installing	95%	
MySQL Utilities 1.6.4	Complete		
Connector/ODBC 5.3.6	Complete		
Connector/C++ 11.7	Downloaded		
Connector/J 5.1.40	Complete		
MySQL Documentation 5.7.16	Downloaded		
Samples and Examples 5.7.16	Complete		

Show Details >

< Back Execute Cancel

Configuração do MySQL

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Machine

Connectivity

Use the following controls to select how you would like to connect to this server:

TCP/IP Port Number: 3306

Open Firewall port for network access

Named Pipe Pipe Name: MYSQL

Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.

Show Advanced Options

Next > Cancel

Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: Repeat Password: Password Strength: Weak

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL Username	Host	User Role
aborges	%	DB Admin

Add User Edit User Delete

< Back Next > Cancel

MySQL Installer

Samples and Examples

Connect To Server

Here are the compatible servers installed. If more than one, please select one.

Server	Architecture	Status
MySQL Server 5.7.16	x86	Running

Now give us the credentials we should use (needs to have root privileges). Click check to make sure they work.

User: Credentials provided in Server configuration
 Password: Connection successful.

Check Connection successful.

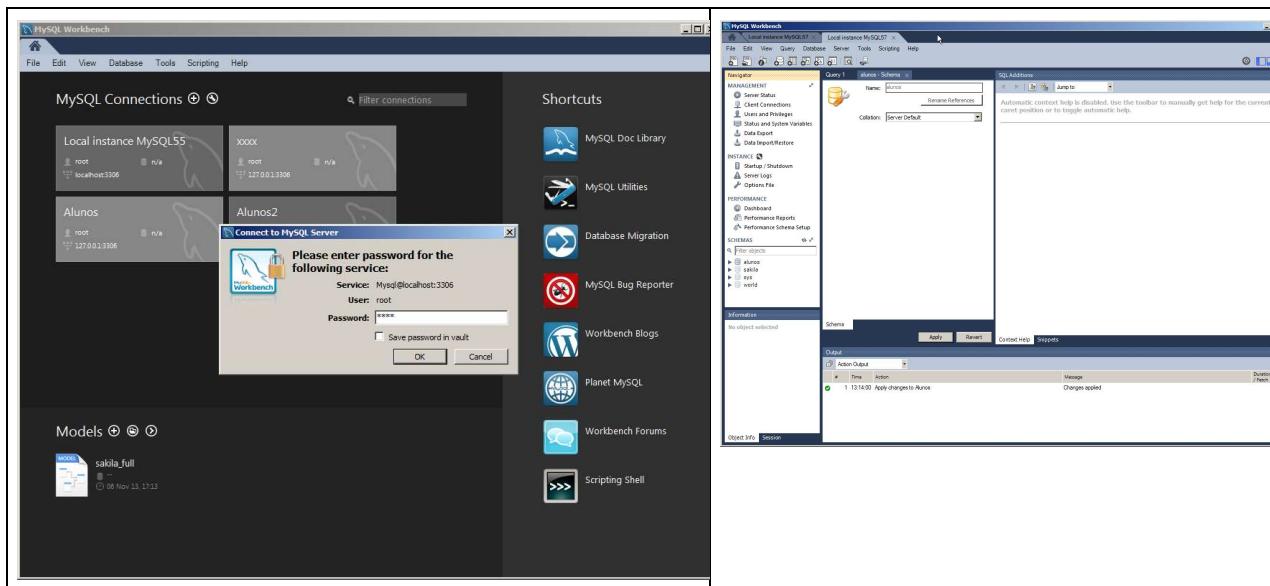
Next > Cancel

Criar um utilizador e atribuir a respetiva password.

Testar a ligação ao servidor.
Atenção à Password.

12.7.2. Exemplo I: Base de dados "Alunos" (Visual Basic/MySQL)

Após a conclusão da instalação, podemos utilizar o MySQL Workbench, para a criação de bases de dados e tabelas. Crie a base de dados (schema): **alunos**



Crie uma tabela associada à base de dados, tabela:

dados_al(idDados_Al, Nome, NumMecanografico, Telefone).

A screenshot of the MySQL Workbench interface showing the creation of a new table. The 'Navigator' panel on the left shows the 'alunos' schema selected. In the center, the 'Query 1' tab is active, displaying the creation of the 'dados_al' table. The table definition is as follows:

```
Table Name: dados_al Schema: alunos
Collation: utf8 - default collation Engine: InnoDB
Comments:
```

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idDados_Al	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL				
Nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
NumMecanografico	VARCHAR(6)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Telefone	VARCHAR(12)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

The 'Columns' tab is selected at the bottom. At the bottom right, there are 'Apply' and 'Revert' buttons.

Aplicação em Visual Basic, para aceder à base de dados e à tabela criada anteriormente:



Imports MySql.Data.MySqlClient

Para executar este programa necessitamos instalar o MySQL

```
Public Class Form1
    Dim cn As New MySqlConnection
    Dim cmd As New MySqlCommand
    Dim data_reader As MySqlDataReader

    Private Sub Btn_Connect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        ' Ligação e Abertura da Base de Dados e Seleção do Esquema
        cn.ConnectionString = "Server=localhost; User Id=root; Password=0000; Database=alunos"

        Try
            If cn.State = ConnectionState.Closed Then
                cn.Open()
                MsgBox("Ligação Correcta à Base de Dados alunos...")
            End If
        Catch ex As Exception
            cn.Close()
            MsgBox("Ligação Incorrecta à Base de Dados alunos...")
        End Try

        cmd.Connection = cn
    End Sub

    Private Sub Btn_LerBaseDados_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        ' Leitura sequencial dos dados existentes na base de dados
        Dim str As String
        ' Seleção da tabela da base de dados para leitura dos dados
        cmd.CommandText = "SELECT * FROM dados_al"
        data_reader = cmd.ExecuteReader

        ' Leitura sequencial da tabela.. Guarda o conteúdo dos registos na variável str
        While data_reader.Read
            str = str & data_reader("idDados_Al") & vbTab & data_reader("Nome") & vbTab & data_reader("Telefone") & vbCrLf
        End While

        Txt_Leitura.Text = str      ' Visualiza os dados de todos os registo
        data_reader.Close()
    End Sub
```

```

Private Sub Btn_InserirRegisto_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ' Insere Registo numa Base de Dados
    cmd.CommandText = "Insert into dados_al(idDados_Al,Nome,Telefone) Values('' & Txt_id.Text & '''' &
Txt_nome.Text & '''' & Txt_telefone.Text & '')"
    cmd.ExecuteNonQuery()
End Sub

Private Sub Btn_UpdateRegisto_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ' Update de um registo existente
    cmd.CommandText = "UPDATE dados_al Set Nome='' & Txt_nome.Text & '', Telefone='' & Txt_telefone.Text &
'' where idDados_Al=''" & CInt(Txt_id.Text) & """
    cmd.ExecuteNonQuery()
End Sub

Private Sub Btn_LerRegisto_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ' Leitura de um Registo Específico
    Dim str As String

    ' Procura do Registo pretendido
    cmd.CommandText = "Select * From dados_al Where idDados_Al=''" & Txt_id.Text & """
    'cmd.CommandText = "Select * From alunos_telef Where nome='xx' "
    data_reader = cmd.ExecuteReader
    ' Leitura do Registo
    data_reader.Read()
    Txt_nome.Text = data_reader("Nome")
    Txt_telefone.Text = data_reader("Telefone")
    data_reader.Close()
End Sub

Private Sub Btn_ApagarRegisto_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ' Apagar Registo através da chave de indexação
    cmd.CommandText = "Delete From dados_al Where idDados_Al=''" & Txt_id.Text & """
    cmd.ExecuteNonQuery()
End Sub

Private Sub Btn_MaxId_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    ' Procura o registo com o max id
    cmd.CommandText = "Select max(idDados_Al) From dados_al"
    ' executa o comando para leitura do máximo id do registo
    Dim total As Integer = cmd.ExecuteScalar
    Txt_MaxId.Text = total.ToString()
End Sub

Private Sub Btn_Terminar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
End ' Fim da aplicação
End Sub

Private Sub Form1_FormClosed(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs)
    MsgBox("Fimar Aplicação...")
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    MsgBox("A Abrir aplicação...")
End Sub
End Class

```

No exemplo anterior foram usados objetos de três tipos:

- MySQLConnection
- MySqlCommand
- MySQLDatareader

Vamos analisá-los considerando três situações:

- estabelecer uma ligação entre o VBasic e o servidor MySQL - (Ligaçāo).
- escrever dados na Base de dados – (Escrever).
- ler dados na base de dados e guardar a resposta - (Leitura).

Ligaçāo

O objeto *cn* é do tipo *MySQLConnection*, estabelece uma ligação TCPIP entre a aplicação VBasic e o servidor MySQL.

```
Dim cn As New MySQLConnection  
...  
cn.ConnectionString = "Database=Travelclub; Data Source=localhost; User Id=root; Password=xxx"  
cn.Open()
```

A propriedade *cn.ConnectionString* é do tipo string, no exemplo contém:

- o nome da base de dados a que se pretende aceder (ex. Travelclub)
- o computador onde o “MySQL server” reside (ex. Localhost)
- O nome de utilizador e password que o servidor “MySQL server” necessita para aceitar a ligação TCPIP.

A ligação é estabelecida com o método *cn.Open* e finalizada com o método *cn.Close*.

Leitura

Para ler um valor da base de dados é necessário executar um comando com o texto SQL (Structured Query Language) adequado. A linha de código seguinte cria o objeto “*cmd*” do tipo *MySQLCommand*. Este objecto contém a mensagem SQL e o nome do objeto “*cn*” definido anteriormente:

```
Dim cmd As New MySqlCommand  
cmd.CommandText = "select VelMotorOrdem from Motor where ID=1"  
cmd.Connection = Cn
```

Ao executar este comando, a informação devolvida pela base de dados é guardada no objeto “*rdr*” que é do tipo *MySQLDataReader*:

```
Dim rdr As MySQLDataReader  
rdr = cmd.ExecuteReader
```

O objeto *rdr* tem todos os registos resultantes do comando SQL (neste caso é apenas um), no entanto estes registos (linhas da tabela) só podem ser acedidos um de cada vez.

O método *read* avança para o próximo registo dos resultados, e além disso retorna um valor booleano, Verdadeiro no caso de haver mais registos e Falso caso contrário.

A linha de código seguinte, avança uma linha no registo, ou seja passa para a primeira linha, e atribui o valor do campo “*VelMotorOrdem*” ao label “*lblVelMotorOrdem*”.

```
If rdr.Read Then lblVelMotorOrdem.Text = rdr("VelMotorOrdem")
```

É ainda necessário fechar o “*datareader*” com o método *Close*.

Escrita

Para a escrita de um valor na base de dados, também é necessário executar um comando com código SQL, mas neste caso não são devolvidos dados, este tipo de comando é designado por *Non Query Command*.

As linhas de código seguintes permitem criar e executar um comando MySQL, de forma a passar o valor de caixa de texto “*txtVelMotorLeitura*”, para o primeiro registo do campo *VelMotorLeitura*

```
Dim cmd As New MySqlCommand  
cmd.CommandText = "update Motor Set VelMotorLeitura="" & txtVelMotorLeitura.Text & "" where ID=1"  
cmd.Connection = Cn  
cmd.ExecuteNonQuery()
```

Neste exemplo são utilizados os objetos do tipo *DataReader* e *Command*. Vamos analisá-los agora com maior detalhe.

Command

Os objetos do tipo *Command* podem enviar perguntas SQL/comandos para o gestor da base de dados através de ligações previamente estabelecida. Para criar um objeto do tipo *Command*, tem de definir a ligação aos “drives” das bases de dados e de definir o comando SQL. Para isso escreva as três linhas seguintes:

```
Dim cmd as new MySQLCommand  
cmd.Connection = Cn  
cmd.commandText= "SELECT * FROM Motor"
```

Os objetos do tipo *Command* possuem diversos métodos e propriedades, nomeadamente:

cmd.ExecuteNonQuery: este método permite enviar um comando escrito em SQL para a base de dados. Retorna o número de registos que foram afectados pelo comando. Neste exemplo retorna o valor 1 pois só o primeiro registo foi afectado pelo comando.

```
cmd.CommandText = "UPDATE Motor SET VelMotorOrdem = 1000 WHERE ID=1"  
TextBox1.text = cmd.executeNonQuery()
```

cmd.ExecuteScalar: este método permite também enviar um comando escrito em SQL para a base de dados e retorna o dado pedido pelo comando, um só valor (escalar) e não um array ou outro tipo de dados. Neste exemplo apenas é retornado um valor, por exemplo: 1000

```
cmd.CommandText = "SELECT VelMotorOrdem FROM Motor WHERE ID=1"  
textBox1.text = cmd.executeScalar()
```

cmd.ExecuteReader: este método envia também um comando escrito em SQL para a base de dados e devolve o resultado da consulta sob a forma de um *DataReader*. Neste caso o resultado da consulta pode consistir num conjunto de registos com vários campos.

```
Dim cmd As MySQLCommand  
Dim rdr As MySQLDataReader  
cmd.CommandText = "SELECT VelMotorOrdem FROM Motor"  
Rdr = cmd.ExecuteReader  
If rdr.Read Then lblVelMotorOrdem.Text = rdr("VelMotorOrdem")  
rdr.Close()
```

DataReader

Os objetos do tipo *DataReader* permitem guardar vários registo, cada registo com vários campos.

```
Dim cmd as new MySQLCommand  
cmd.Connection = Cn  
cmd.CommandText= "SELECT * FROM Motor"  
  
Dim reader As MySQLEader  
  
Reader = cmd.ExecuteReader()  
While reader.Read()  
    txtResults.Text = txtResults.Text & Reader("VelMotorOrdem") & Reader("VelMotorLeitura")  
End While  
reader.Close()
```

Os campos de cada registo podem ser acedidos através dos métodos seguintes:

X = Reader("VelMotorOrdem")	‘ obtém o valor do campo “VelMotorOrdem”
X = Reader.GetInt16(0)	‘ obtém o valor inteiro do campo/coluna indicado
Str = Reader.GetString(2)	‘ obtém o conteúdo da terceira coluna

Cada objeto do tipo *DataReader* pode conter vários registo/linhas, mas os métodos anteriores aplicam-se apenas ao registo ativo. Para mudar de registo ativo usa-se a instrução *Reader.Read()*, esta instrução copia para o registo ativo o registo seguinte. Quando esta função retornar um valor “falso” significa que já não existem mais registo para serem lidos

12.7.3. Exemplo II: Da BaseDados “Reservatorio” ao PLC (MySQL + FaconSrv)

Neste exemplo, pretende-se ler a tabela “ControloReservatorio” e em função dos valores de Y0, Y1 e Y2 da tabela , ativar as saídas digitais correspondentes do PLC.

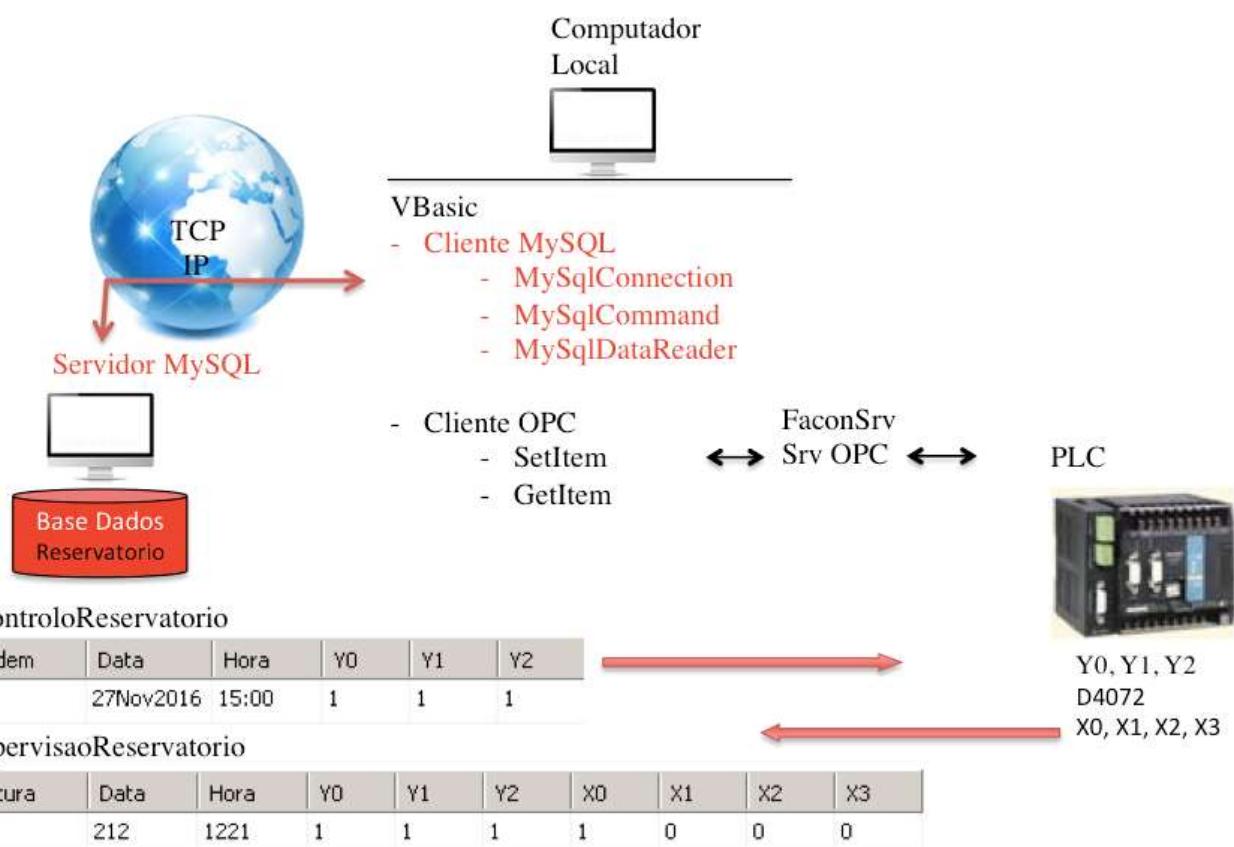
Pretende-se também, ler o estado das entradas e saídas digitais do PLC : X0,X1,X2,X3,Y0,Y1,Y2 e atualizar a tabela “SupervisaoReservatorio”.

As duas tabelas fazem parte da base de dados “**Reservatorio**”.

- **SupervisaoReservatorio** (Leitura,Y0,Y1,Y2,X0,X1,X2,X3)
- **ControloReservatorio** (Ordem,Y0,Y1,Y2)

Use o MySQL Workbench para criar a base de dados e as tabelas.

Edite as tabelas de acordo com os valores que a figura seguinte apresenta.



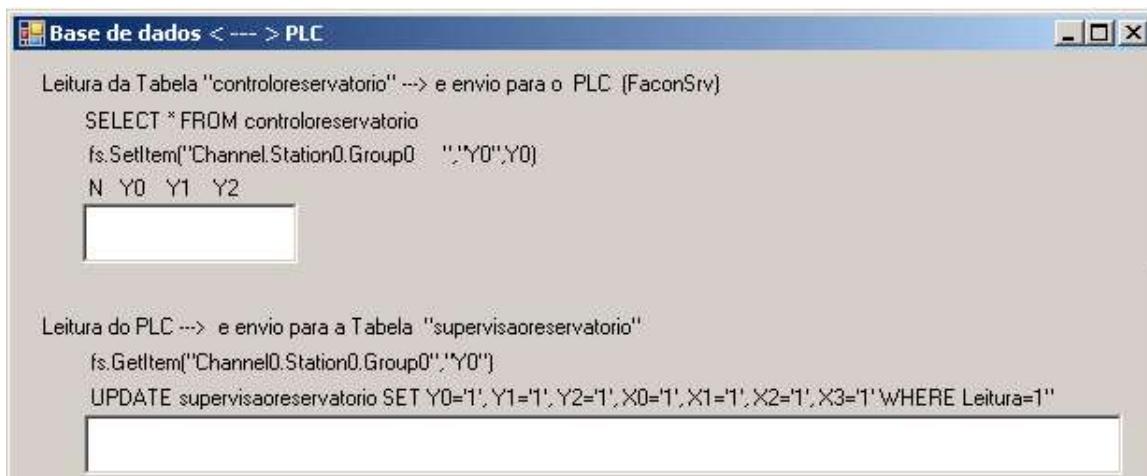
Crie em VBasic um projecto com o nome “**Trab12_Database_To_PLC**”. A aplicação deste exemplo usa objetos do tipo MySQL e do tipo FaconSrv para aceder à base de dados “Reservatorio” e ao PLC.

A aplicação VB, no procedimento Timer1_Tick, chama dois subprocedimentos:

- **Private Sub ControloReservatorio_To_PLC()**
Este subprocedimento lê a tabela “**ControloReservatorio**” e atualiza as saídas digitais do PLC: Y0,Y1 e Y2.

- **Private Sub PLC_To_SupervisaoReservatorio()**
Este subprocedimento lê o estado das saídas e entradas digitais do PLC e guarda na tabela “**SupervisaoReservatorio**” o seu estado.

Na interface “Form1”, existem apenas duas textbox. A textbox de cima “txt_BD_to_PLC” apresenta o conteúdo da tabela “ControloReservatorio”: Y0,Y1 e Y2. A textbox de baixo “txt_PLC_to_BaseDados” apresenta o estado das saídas e entradas digitais do PLC.



```

Imports MySql.Data.MySqlClient

Public Class Form1
    Dim cn As New MySqlConnection   'assegura a ligacao TCP/IP
    Dim cmd As New MySqlCommand      ' Enviar SQL
    Dim dr As MySqlDataReader       ' Guarda a resposta do MySQL server
    Dim Ordem, Y0, Y1, Y2, X0, X1, X2, X3 As Integer
    Dim fs As New FaconSvr.FaconServer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        cn.ConnectionString = "database=reservatorio ; server=localhost; user=root"
        cn.Open()      ' Estabele ligacao com o MySQL server, e BD
        cmd.Connection = cn      ' Command passa a conhecer o CN

        fs.OpenProject("c:\reservatorio.fcs")

        Timer1.Interval = 1000
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        ControloReservatorio_To_PLC()
        PLC_To_SupervisaoReservatorio()
    End Sub

    Private Sub ControloReservatorio_To_PLC()
        'Le a tabela "ControloReservatorio" da base de dados "Reservatorio"
        cmd.CommandText = "SELECT * FROM controloreservatorio" ' Pretendo todas as linhas e colunas
        dr = cmd.ExecuteReader()

        dr.Read()
        Ordem = dr("Ordem")
        Y0 = dr("Y0")
        Y1 = dr("Y1")
        Y2 = dr("Y2")

        txtBD_to_PLC.Text = Ordem & "      " & Y0 & "      " & Y1 & "      " & Y2 & vbCrLf
        dr.Close()

        ' Escreve no PLC, os valores lidos na Base de dados: Y0,Y1 e Y2
        fs.SetItem("Channel10.Station0.Group0", "Y0", Y0)
        fs.SetItem("Channel10.Station0.Group0", "Y1", Y1)
        fs.SetItem("Channel10.Station0.Group0", "Y2", Y2)
    End Sub

```

```

Private Sub PLC_To_SupervisaoReservatorio()
    'Le os itens do PLC X0,X1,X2,X3,Y0,Y1,Y2
    ' e escreve na tabela "SupervisaoReservatorio" da base de dados "Reservatorio"
    X0 = fs.GetItem("Channel0.Station0.Group0", "X0")
    X1 = fs.GetItem("Channel0.Station0.Group0", "X1")
    X2 = fs.GetItem("Channel0.Station0.Group0", "X2")
    X3 = fs.GetItem("Channel0.Station0.Group0", "X3")
    Y0 = fs.GetItem("Channel0.Station0.Group0", "Y0")
    Y1 = fs.GetItem("Channel0.Station0.Group0", "Y1")
    Y2 = fs.GetItem("Channel0.Station0.Group0", "Y2")

    Dim valores As String
    valores = " Y0=" & Y0 & ","
    valores = valores & ",Y1=" & Y1 & ","
    valores = valores & ",Y2=" & Y2 & ","
    valores = valores & ",X0=" & X0 & ","
    valores = valores & ",X1=" & X1 & ","
    valores = valores & ",X2=" & X2 & ","
    valores = valores & ",X3=" & X3 & ","

    'cmd.CommandText = "UPDATE supervisaoreservatorio SET Y0='1',Y1='1',Y2='1',X0='1',X1='1',
    cmd.CommandText = "UPDATE supervisaoreservatorio SET " & valores & " WHERE Leitura=1"
    txt_PLC_to_BaseDados.Text = cmd.CommandText
    cmd.ExecuteNonQuery()
End Sub

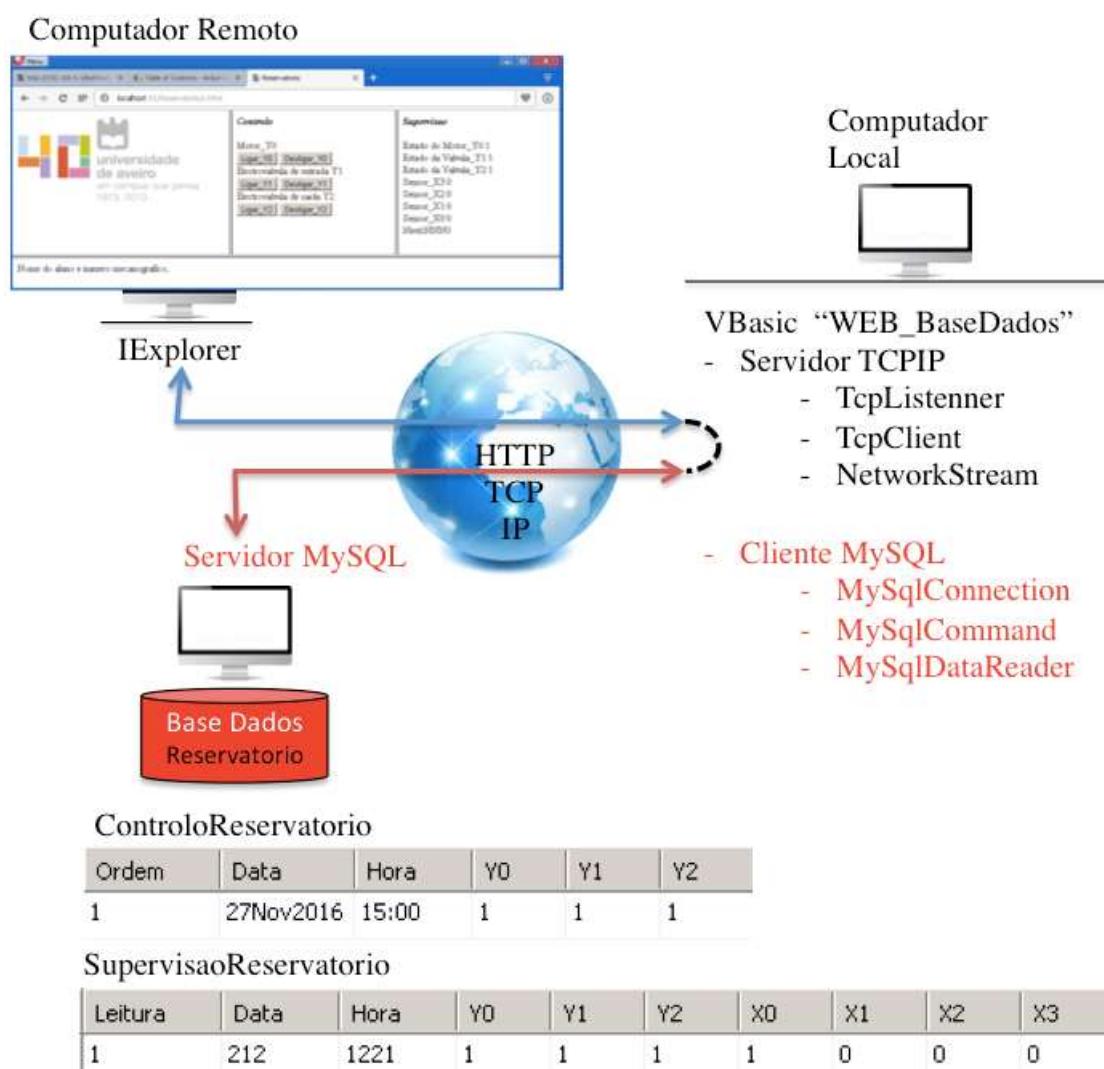
End Class

```

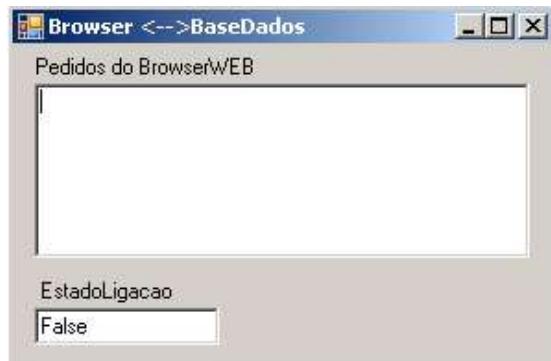
12.7.4. Exemplo III: Do BrowserWEB à BaseDados (HTTP/TCP/IP + MySQL)

Neste exemplo, pretende-se criar em VBasic um servidor WEB que permita escrever na tabela “ControloReservatorio” os pedidos enviados pelo Browser, por exemplo, fazer com que Y0 da tabela fique com o valor 1, quando o utilizador premir o botão “LigarY0” e Browser enviar o pedido “GETLigarY0...”.

Pretende-se também, que o Browser possa receber os dados da tabela “SupervisaoReservatorio” e os visualize na frame da direita “Supervisao”.



Crie um projecto em VBasic com o nome “**Trab12_Browser_To_Database**”. A interface da aplicação desenvolvida neste exemplo é a da figura seguinte. Tem apenas duas textbox, a textbox de cima permite visualizar os pedidos do BrowserWEB (ex. GET) . A textbox de baixo permite visualizar o estado da ligação TCP/IP.



O código é o seguinte:

```
Imports MySql.Data.MySqlClient
Imports System.Net
Imports System.Net.Sockets ' permite usar TcpClient, TcpListener, NetworkStream
Imports System.IO ' manipulacao de ficheiros em disco
Imports System.Text ' encoding converte array bytes em string

Public Class Form1

    Dim cn As New MySqlConnection 'assegura a ligacao TCP/IP
    Dim cmd As New MySqlCommand ' Enviar SQL
    Dim dr As MySqlDataReader ' Guarda a resposta do MySQL server
    Dim Leitura, Ordem, Y0, Y1, Y2, X0, X1, X2, X3 As Integer
    Dim valores As String

    Dim EsperaLigacaoTCP As New TcpListener(81) 'espera pedidos de ligacao TCPIP
    Dim LigacaoTCP As New TcpClient 'controla a ligacao TCPIP
    Dim Mensagem As NetworkStream 'recebe ou envia array de bytes, usando o TcpClient
    Dim caracteres(4000) As Byte
    Dim nCaracteres As Integer
    Dim textorecebido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.L
        cn.ConnectionString = "database=reservatorio ; server=localhost; user=root"
        cn.Open() ' Estabele ligacao com o MySQL server, e BD
        cmd.Connection = cn ' Command passa a conhecer o CN

        EsperaLigacaoTCP.Start() ' comega à escuta da port 81
        Timer1.Interval = 1000
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.
        If EsperaLigacaoTCP.Pending() = True Then ' Há ligacao à espera de ser aceite
            LigacaoTCP = EsperaLigacaoTCP.AcceptTcpClient() ' aceita ligacao TCPIP
            Mensagem = LigacaoTCP.GetStream() ' Le msg recebida no formato NetwokStream
        End If
    End Sub

```

```

txtEstadoLigacao.Text = LigacaoTCP.Connected ' retorna o estado da ligacao TCP

If LigacaoTCP.Connected = True Then
    nCaracteres = LigacaoTCP.Available ' há bytes recebidos ?
End If

If nCaracteres > 0 Then
    Mensagem.Read(caracteres, 0, nCaracteres) 'Le os bytes recebidos

    ' Converte o array de bytes "caracteres", numa string "textorecebido"
    textorecebido = Encoding.Default.GetChars(caracteres)
    textorecebido = Mid(textorecebido, 1, 40)

    'Atualizar Itens do PLC na tabela ControloReservatorio
    If InStr(textorecebido, "Ligar_Y0") Then
        valores = " Y0='1'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If
    If InStr(textorecebido, "Desligar_Y0") Then
        valores = " Y0='0'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If
    If InStr(textorecebido, "Ligar_Y1") Then
        valores = " Y1='1'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If
    If InStr(textorecebido, "Desligar_Y1") Then
        valores = " Y1='0'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If
    If InStr(textorecebido, "Ligar_Y2") Then
        valores = " Y2='1'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If
    If InStr(textorecebido, "Desligar_Y2") Then
        valores = " Y2='0'"
        cmd.CommandText = "UPDATE ControloReservatorio SET " & valores & " WHERE Ordem=1"
        cmd.ExecuteNonQuery()
    End If

    TextBox1.Text = TextBox1.Text + textorecebido + vbCrLf

    ' Se na mensagem recebida constar a a palavra MotorOn ou MotorOff
    ' deve ser aberto o ficheiro e enviado par a browser
    If InStr(textorecebido, "MotorOn.html") Then
        EnviarFicheiro("MotorOn.html")
    ElseIf InStr(textorecebido, "MotorOff.html") Then
        EnviarFicheiro("MotorOff.html")
    ElseIf InStr(textorecebido, "Controlo.html") Then
        EnviarFicheiro("Controlo.html")
    ElseIf InStr(textorecebido, "Supervisao.html") Then
        EnviarFicheiro("Supervisao.html")
    ElseIf InStr(textorecebido, "Reservatorio.html") Then
        EnviarFicheiro("Reservatorio.html")
    ElseIf InStr(textorecebido, "Reservatorio2.html") Then
        EnviarFicheiro("Reservatorio2.html")
    ElseIf InStr(textorecebido, "Logo.html") Then
        EnviarFicheiro("Logo.html")
    ElseIf InStr(textorecebido, "Rodape.html") Then
        EnviarFicheiro("Rodape.html")
    End If

    LigacaoTCP.Close()      ' O Protocolo HTTP prevê o fim da ligacao TCP
    LigacaoTCP = New TcpClient
    nCaracteres = 0
    textorecebido = ""
End If
End Sub

```

```

Private Sub EnviarFicheiro(ByVal sss As String)
    If LigacaoTCP.Connected = True Then
        Dim estadoY0, estadoY1, estadoY2, estadoX0, estadoX1, estadoX2, estadoX3, Nivel As String
        Dim ss As String
        Dim ler As StreamReader
        ler = New StreamReader(ss)

        ' Le ficheiro/documento HTML e guarda o texto na string "ss"
        ss = ler.ReadToEnd()

        ' Le os Itens da tabela SupervisaoReservatorio
        cmd.CommandText = "SELECT * FROM SupervisaoReservatorio" ' Pretendo todas as linhas e colunas
        dr = cmd.ExecuteReader()
        dr.Read()
        Leitura = dr("Leitura")
        estadoY0 = dr("Y0")
        estadoY1 = dr("Y1")
        estadoY2 = dr("Y2")
        estadoX0 = dr("X0")
        estadoX1 = dr("X1")
        estadoX2 = dr("X2")
        estadoX3 = dr("X3")
        dr.Close()

        ' Altera string "ss" , troca os caracteres "YYY0" que estiverem no documento HTML
        ' pelo estado da saida Y0 do PL ("0 ou "1")
        ss = Replace(ss, "YYY0", estadoY0)
        ss = Replace(ss, "YYY1", estadoY1)
        ss = Replace(ss, "YYY2", estadoY2)
        ss = Replace(ss, "XXX0", estadoX0)
        ss = Replace(ss, "XXX1", estadoX1)
        ss = Replace(ss, "XXX2", estadoX2)
        ss = Replace(ss, "XXX3", estadoX3)

        'Converte a string "ss" num array de bytes, de acordo com tabela ASCII
        Dim arraybytes() As Byte = Encoding.Default.GetBytes(ss)

        ' Envia o array de bytes para o Browser
        Mensagem.Write(arraybytes, 0, Len(ss))
        ler.Close()
        ss = ""
    End If
End Sub

End Class

```

12.7.5. Exemplo IV: Do Browser ao PLC, com Base de dados

Crie a base de dados **reservatorio** com as tabelas **controloreservatorio** e **supervisaoreservatorio** como mostrado nas figuras seguintes:

The screenshot shows the MySQL Workbench interface with the 'controloreservatorio - Table' tab selected. The table definition is as follows:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ordem	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
data	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
hora	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y0	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y1	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y2	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

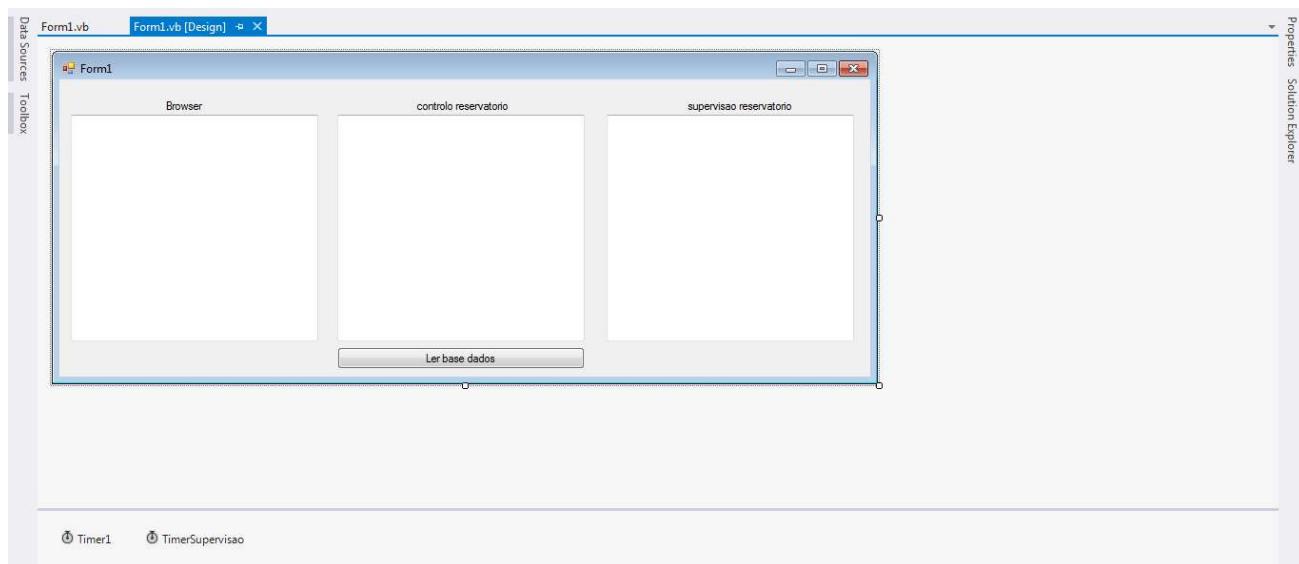
Below the table definition, there is a section for adding new columns, and at the bottom, tabs for Columns, Indexes, Foreign Keys, Triggers, Partitioning, Options, and buttons for Apply, Revert, Context Help, and Snippets.

The screenshot shows the MySQL Workbench interface with the 'supervisaoreservatorio - Table' tab selected. The table definition is as follows:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
leitura	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
data	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
hora	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y0	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y1	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Y2	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
X0	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
X1	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
X2	INT(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Below the table definition, there is a section for adding new columns, and at the bottom, tabs for Columns, Indexes, Foreign Keys, Triggers, Partitioning, Options, and buttons for Apply, Revert, Context Help, and Snippets.

A aplicação final deve ter o seguinte aspeto:



O código é o seguinte:

```
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Text

Imports MySql.Data.MySqlClient
'Must add the MySql.Data.dll file (see http://www.visual-basic-tutorials.com/Tutorials/MsCodes/code8.htm)
'It should be located in one of these folders:
'C:\Program Files (x86)\MySQL\Samples and Examples 5.6.15
'C:\Program Files (x86)\MySQL\MySQL Connector Net 6.6.4\Assemblies\v2.0
'C:\Program Files (x86)\MySQL\MySQL Connector Net 6.6.4\Assemblies\v4.0
'C:\Program Files (x86)\MySQL\MySQL Installer for Windows

Public Class Form1
    '
    ' Outras variáveis
    '
    Dim caminho_pasta_html As String = "C:\Users\mike\Google Drive \Informática_Industrial_2016-2017\Práticas\Aula13\ExerciciosMRIem\TesteCasa\html"
    Dim caminho_pasta_facon_fcs_file = "C:\Users\mike\Google Drive \Informática_Industrial_2016-2017\Práticas\Aula13\ExerciciosMRIem\TesteCasa\"

    '
    ' TCP / IP
    '
    Dim EsperaLigacaoTCP As New TcpListener(81)
    Dim LigacaoTCP As New TcpClient
    Dim mensagem As NetworkStream
    Dim mensagem_out As NetworkStream
    Dim caracteres(4000) As Byte
    Dim nCaracteres As Integer
```



```

'texto_recebido = Encoding.Default.GetBytes(CStr(carateres)) ' Esta
alternativa não sei porque não funciona

'Copiar para a caixa de texto (só para visualização do último
pedido)
TextBox1.Text = texto_recebido

texto_recebido = Mid(texto_recebido, 1, 40) 'Analizar apenas a
primeira parte da mensagem

'-----
' ----- Processamento da string texto_recebido -----
'-----

'-----
' Atuação de saídas para o FaconSrv
'-----
Dim order_given As Boolean = False ' checks if an order has been
given
Dim v_a_controlar() As String = {"Y0", "Y1", "Y2"}

For j As Integer = 0 To v_a_controlar.Length - 1
    If InStr(texto_recebido, "Ligar_" + v_a_controlar(j)) Then
        fs.SetItem(CSG, v_a_controlar(j), 1)
        order_given = True
    ElseIf InStr(texto_recebido, "Desligar_" + v_a_controlar(j))
Then
        fs.SetItem(CSG, v_a_controlar(j), 0)
        order_given = True
    End If
Next j

'-----
' If an order was given then write to database
'-----
If order_given Then
    Dim date_now As String = Date.Now.ToShortDateString
    Dim hour_now As String = Date.Now.ToShortTimeString
    Dim Y0 As Integer = fs.GetItem(CSG, "Y0")
    Dim Y1 As Integer = fs.GetItem(CSG, "Y1")
    Dim Y2 As Integer = fs.GetItem(CSG, "Y2")

    ' Insert new entry in table controloreservatorio
    'Ver exemplo aqui http://www.w3schools.com/sql/sql\_insert.asp

    'cmd.CommandText = "Insert into dados_al
(idDados_A1, Nome, Telefone) Values('" + Txt_id.Text + "','" + Txt_nome.Text +
"', '" + Txt_telefone.Text + "')"
    cmd.CommandText = "Insert into controloreservatorio
(data,hora,Y0,Y1,Y2) Values('" + date_now + "','" + hour_now + "','" + CStr(Y0) +
"', '" + CStr(Y1) + "','" + CStr(Y2) + "')"
    cmd.ExecuteNonQuery()
End If

'-----
'Envio das páginas html solicitadas (devidamente alteradas)
'Usar a variável caminho_pasta_html definida em cima
'-----
If InStr(texto_recebido, "MotorOn.html") Then
    EnviarFicheiro(caminho_pasta_html + "\MotorOn.html")
ElseIf InStr(texto_recebido, "MotorOff.html") Then
    EnviarFicheiro(caminho_pasta_html + "\MotorOff.html")
ElseIf InStr(texto_recebido, "Supervisao.html") Then
    EnviarFicheiro(caminho_pasta_html + "\Supervisao.html")
ElseIf InStr(texto_recebido, "Controlo.html") Then
    EnviarFicheiro(caminho_pasta_html + "\Controlo.html")
ElseIf InStr(texto_recebido, "Reservatorio.html") Then

```

```

        EnviarFicheiro(caminho_pasta_html + "\Reservatorio.html")
    End If

    LigacaoTCP.Close() ' o protocolo http prevê o fim da ligação
    LigacaoTCP = New TcpClient
End If
End Sub

]  Private Sub EnviarFicheiro(ByVal caminho_do_ficheiro As String)
    If LigacaoTCP.Connected Then
        '-----
        'Conversão de stream para string
        '-----
        Dim stream_do_ficheiro As StreamReader = New StreamReader
(caminho_do_ficheiro)
        Dim string_do_ficheiro As String = stream_do_ficheiro.ReadToEnd

        '-----
        'Alteração da string substituindo os códigos (e.g., VALOR_Y0) pelo
valor lido do FaconSrv
        '-----
        Dim v_a_mon() As String = {"X0", "X1", "X2", "X3", "Y0", "Y1", "Y2"}
        For i As Integer = 0 To v_a_mon.Length - 1
            string_do_ficheiro = Replace(string_do_ficheiro, "VALOR" +
v_a_mon(i), fs.GetItem(CSG, v_a_mon(i)))
        Next i

        '-----
        'Conversão de string para array de bytes e envio
        '-----
        Dim bytes_do_ficheiro() As Byte = Encoding.Default.GetBytes
(string_do_ficheiro)
        mensagem.Write(bytes_do_ficheiro, 0, Len(string_do_ficheiro))

    End If
End Sub

]  Private Sub ButtonLerBaseDados_Click(sender As Object, e As EventArgs)
Handles ButtonLerBaseDados.Click
    ' Leitura sequencial dos dados existentes na base de dados
    Dim str As String = ""

    '-----
    'Leitura da Tabela controloreservatorio
    '-----

    'Selecção da tabela da base de dados para leitura dos dados
    'Exemplo http://www.w3schools.com/sql/sql\_select.asp
cmd.CommandText = "SELECT * FROM controloreservatorio"
data_reader = cmd.ExecuteReader

    ' Leitura sequencial da tabela...
    ' Guarda o conteúdo dos registos na variável str
    While data_reader.Read
        str = str & data_reader("ordem") & " " & data_reader("data") & " "
& data_reader("hora") & " " & data_reader("Y0") & " " & data_reader("Y1") & "
" & data_reader("Y2") & vbCrLf
    End While

    ' Visualiza os dados de todos os registos
    TextBoxControloReservatorio.Text = str
    data_reader.Close()

    '-----
    'Leitura da Tabela supervisaoreservatorio
    '-----

```

```

        str = "" 'limpar variável str

        'Selecção da tabela da base de dados para leitura dos dados
        'Exemplo http://www.w3schools.com/sql/sql\_select.asp
        cmd.CommandText = "SELECT * FROM supervisaoreservatorio"
        data_reader = cmd.ExecuteReader

        ' Leitura sequencial da tabela...
        ' Guarda o conteúdo dos registos na variável str
        While data_reader.Read
            str = str & data_reader("leitura") & " " & data_reader("data") &
            " " & data_reader("hora") & vbTab & data_reader("Y0") & " " & data_reader("Y1") &
            " " & data_reader("Y2") & " " & data_reader("X0") & " " & data_reader("X1") &
            " " & data_reader("X2") & " " & data_reader("X3") & vbCrLf
        End While

        ' Visualiza os dados de todos os registos
        TextBoxSupervisaoReservatorio.Text = str
        data_reader.Close()

    End Sub

] Private Sub TimerSupervisao_Tick(sender As Object, e As EventArgs) Handles
TimerSupervisao.Tick

    Dim date_now As String = Date.Now.ToShortDateString
    Dim hour_now As String = Date.Now.ToShortTimeString
    Dim Y0 As Integer = fs.GetItem(CSG, "Y0")
    Dim Y1 As Integer = fs.GetItem(CSG, "Y1")
    Dim Y2 As Integer = fs.GetItem(CSG, "Y2")
    Dim X0 As Integer = fs.GetItem(CSG, "X0")
    Dim X1 As Integer = fs.GetItem(CSG, "X1")
    Dim X2 As Integer = fs.GetItem(CSG, "X2")

    Dim X3 As Integer = fs.GetItem(CSG, "X3")

    ' Insert new entry in table controloreservatorio
    'Ver exemplo aqui http://www.w3schools.com/sql/sql\_insert.asp

    cmd.CommandText = "Insert into supervisaoreservatorio
    (data,hora,Y0,Y1,Y2,X0,X1,X2,X3) Values('" + date_now + "','" + hour_now + "','" +
    + CStr(Y0) + "','" + CStr(Y1) + "','" + CStr(Y2) + "','" + CStr(X0) + "','" + CStr(X1) +
    + "','" + CStr(X2) + "','" + CStr(X3) + "')"
    cmd.ExecuteNonQuery()
End Sub
End Class

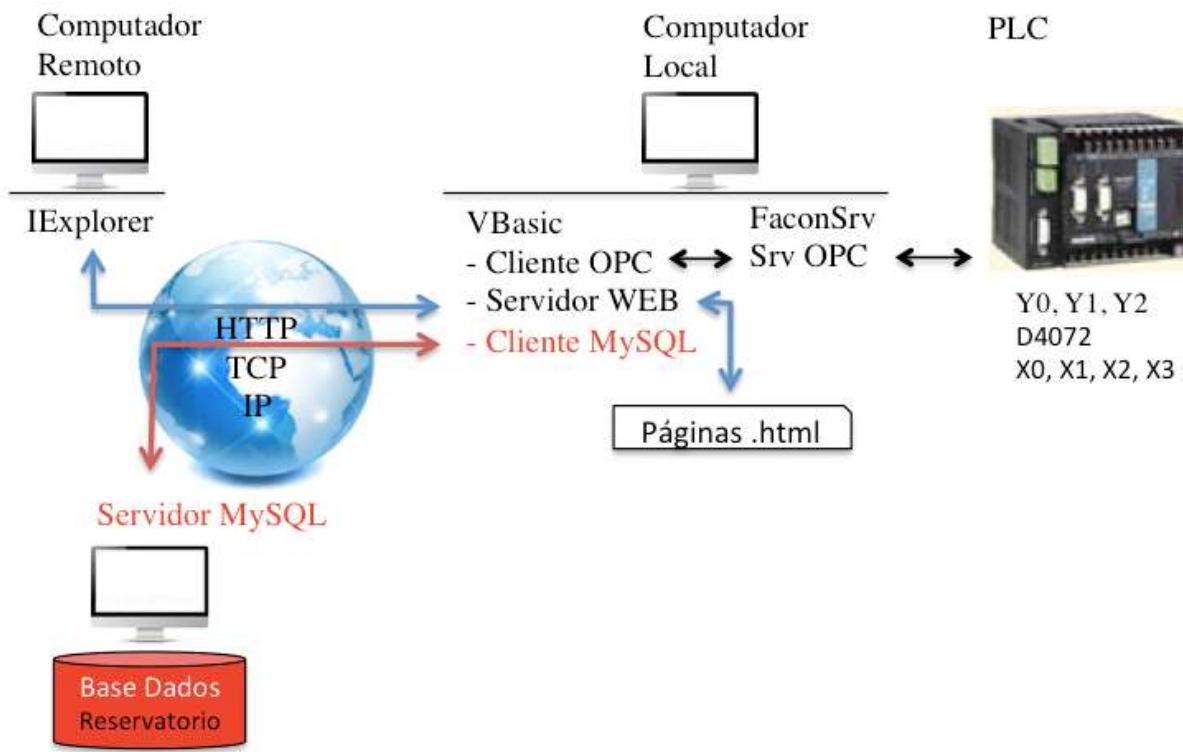
```



Informática Industrial 2022/2023

Trabalho Prático nº 11 - SCADA remoto do PLC c/ Base de Dados MySql

Neste trabalho, vamos continuar a monitorizar e a controlar remotamente o funcionamento do PLC “Reservatório”, mas os dados transferidos, de e para o PLC, devem ser armazenados numa base de dados MySQL.



Conforme a figura ilustra, além do PLC que controla o processo industrial, dispomos de três computadores. A partir do computador remoto o utilizador pode usar um Browser WEB para monitorizar e controlar o PLC/Reservatório (SCADA). O **computador local** possui duas aplicações: a aplicação desenvolvida em **Visual Basic** e o **FaconSrv**. Utilizando o **FaconSrv como servidor OPC**, a aplicação VB pode ler e escrever nos Itens do PLC como no trabalho nº 7. Utilizando objetos do tipo **TcpClient**, **TcpListener**, **NetworkStream**, e **StreamReader**, a aplicação VB pode atuar como um servidor WEB, respondendo aos pedidos HTTP do Browser, com o conteúdo dos documentos HTML disponíveis no computador local (como no trabalho nº11).

Neste trabalho, a aplicação **VB** utiliza objetos do tipo **MySQLConnection**, **MySQLCommand** e **MySQLDataReader** para aceder ao computador/ServidorMySQL, ler e escrever na base de dados “Reservatório”.

Introdução

Recorda-se que a partir da aplicação remota (IExplorer), usada no trabalho anterior, era possível controlar as saídas do PLC: Y0(Motor), Y1(EV_in) e Y2(EV_out); e monitorizar o estado das entradas digitais: X0,X1,X2,X3 (sensores de nível).

O controlo e a monitorização eram implementadas através da aplicação local, desenvolvida em Visual Basic:

- Recebia pela internet as ordens enviadas pelo “Browser WEB” (mensagens HTTP/TCP/IP), e através do Faconsrv controlava as saídas do PLC (Y0, Y1 e Y2).
- Através do Faconsrv, lia o estado das saídas e das entradas digitais do PLC (Y0, Y1, Y2, X0, X1, X2, X3) e enviava essa informação para o “Browser WEB”.

Saídas digitais do PLC:

- Y0 - Controlo do motor “Y0_Motor”
- Y1 - Controlo da electroválvula de entrada de água no reservatório “Y1_EV_In”.
- Y2 - Controlo da electroválvula de saída de água do reservatório “Y2_EV_Out”.

Entradas digitais do PLC:

- X0 - sensor “X0_AlarmEmpty”,
- X1 - sensor ”X1_Empty”,
- X2 - sensor ”X2_Full” e
- X3 - sensor ”X3_AlarmMax”.

Assume-se que os sensores ficam ativos quando detectam água fazendo com que o PLC, na entrada digital respectiva, receba 24 V.

Objectivo

Neste trabalho pretende-se guardar na base de dados MySQL o estado das saídas e das entradas do PLC ao longo do tempo, de minuto a minuto ou sempre que uma delas se altere.

Aplicação local

Deve alterar a aplicação local, desenvolvida no trabalho anterior, para que além de fazer o que fazia, possa aceder também ao gestor de base de dados “MySQL server”.

Crie uma base de dados “Reservatorio” com duas tabelas (Relações): “SupervisaoReservatorio” e “ControloReservatório”.

Sempre que a aplicação remota enviar uma ordem para ativar ou desativar uma das saídas do PLC, a aplicação local deve inserir na tabela “ControloReservatorio” uma nova linha (registo) com o novo valor pretendido para todas as saídas Y0, Y1, e Y2, bem como com a data e a hora em que a ordem (mensagem TCP/IP) foi recebida. Cada linha da tabela tem um número único (#ordem) que é incrementado sempre que chega uma nova ordem.

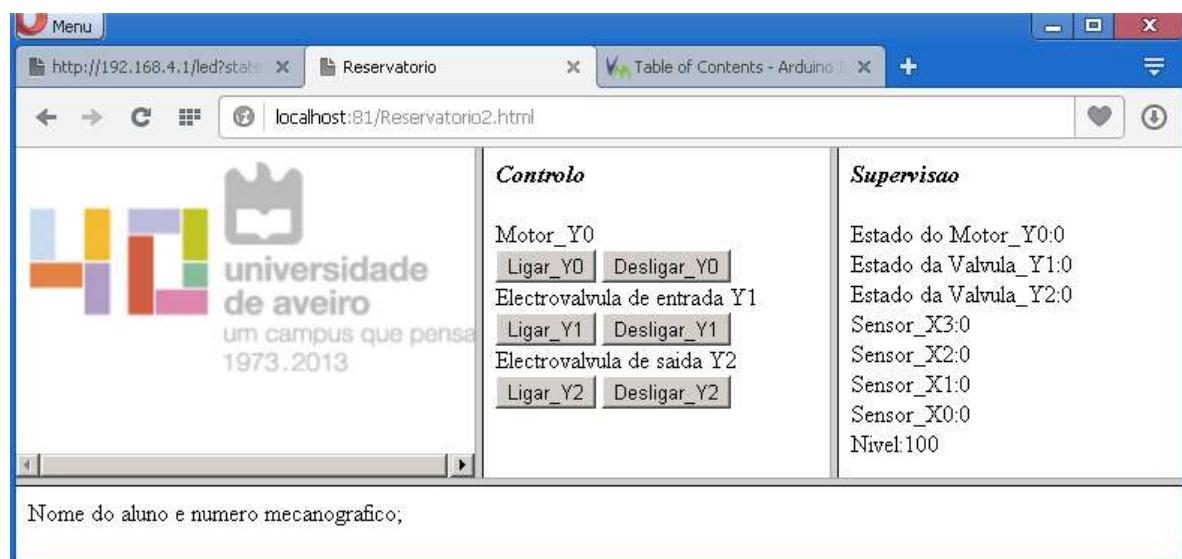
ControloReservatorio(#ordem, Data, Hora,Y0, Y1, Y2)

Sempre que uma das entradas ou saídas digitais Y0, Y1, Y2, X0, X1, X2, X3 for alterada, a aplicação local deve guardar o valor de todas elas na tabela “SupervisaoReservatorio”

SupervisaoReservatorio(#leitura, Data, Hora, Y0, Y1, Y2, X0, X1, X2, X3)

Aplicação remota

A aplicação remota não deve ser alterada, deve continuar a ter a mesma interface e as mesmas funcionalidades.



Conhecimentos a adquirir

Introdução aos gestores de bases, aos objetos em Visual Basic necessários para aceder aos gestores de bases de dados, e às mensagens SQL.

Familiarização com:

- Linguagem SQL (Structured Query Language)

Tutorial: <http://www.w3schools.com/sql/>

- MySQL

MySQL **Server** (Gestor de base de dados)

MySQL phpMyAdmin (Cliente remoto da base de dados)

MySQL **Connector/NET** (Controlo VBasic)

Tutorial: <http://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-intro.html>

Para instalar o “MySQL Server” instale o XAMPP , e para instalar o “Connector/NET”, aceda a: <https://dev.mysql.com/downloads/windows/installer/5.7.html>

Importante:

- Neste trabalho o PLC não tem programa, todo o controlo e supervisão é efectuado a partir do computador/Visual Basic remoto, com a intermediação da aplicação local.
- O trabalho será avaliado por questionário individual, na semana seguinte à entrega do mesmo.
- Deve enviar para o elearning os programas desenvolvidos sob pena da nota obtida no questionário não ser considerada.

Bibliografia:

Database Access with Visual Basic® .NET, Third Edition, By [Jeffrey P. McManus](#), [Jackie Goldstein](#), Addison Wesley, 2003 - Capítulo 4: ADO.NET Data Provider

MySQL Stored Procedure Programming, By Steven Feuerstein, Guy Harrison Publisher: O'Reilly Pub Date: March 2006 - Capítulo 17

Database Access with Visual Basic® .NET, Third Edition, By [Jeffrey P. McManus](#), [Jackie Goldstein](#), Addison Wesley, 2003 - Capítulo 6: ADO.NET - The DataAdapter

MySQL **Connector/NET** (Controlo VBasic)

Tutorial: <http://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-intro.html>



CAPÍTULO

12

PHP

JPSantos
2023

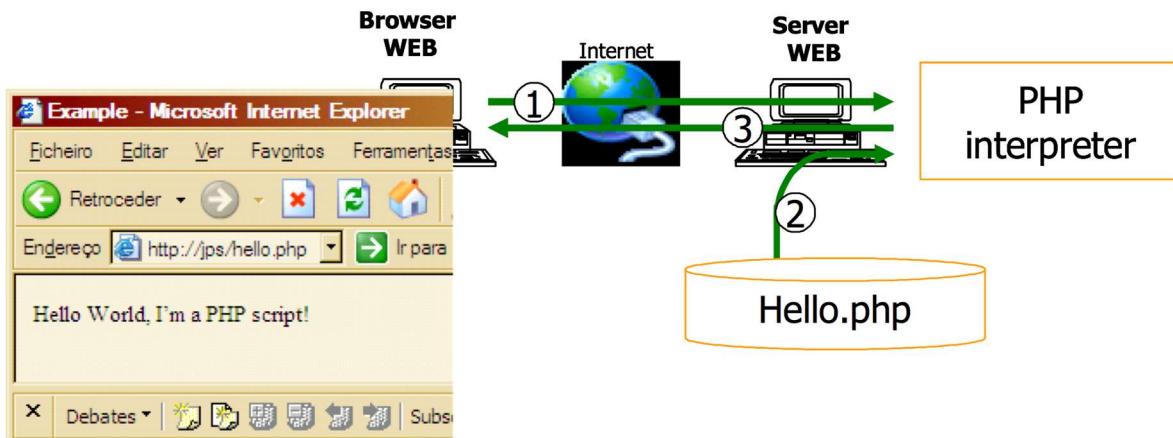
13. PHP

13.1. Conceitos

O *PHP (Hypertext Preprocessor)* é uma linguagem simples, gratuita e aberta que permite elaborar páginas dinâmicas para a WEB. É possível escrever e gravar no disco, de um *ServidorWEB*, simples ficheiros de texto utilizando a linguagem PHP. Os ficheiros de texto escritos nesta linguagem não são interpretados pelo servidor WEB mas sim por outro programa por ele designado, neste caso o *PHP_interpreter*. Os programas escritos nesta linguagem podem ser inseridos dentro do código HTML.

O exemplo seguinte apresenta um ficheiro de texto do tipo *php* intitulado “Hello.php” guardado no disco duro do servidor. Quando este ficheiro é pedido pelo *BrowserWEB*, através do protocolo HTTP, o servidor WEB abre esse ficheiro e envia-o para o “*PHP_interpreter*”(figura 12). O “*PHP_interpreter*”que também está instalado no computador servidor, interpreta o ficheiro e converte para HTML as instruções PHP. Neste caso, o resultado da conversão consiste no texto “*Hello World, I'm a PHP script*”, o servidor, por sua vez, reenvia esse texto para o *BrowserWEB* numa mensagem HTTP de resposta.

Para que as instruções PHP sejam processadas pelo “*PHP_interpreter*” e convertidas em HTML devem estar contidas na etiqueta <?php ?>



```
// ficheiro C:\Inetpub\wwwroot\Hello.php
<html>
<head>
    <title>Example</title>
</head>
<body>

<?php
echo "Hello World, I'm a PHP script!";
?>
</body>
</html>
```

figura: Página PHP – Hello World

O programa que interpreta as páginas/programas PHP pode ser instalado em várias **plataformas** (Windows, Unix, Linux) e pode interagir com vários **servidores WEB** (IIS, Apache, Caudium, Netscape, OmniHTTPD, Sambar, Xitami,...)

Nos exemplos seguintes os programas PHP estão gravados no “home directory” do *ServidorWEB*. No caso de um servidor IIS/Microsoft o “home directory” é o “C:\Inetpub\wwwroot”. No caso de um servidor Apache2 o “home directory” é por defeito o “C:\Programas\Apache Group\Apache2\htdocs”. No caso do XAMPP a “home directory” é o diretório “C:\Programas\XAMPP\htdocs”.

Os browsers WEB não possuem um interpretador de PHP. Para que o documento PHP seja convertido em HTML e o Browser o possa apresentar ao utilizador, o browser tem de aceder a um servidor WEB com um interpretador de PHP instalado:

<http://localhost/hello.php>

O documento tem de ter uma extensão *.php e pode conter apenas etiquetas HTML , apenas etiquetas PHP ou ambas, de forma alternada.

```
<HTML>
<BODY>
<! Comentário em HTML >
```

Zona HTML


```
<?php
/* Os comentários em PHP podem ocupar várias linhas
   segunda linha de comentário */
echo "Zona PHP <BR>"; // Isto é um comentário: a instrução echo converte para HTML o que estiver entre
aspas
?>

<i>Outra zona HTML <BR></i>

<?PHP
echo "Outra Zona PHP <BR>";
?>

</BODY>
</HTML>
```

13.2. Tipos de variáveis

O nome das variáveis em PHP começa sempre por um cifrão \$ seguido de um conjunto de caracteres alfanuméricos. Existe apenas uma limitação, depois do cifrão, o primeiro caracter alfanumérico não pode ser um número. O interpretador de PHP distingue as letras minúsculas e maiúsculas no nome de uma variável.

```
<?php
$a = 1.234;           // cria uma variável
$b = 1.2e3;
$c = 7E-10;

// a instrução "echo" gera uma string HTML
echo $a . "<BR>". $b. "<BR>". $c . "<BR>"; // neste caso igual: 1.234<BR>1200<BR>7.0E-10<BR>

// a instrução "array" cria um array, neste caso um array com dois elementos, o primeiro elemento contém a
string "bar" e o segundo elemento contém o valor "true". Os índices deste array em vez do habitual 1,2,3.. são
[foo] e [12]
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"];      // o elemento $arr["foo"] contém o valor "bar"
echo "<BR>";          // gera o texto <BR> que provoca a mudança de linha
echo $arr[12];         // o elemento $arr[12] contém o valor 1
echo "<BR>";

$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));
echo $arr["somearray"][6]; // 5
echo "<BR>";
echo $arr["somearray"][13]; // 9
echo "<BR>";
echo $arr["somearray"]["a"]; // 42
?>
```

13.3. Tipos de operadores

13.3.1. Operadores aritméticos

Em PHP os operadores aritméticos são: +, - , * , / , %

O exemplo seguinte ilustra a sintaxe e finalidade dos operadores aritméticos.

```
<?PHP
$a = 10;
$b = 5;
echo $a + $b;           // retorna 15
echo "<BR>";           // muda de linha
echo $a - $b;           // retorna 5
echo "<BR>";
echo $a * $b;           // retorna 50
echo "<BR>";
echo $a / $b;            // retorna 2
echo "<BR>";
$b = 3;
echo $a / $b;           // retorna 3,3333
echo "<BR>";
echo $a % $b;           // retorna 1
echo $a % $b;           // retorna 1
?>
```

13.3.2. Operadores relacionais/comparação

Os operadores booleanos comparam dois valores e com base na comparação realizada retornam um valor booleano “Verdadeiro” ou “Falso”.

Maior que	>
Maior ou igual	≥
Menor que	<
Menor ou igual	≤
Igual	==
Diferente	!= ou <>

```
<?PHP
$A=10;
$B=15;
$C=20;

echo $A < $B;           // O resultado da comparação é TRUE, 10 é menor que 15
echo "<BR>";
echo $B > $A;           // O resultado da comparação é TRUE, 15 é maior que 10
echo "<BR>";
echo $A != $B;          // O resultado da comparação é TRUE, 10 é diferente de 10
echo "<BR>";
echo $A == $A;           // O resultado da comparação é TRUE, 10 é igual a 10
?>
```

13.3.3. Operadores lógicos

Intercepção de conjuntos	(AND)	and ou &&
Reunião de conjuntos	(OR)	or ou
Negação	(NOT)	!
Ou exclusivo	(XOR)	xor

```
<?PHP  
$A=TRUE;  
$B=FALSE;  
  
echo ($A and $B); // O resultado é FALSE  
echo ($B or $A); // O resultado é TRUE  
echo (!$B); // O resultado é TRUE  
echo ($A xor $A); // O resultado FALSE  
?>
```

13.3.4. Operadores bit a bit

AND bit a bit	&
OR bit a bit	
Complemento bit a bit	~
Desloca um bit para esquerda	<<
Desloca um bit para a direita	>>
XOR bit a bit	^

O operador “**&**” tem dois operandos e faz uma intercepção lógica bit a bit dos dois operandos

```
<?PHP  
$a = 18; // 18 é igual ao número binário 00010010  
$b = 170; // 170 é igual ao número binário 10101010  
echo $a & $b; // Retorna o valor 00000010 = 2 decimal  
?>
```

O operador “**|**” tem dois operandos e faz uma reunião lógica bit a bit dos dois operandos,

```
<?PHP  
$a = 18; // 18 é igual ao número binário 00010010  
$b = 170; // 170 é igual ao número binário 10101010  
echo $a | $b; // Retorna o valor 10111010 = 186 decimal  
?>
```

O operador **^** permite fazer o XOR de dois bytes, bit a bit. Só quando dois bits são diferentes o resultado é um bit a “1”

```
<?PHP  
$a = 18; // 18 é igual ao número binário 00010010  
$b = 170; // 170 é igual ao número binário 10101010  
echo $a ^ $b; // Retorna o valor 10111000 = 184 decimal  
?>
```

O operador Complemento “`~`” altera todos os bits do byte, os bits que valiam 0 passam a valer 1 e vice-versa.

<?PHP

```
?>  
$a = 18;           // 18 é igual ao número binário      00010010  
echo ~$a;         // Retorna o valor          11101101 = 237 = -19 decimal
```

Os operadores de deslocamento “`>>`” e “`<<`” deslocam os bits para a esquerda ou direita. No exemplo seguinte, os bits são deslocados uma posição para a direita e depois são deslocados duas posições para a esquerda

<?PHP

```
?>  
$a = 18;           // 18 é igual ao número binário      00010010  
echo $a >> 1;    // desloca para a direita um bit e retorna o valor 00001001 = 9 decimal  
echo "<BR>";     // Muda de linha  
$a = 18;           // 18 é igual ao número binário      00010010  
echo $a << 2;    // desloca para a esquerda dois bits e retorna o valor 01001000 = 72 decimal
```

13.4. Estruturas de controlo

A execução de um programa é sequencial, cada instrução é gravada na memória do uC em posições consecutivas e é executada pela ordem com que foi gravada, a menos que o programador tenha inserido no próprio código instruções que permitam alterar essa sequência de execução.

13.4.1. WHILE

A instrução WHILE(*condição*){ ... } permite repetir a execução de algumas linhas código, mais exactamente das linhas de código que tenham sido escritas entre as chavetas desta instrução. Estas linhas de código são repetidas enquanto a *condição* booleana do WHILE for verdadeira.

A *condição* do ciclo WHILE pode conter vários operandos e operadores mas o resultado final das operações tem de ser um valor booleano, verdadeiro ou falso. Enquanto a condição for verdadeira o ciclo repete-se.

| | | |
|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------|
| <code>while (<i>condição</i>)
{
 <i>instruções</i>;
 <i>instruções</i>;
}</code> | <code>\$i = 0;
while (\$i < 3){
 \$i = \$i+1;
}
echo \$i;</code> | Depois de 3 ciclos, retorna \$i=3 |
|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------|

13.4.2. FOR

O ciclo FOR{ ... } permite repetir a execução das linhas de código que estiverem escritas entre as chavetas do próprio ciclo. Este ciclo é semelhante ao ciclo WHILE, contudo a “condição” do ciclo WHILE pode assumir um valor falso em função da execução das instruções do próprio ciclo, fazendo com que o ciclo possa terminar mais cedo ou tarde, ao passo que o ciclo FOR tem um número de repetições predefinido.

| | | |
|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|-----------------------------------|
| <code>for (\$i=0; \$i < n; \$i++)
{
 <i>instruções</i>;
 <i>instruções</i>;
}</code> | <code>for (\$i=0; \$i< 3; \$i++){
}
echo \$i;</code> | Depois de 3 ciclos, retorna \$i=3 |
|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|-----------------------------------|

13.4.3. IF

Quando se pretende executar algumas linhas de código se, e apenas se “*IF*”, uma determinada condição for verdadeira e pelo contrário “*Else*” executar outras linhas de código se a condição for falsa, o programador pode usar a instrução IF.

| | |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <code>if (<i>condição</i>)
{
 //se condição verdadeira
 <i>instruções</i>;
} else {</code> | <code>\$i= 5;
if(\$i < 10)
{
 //Se verdadeira
 echo "verdadeiro";</code> |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|

```

//se condição falsa
instruções;
}
} else{
//Se falsa
echo "falso";
}

```

Em suma, esta instrução permite que apenas um de dois conjuntos de instruções seja executado, em função de uma *condição* ser verdadeira ou falsa.

13.4.4. SWITCH

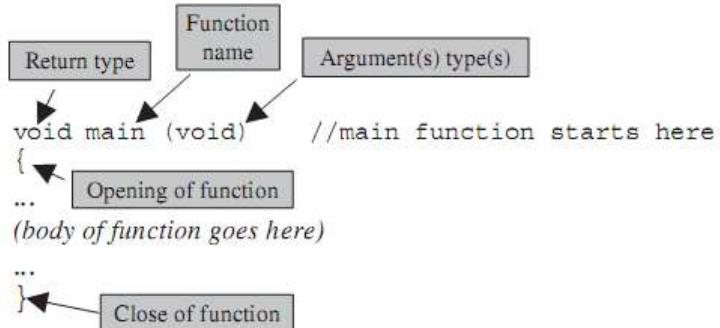
A instrução SWITCH permite que apenas um de muitos conjuntos de instruções seja executado, em função do valor numérico ou alfanumérico de uma variável.

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> \$ i = 1; switch (\$i) { case 1: instruções; break; case 2: instruções; break; default: instruções; } </pre> | <pre> \$ i = 2; switch (\$i) { case 1: echo "1"; break; case 2: echo "2"; break; default: echo "default"; } </pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> \$ s = 'a'; switch (\$s) { case 'a': instruções; break; case 'b': instruções; break; default: instruções; } </pre> | <pre> \$ s = 'a'; switch (\$s) { case 'b': echo "b"; break; case 'a': echo "a"; break; default: echo "default"; } </pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|

13.5. Funções e procedimentos

As funções e os procedimentos permitem “agrupar” código, tornando mais fácil a sua organização e reutilização ao longo do programa, mas só as funções podem retornar valores.



13.5.1. Passagem de parâmetros por valor

As funções podem receber parâmetros por valor. Isto significa que quando a função é chamada o valor do(s) parâmetros é copiado para outra zona de memória do computador e essa zona de memória tem o(s) nomes definidos na função.

No exemplo seguinte, a função *duplica* recebe um parâmetro que é copiado para a zona de memória/variável \$i. Esta variável existe na memória do computador enquanto a função estiver a ser executada. Neste exemplo a função *echo* devolve o valor *10*.

```
<?php  
duplica(5);  
  
function duplica($i)  
{  
    echo $i*2;  
}  
  
?>
```

13.5.2. Passagem de parâmetros por referência

As funções e os procedimentos também podem receber parâmetros por referência. Neste caso o valor dos parâmetros não é copiado para uma nova zona de memória associada à execução da função, apenas é enviado como parâmetro(s) os endereços de memória onde esses valores se encontram armazenados. De facto passam a existir dois nomes diferentes para a mesma zona de memória.

No exemplo seguinte a variável *\$str* e a variável *\$string* correspondem à mesma zona de memória do computador. Desta forma quando dentro da função a variável *\$string* é alterada estamos também a alterar a variável *\$str*. É por essa razão que a função *echo \$str* devolve o valor *Isto é uma string, e alguma coisa mais*.

```

<?php

$str = 'Isto é uma string,';
concatena($str);
echo $str;      // devolve 'Isto é uma string, e alguma coisa mais.'

function concatena(&$string)
{
$string = $string . ' e alguma coisa mais';
}

?>

```

O exemplo seguinte ilustra a passagem de um parâmetro por referência, mais exactamente é passado o endereço de memória do array \$i[]. Dessa forma, dentro da função, o array \$input dentro da função refere-se à mesma zona de memória que a variável \$i[]

```

<?php

$i[2];          // criação do array $i[2], este array tem dois elementos
$i[0]=1;        // é atribuído ao primeiro elemento de array o valor 1
$i[1]=2;        // é atribuído ao segundo elemento de array o valor 2

soma($i);       // é chamada a função "soma" e é passada a referência ao array $i

function soma(&$in)    // parâmetro passado por referência "&"
{ // dentro da função, o array $in corresponde à mesma zona de memória do array $i[]
echo $in[0];    // devolve 1, pois o elemento $in[0] é o mesmo que $i[0]
echo $in[1];    // devolve 2, pois o elemento $in[1] é o mesmo que $i[1]
}

?>

```

13.5.3. Retorno de valores

As funções podem retornar valores para o programa principal, a partir do qual foram chamadas. O exemplo seguinte ilustra uma função que recebe dois parâmetros por valor, ou por outras palavras recebe dois valores e devolve o resultado da soma desses dois valores.

```

<?php

$resultado= soma(5,7); // a função soma retona o valor da soma dos dois argumentos 12
echo $resultado;        // a função echo devolve o conteúdo da variável $resultado 12

function soma($i,$j)
{
return $i+$j;
}

?>

```

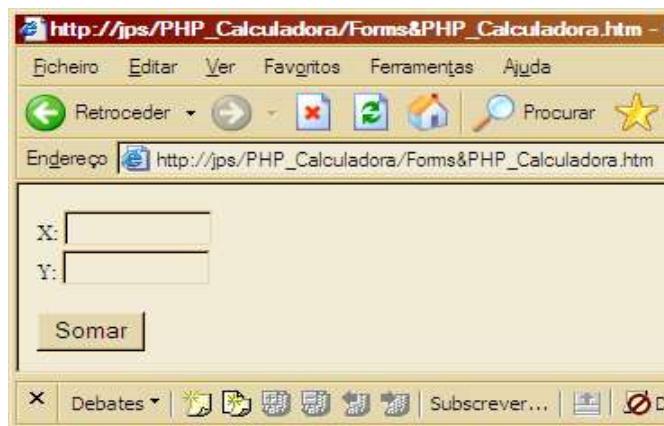
13.6. PHP - Calculadora

Este exemplo apresenta uma calculadora desenvolvida em HTML e PHP. Esta calculadora apenas permite adicionar dois números.

O código em HTML necessário à criação da interface (“Form”) está gravado no servidor, mais exactamente no ficheiro “Calculadora.html” (canto inferior esquerdo), e o código PHP necessário ao processamento dos dados introduzidos nessa interface está gravado no ficheiro “Calculadora.php” (lado direito).

Para entendermos este exemplo necessitamos de saber utilizar a interface “form” disponível em HTML bem como as entradas “input” do tipo texto “Text”. Podemos ver no exemplo seguinte que uma das entradas de texto tem o nome “name” X e a outra o nome Y. Podemos também verificar que o utilizador ao seleccionar o botão “Somar” está de facto a executar (Action) o ficheiro escrito em PHP que irá processar os valores previamente introduzidos pelo utilizador no seu *BrowserWEB*.

Quando o ficheiro PHP é executado no lado do servidor, ele pode aceder aos valores introduzidos pelo utilizador na interface “Form” usando o array `$_POST[nome do input da interface HTML]`. Neste caso os elementos `$_POST['X']` e `$_POST['Y']` contêm os valores introduzidos pelo utilizador nas entradas X e Y da interface. A variável \$Z recebe o resultado da soma.



//ficheiro C:\Inetpub\wwwroot\Calculadora.php

```
<html>
<body>

<?php
$x = $_POST['X']
$y = $_POST['Y']
$Z=$x +$y;
echo $x,"+",$y,"=",$Z;
?>

<br><a href="Calculadora.html" >tentar outra vez
</a>
</body>
</html>
```

//ficheiro C:\Inetpub\wwwroot\Calculadora.html

```
<html>
<body>

<form action="Calculadora.php" method="post" >
X: <input type=text lenght="2" name="X" size="10"><br>
Y: <input type=text length="2" name="Y" size="10">
<br><br>
<input type="submit" value="Somar">
</form>

</body>
</html>
```

figura: Página PHP – Calculadora

13.7. PHP – Acesso às bases de dados MySQL

A linguagem PHP permite estabelecer ligações TCP/IP com o gestor de bases de dados “MySQL server” e enviar-lhe comandos SQL. Quando o programa “MySQL server” recebe comandos de texto SQL na porta TCP 3306 executa esses comandos. Ao executar esses comandos o servidor MySQL pode por exemplo criar novas bases de dados, com tabelas, campos e registo.

13.7.1. Interação entre o Browser WEB, o Servidor WEB e o Servidor MySQL

Neste exemplo, vamos considerar a utilização de três computadores ligados à Internet. Num deles reside o Browser WEB, no outro reside o Servidor WEB e no terceiro reside o Servidor MySQL. As páginas WEB escritas em PHP são pedidas pelo Browser WEB mas são executadas/interpretadas no servidor WEB. Por sua vez o interpretador de páginas PHP, usado neste exemplo, estabelece uma ligação TCP/IP com o servidor MySQL e envia-lhe pedidos/comandos SQL, recebe as respostas do servidor MySQL e converte-as para a linguagem HTML. Nessa altura o servidor WEB envia essas páginas para o Browser WEB onde o utilizador poderá visualizar os dados presentes na base de dados MySQL.

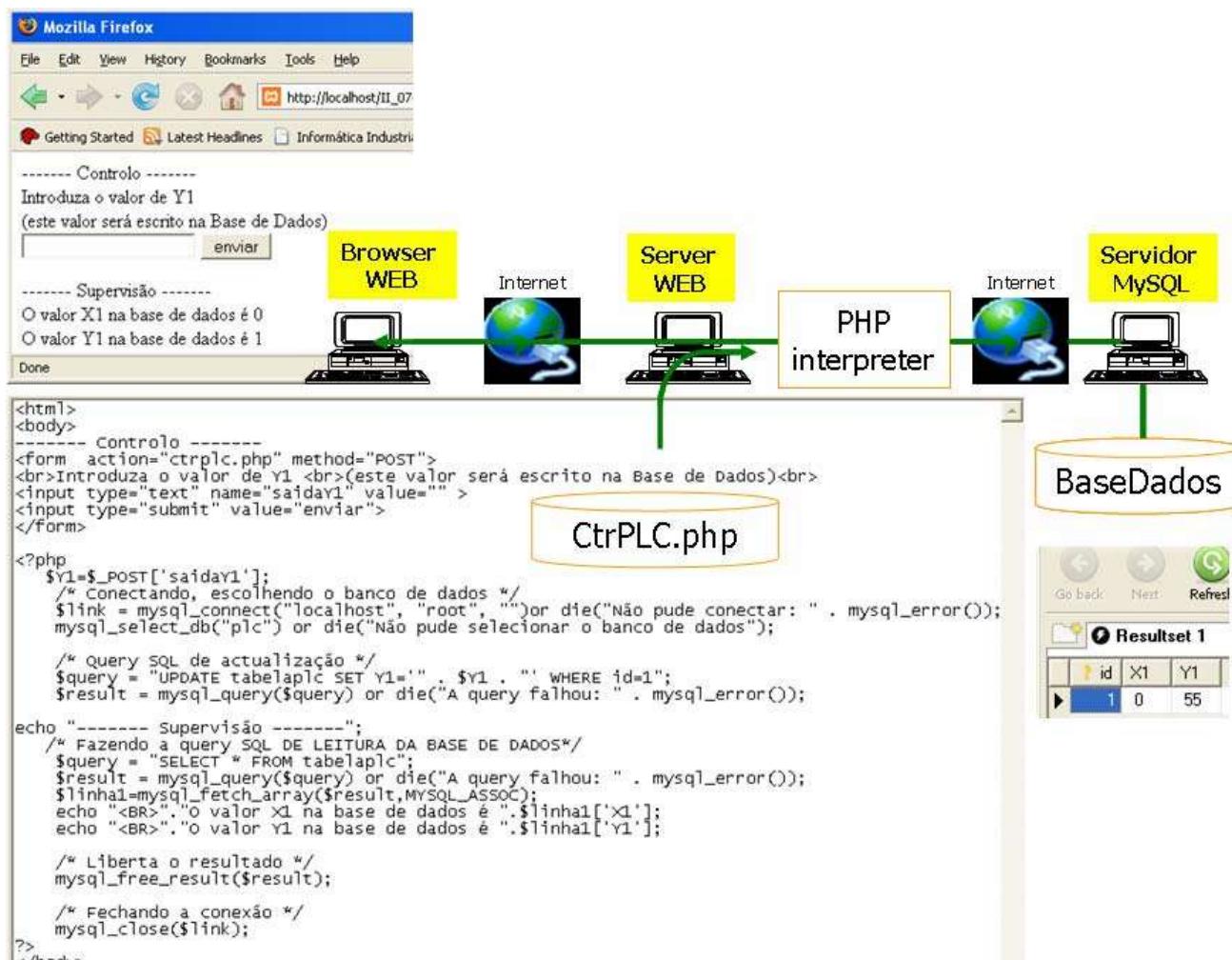


figura: PHP – Interacções entre o browser WEB, o servidor WEB e o servidor MySQL

Ficheiro “CtrPlc.php” (usando o msqli)

```
<html>
<body>
----- Controlo -----
<form action="ctrplc.php" method="POST">
Introduza os valores de Y1, <br>Este valor sera escritos na base de dados PLC"<br>
na tabela "tabalaplc (id, X1, Y1) "<br><br>
<input type="text" name="saidaY1" value="1" >
<input type="submit" value="enviar">
</form>

<?php

$Y1=$_POST['saidaY1'];

/* Conectando, escolhendo o banco de dados */
$link = mysqli_connect("localhost", "root", "") or die("Nao pude conectar: ". mysqli_error());
mysqli_select_db($link,"plc") or die("Nao pude selecionar o banco de dados");

/* Query SQL de actualizacao */
$query = "UPDATE tabelaplc SET Y1='".$Y1 . "' WHERE id=1";
$result = mysqli_query($link,$query) or die("A query falhou: ". mysqli_error());

echo "----- Supervisao -----<br>Os valores lidos na base de dados sao:";

/* Fazendo a query SQL DE LEITURA DA BASE DE DADOS*/
$query = "SELECT * FROM tabelaplc";
$result = mysqli_query($link,$query) or die("A query falhou: ". mysqli_error());
$linha1=mysqli_fetch_array($result,MYSQLI_ASSOC);
echo " Y0:".$linha1['X1'];
echo " Y1:".$linha1['Y1'];

/* Liberta o resultado */
mysqli_free_result($result);

/* Fechando a conex,,o */
mysqli_close($link);
?>
</body>
</html>
```

13.7.2. Funções PHP para aceder ao Servidor MySQL

`mysql_affected_rows` -- Devolve o número de linhas afectadas na operação anterior com o MySQL
`mysql_change_user` -- Muda o utilizador
`mysql_client_encoding` -- Retorna o nome do conjunto de caracteres
`mysql_close` -- Fecha a ligação com o MySQL
`mysql_connect` -- Abre uma ligação com o servidor MySQL
`mysql_create_db` -- Cria um base de dados MySQL
`mysql_data_seek` -- Move o ponteiro interno do resultado
`mysql_db_name` -- Retorna os nomes das bases de dados
`mysql_db_query` -- Envia uma query ao MySQL
`mysql_drop_db` -- Apaga uma base de dados do MySQL
`mysql_errno` -- Retorna o valor numérico da mensagem de erro da operação anterior do MySQL
`mysql_error` -- Retorna o texto da mensagem de erro da operação anterior do MySQL
`mysql_escape_string` -- Escapa uma string para uso com o mysql_query.
`mysql_fetch_array` -- Retorna uma linha do resultado da query sob forma de uma matriz associativa, matriz numérica ou ambas.
`mysql_fetch_assoc` -- Retorna uma linha do resultado e sob a forma de uma matriz associativa
`mysql_fetch_field` -- Retorna informação sobre uma coluna de um resultado como um objeto
`mysql_fetch_lengths` -- Retorna o tamanho de cada campo do resultado
`mysql_fetch_object` -- Retorna uma linha do resultado sob a forma de um objeto
`mysql_fetch_row` -- Retorna uma linha do resultado sob a forma de uma matriz numérica
`mysql_field_flags` -- Pega as flags do campo especificado no resultado
`mysql_field_len` -- Retorna o tamanho do campo
`mysql_field_name` -- Retorna o nome do campo especificado no resultado de uma query
`mysql_field_seek` -- Move o ponteiro do resultado para um campo especificado
`mysql_field_table` -- Retorna o nome da tabela onde está o campo especificado
`mysql_field_type` -- Retorna o tipo do campo especificado em um resultado de query
`mysql_free_result` -- Liberta a memória do resultado de uma query
`mysql_get_client_info` -- Retorna informação da versão do cliente MySQL
`mysql_get_host_info` -- Retorna informação sobre o host do MySQL
`mysql_get_proto_info` -- Retorna informação do protocolo do MySQL
`mysql_get_server_info` -- Retorna informação do servidor MySQL
`mysql_info` -- Retorna informação sobre a última query
`mysql_insert_id` -- Retorna o ID gerado da operação INSERT anterior
`mysql_list_dbs` -- Lista as bases de dados disponíveis no servidor do MySQL
`mysql_list_fields` -- Lista os campos de uma tabela
`mysql_list_processes` -- Lista os processos MySQL
`mysql_list_tables` -- Lista as tabelas de uma base de dados MySQL
`mysql_num_fields` -- Retorna o número de campos do resultado
`mysql_num_rows` -- Retorna o número de linhas em um resultado
`mysql_pconnect` -- Abre uma conexão persistente com um servidor MySQL
`mysql_ping` -- Pinga uma ligação ou restabelece a ligação se não houver ligação
`mysql_query` -- Realiza uma query MySQL
`mysql_real_escape_string` -- Escapa os caracteres especiais numa string para usar em um comando SQL, levando em conta o conjunto actual de caracteres.
`mysql_result` -- Retorna dados do resultado
`mysql_select_db` -- Seleciona um banco de dados MySQL
`mysql_stat` -- Retorna o status actual do sistema
`mysql_tablename` -- Retorna o nome da tabela do campo
`mysql_thread_id` -- Retorna o ID da thread actual



```
<html>
<body>
----- Controlo -----
<form action="ctrplc.php" method="POST">
Introduza os valores de Y0, Y1, e Y2 <br>Estes valores serao escritos na base de dados "Reservatorio,"<br>
na tabela "SupervisaoReservatorio(Leitura,Data,Hora,Y0,Y1,Y2,X0,X1,X2,X3)"<br><br>
Y0:<input type="text" name="saidaY0" value="I" >
Y1:<input type="text" name="saidaY1" value="I" >
Y2:<input type="text" name="saidaY2" value="I" >
<input type="submit" value="enviar">
</form>

<?php
$Y0=$_POST['saidaY0'];
$Y1=$_POST['saidaY1'];
$Y2=$_POST['saidaY2'];

/* Conectando, escolhendo o banco de dados */
$link = mysqli_connect("localhost", "root", "")or die("N.,o pude conectar: ". mysqli_error());
mysqli_select_db($link, "Reservatorio") or die("N.,o pude selecionar o banco de dados");

/* Query SQL de actualizacao */
$query = "UPDATE SupervisaoReservatorio SET Y0='' . $Y0 . "", Y1='' . $Y1 . "", Y2='' . $Y2 . "" WHERE
Leitura=1";
$result = mysqli_query($link,$query) or die("A query falhou: ". mysqli_error());

echo "----- Supervisao -----<br>Os valores lidos na base de dados sao:";

/* Fazendo a query SQL DE LEITURA DA BASE DE DADOS*/
$query = "SELECT * FROM SupervisaoReservatorio";
$result = mysqli_query($link,$query) or die("A query falhou: ". mysqli_error());
$linha1=mysqli_fetch_array($result,MYSQLI_ASSOC);
echo " Y0:".$linha1['Y0'];
echo " Y1:".$linha1['Y1'];
echo " Y2:".$linha1['Y2'];

/* Liberta o resultado */
mysqli_free_result($result);

/* Fechando a conex,,o */
mysqli_close($link);
?>
</body>
</html>
```

É necessário desenvolver um programa em VisualBasic “VBasicPLC” que possa aceder à base de dados e por sua vez contactar e controlar o PLC.

Bibliografia

Sobre Apache

Site <http://www.apache.org>

Sobre PHP

Site <http://www.php.net>

Site [Manual de PHP](#)

Carlos Serrão, Joaquim Marques - Programação com PHP 4.3, FCA – Editora de Informática Ida. – Julho 2004

Carlos Serrão, Joaquim Marques - Programação com PHP 3 e 4, FCA – Editora de Informática Ida. – Setembro 2000.
(cota 681.3.06A.214)

Sobre SQL

Introduction to SQL http://www.w3schools.com/sql/sql_intro.asp

Sobre MySQL

• [MySQL Stored Procedure Programming](#), Steven F., Guy H., 2006.

Chapter 13

• [Teach.Yourself.MySQL.in.10.Minutes](#), By Sams.Sams, 2006.

Lesson 24.

