Faculdade de Engenharia da Universidade do Porto

FEUP

COMANDO DE KIT DE CHÃO-DE-FABRICA MINIATURIZADO:

Controlo de Simulação e Supervisão

Amândio de Oliveira Campos Pereira

Dissertação realizada no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major Automação

Orientador: Prof. Dr. Armando Jorge Miranda de Sousa

Julho de 2008

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Comando de Kit de Chão-de-Fábrica Miniaturizado"

foi aprovada em provas realizadas em 16/Julho/2008

o júri

Presidente Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar, Faculdade de Engenharia da Universidade do Porto

Professor Doutor Rui Paulo Pinto da Rocha
Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Coimbra

Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar, Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - Amandio de Oliveira Campos Pereira

Faculdade de Engenharia da Universidade do Porto

Resumo

As várias disciplinas de engenharia associadas às tecnologias de gestão e sistemas de automação estão em constante evolução, sendo necessário a constante actualização dos métodos de ensino. Deste modo, a FEUP adquiriu uma miniatura que simula um chão-de-fábrica para permitir o contacto dos alunos com um sistema "real", mesmo que em miniatura. Para que este sistema real tenha um funcionamento interessante e autónomo, é necessário interceptar os sinais de controlo de e para o kit e ainda gerar sinais adicionais. A intercepção de sinais permite a simulação de avarias, atrasos de comando, enquanto que a geração de sinais virtuais tem como função acrescentar funcionalidades que o kit não tem, tal como por exemplo, dar a indicação que o trabalho numa máquina foi concluído com (ou sem) sucesso. Surge então a necessidade de criar um "Sistema de Supervisão dos Sinais de Controlo" que circulam entre o kit miniaturizado do sistema de produção e o dispositivo de comando. O dispositivo de comando será provavelmente um autómato programável que será programado pelos alunos.

Nesta dissertação serão descritos os passos dados para a criação do referido "Sistema de Supervisão dos Sinais de Controlo", serão estudadas várias tecnologias, arquitecturas e protocolos de comunicação passíveis de ser usadas neste projecto. Será projectada e implementada uma solução que será validada testando a intercepção dos sinais de comando entre um controlador implementado na ferramenta Isagraf e uma miniatura de linha de produção com duas máquinas.

A solução proposta para o "Sistema de Supervisão dos Sinais de Controlo" é genérica, autónoma, compacta, de fácil utilização e configuração. Foram implementadas funcionalidades de intercepção de sinais de sensores e de actuadores, a ligação com *encoders* incrementais e a geração de sinais adicionais com diversas funcionalidades. O sistema é facilmente configurável através de um browser de Web. Os sinais de comando a interceptar podem ser eléctricos ou ao nível da rede de campo ModBus TCP. A solução de intercepção dos sinais eléctricos apresenta isolamento eléctrico no sentido de se conseguir um sistema robusto e flexível. Os níveis eléctricos são adequados à intercepção de sinais reais de comando recebidos ou enviados para autómatos programáveis.

Abstract

The engineering disciplines associated with automation systems and management techniques are in permanent evolution, so it's necessary to constantly update the teaching methods. Because of this, FEUP has acquired a miniaturized factory floor simulator to allow students to contact whit a "real life system" even if in a miniaturized system is used. To make this real system interesting and autonomous is necessary to intercept the control signals from and to the kit and also to generate additional signals. The interception of signals allows to simulate malfunctions and to generate delays in the delivery of control commands, in the other hand, the interest in generating virtual signals is to add features that the kit does not have, this virtual signals can be used, for example, to indicate that the work in some machine has been successfully (or not) completed. As such, it is interesting to create a "Signal Control and Supervision System" that will be placed between the miniaturized production system kit and the control device. This control device will probably be a programmable automaton that will be programmed by the students.

This dissertation will describe the steps taken to design and implement the "Signal Control and Supervision System", the studies of various technologies, architectures and communication protocols which can be used in this project. Beside this, it will be designed and implemented a solution, and this solution will be validated by intercepting some control signals between the implemented controller, on the Isagraf tool, and a miniature two machines production line.

The proposed solution to the "Signal Control and Supervision System" is a generic, autonomous, compact, easy to use and to configure. There were implemented features like intercepting signals from sensors and actuators, connection of incremental encoders and the generation of additional multifunction signals. The system is easily configurable through a web browser. The signal interception can be at the physical level (electrical interception) or at the network level (Modbus TCP). The interception of electrical signals presents electrical insulation in order to archive a robust and flexible system. The electrical levels are adequate to the interception of signals send and received by programmable logic controllers.

Aos Meus Pais, amigos e família.

Agradecimentos

Ao longo deste trabalho foram vários os que contribuíram para que fosse possível a realização desta dissertação.

Agradeço em primeiro lugar ao meu orientador, o Professor Doutor Armando Jorge Sousa, por todo o apoio prestado ao longo das várias etapas desta dissertação.

Agradeço à Faculdade de Engenharia pela disponibilização de todos os meios necessários, principalmente aos técnicos Rui Fernandes e Nuno Guerra pelo apoio prestado.

Agradeço em particular ao técnico David Lobo pela ajuda no desenvolvimento das placas de interface e ao Professor Doutor Mário Sousa pela ajuda na implementação do cliente modbus.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

PC – Computador pessoal (*personal computer*)

PLC – Controlador lógico programável (*programmable logic controller*)

Scada – Sistemas de Controlo e Aquisição de Dados (*Supervisory Control And Data Acquisition*)

SO – Sistema Operativo

GPIO – Ponto de entrada/saída genérico (*General Purpose Input/Output*) configurável por software como estrada ou saídas.

RFID – Identificadores por rádio frequência.

ISR – Rotina do sistema operativo ou do driver de dispositivo que é executada aquando da ocorrência de uma interrupção, acrónimo para *Interrupt Service Routine*.

OSI – É um modelo de referência que divide os protocolos de redes de comunicações em 7 camadas bem definidas, acrónimo para *Open Systems Interconnection*.

GPL – É uma licença de distribuição de software livre, que obriga a manter o código fonte de uma aplicação de software aberto para qualquer pessoa o alterar. Do acrónimo inglês *GNU General Public License* .

Capítulo 1

1. Introdução

1.1 Motivação

O Departamento de Engenharia Electrotécnica e de Computadores (DEEC) está a melhorar o seu laboratório de automação, para o ensino e investigação nas áreas da automação e gestão industrial. Este laboratório incluirá um conjunto de sistemas e tecnologias tais como sistemas físicos de produção (chão-de-fábrica em miniatura), sistemas de identificação automática, redes de comunicações industriais, sistemas de supervisão e controlo e sistemas de simulação de erros.

A motivação principal para a realização deste trabalho resulta da importância que a utilização de sistemas "reais" (exemplo: Kit chão-de-fábrica miniatura) face a utilização de sistemas virtuais de simulação (exemplo: Aplicação a correr num PC), tem no estudo das várias tecnologias e técnicas relacionadas com a automação de sistemas. Para isso torna-se necessário a criação de uma plataforma para o controlo e supervisão de simulação, o "Sistema de Supervisão dos Sinais de Controlo", de modo a evitar o envio de comandos que possam danificar o kit, bem como a geração de erros de funcionamento e atrasos de comandos para testar num tempo aceitável a aplicação desenvolvida.

1.2 Enquadramento

O trabalho descrito neste relatório relaciona-se com a necessidade de criar um sistema capaz de gerar modos de funcionamento erróneo ou degradado para uma determinada planta a controlar, bem como efectuar a supervisão dessa mesma planta mediante a utilização de sistemas tipo SCADA. A utilização de um sistema capaz de gerar erros de funcionamento, juntamente com a criação das aplicações de controlo, trazem uma enorme vantagem a nível académico, pois permite aos alunos a compreensão das aplicações de controlo desenvolvidas, principalmente ao nível da detecção de falhas/erros de *hardware*, num tempo aceitável, isto é, passa a ser possível gerar erros na planta mediante a configuração do "Sistema de Supervisão dos Sinais de Controlo".

Para isso serão estudadas as diferentes plataformas capazes de albergar um sistema de simulação e supervisão, os diferentes métodos de efectuar a interface entre esta plataforma e os sistemas de controlo (autómatos, SoftPLC, ilhas remotas) e os kits didácticos que simulam o chão-de-fábrica, bem como a possibilidade da implementação do "Sistema de Supervisão dos Sinais de Controlo" com recurso a plataformas *web*. Para validação da solução desenvolvida será criada uma aplicação de controlo com recurso à ferramenta Isagraf da *ICS Triplex ISaGRAF Inc* para controlo de uma linha de produção miniatura com duas maquinas, das quais serão interceptados alguns sinais de controlo e gerados dois sinais virtuais para indicar que as maquinas terminaram o processo com sucesso.

1.3 Objectivos

O objectivo deste trabalho consiste em analisar e implementar uma das diferentes plataformas capazes de suportar um sistema de aquisição, simulação e controlo totalmente adaptável as diferentes situações de funcionamento e aos diferentes kits miniatura, projectar e implementar quer o algoritmo de controlo da planta, para efeitos de validação de resultados, quer a arquitectura de software a utilizar na plataforma de interface.

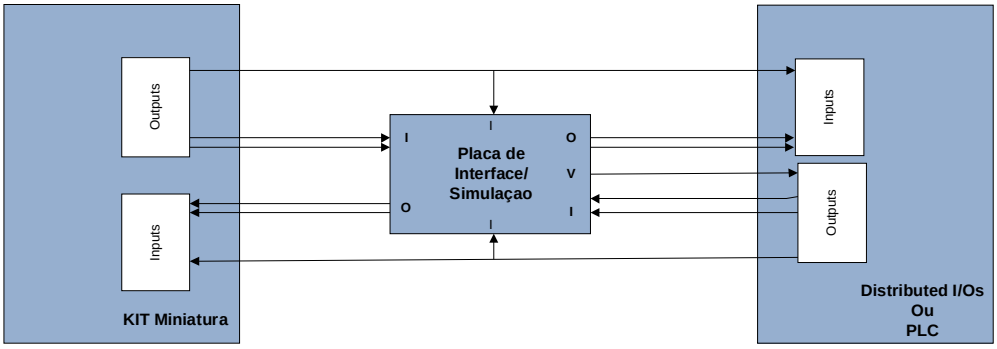


Figura 1.1 *- Esquema funcional do sistema*

Na figura 1.1 é apresentado o esquema funcional requerido para o sistema, onde são mostrados os vários modos possíveis de funcionamento da placa do "Sistema de Supervisão dos Sinais de Controlo". Esta placa deve ser capaz de interceptar os sinais eléctricos entre o sistema de controlo, tipicamente PLCs ou ilhas de IOs remotos, e o kit miniatura, para deste modo poder simular erros de comunicação de dados ou verificar se determinado conjunto de comandos irá causar ou não danos no kit físico. Esta placa deve também ser capaz de criar IOs virtuais de modo a simular sinais que o kit físico não tenha, para deste modo permitir funcionalidades do tipo sinalização de término com (ou sem) sucesso de operações de determinada máquina ou processo.

1.4 Estrutura do Relatório

Esta dissertação encontra-se estruturada em 6 capítulos dos quais, o primeiro é composto por esta introdução ao trabalho.

No segundo capitulo são apresentados os conceitos base dos sistemas de automação, são apresentados os vários kits e as respectivas características, bem como algumas das plataformas capazes de suportas o sistema de controlo de simulação.

O terceiro capítulo é composto pelo estudo de algumas tecnologias *web* bem como alguns protocolos de comunicações industriais passíveis de serem usadas no referido sistema de simulação.

No quarto capitulo são abordadas as várias arquitecturas para o software do sistema de simulação e controlo, bem como as soluções de hardware para os sistemas de isolamento de sinais eléctricos.

No quinto capítulo é abordado a implementação da aplicação de controlo do sistema de simulação, a implementação do algoritmo de controlo de uma das partes do kit e respectivos testes de limites de funcionamento do sistema.

O sexto e último capítulo contém as conclusões gerais deste trabalho, analisa os seus principais resultados, e apresenta algumas perspectivas de desenvolvimentos futuros.

Capítulo 2

2. Sistemas de automação

Ao longo dos últimos anos, a necessidade de interligar PLCs entre eles próprios, e com outros sistemas, tornou-se uma das principais directrizes. Começou com a necessidade de criar sistemas HMI baseados em PCs capazes de comunicar com os PLCs e de criar sistemas de gestão de fabrico e produção (MES). Os HMIs, que eram inicialmente unidades dedicadas que comunicavam com os PLCs através de ligações série, vieram tornar-se, devido ao desenvolvimento do PC, uma escolha bastante popular. A introdução do PC veio permitir a interligação dos sistemas de controlo com outros sistemas de níveis superiores, uma vez que estes têm uma vasta gama de periféricos de comunicações disponíveis, tais como a ethernet com TCP/IP. Permitiu também a criação de ferramentas de desenvolvimento abertas com recurso a componentes standard, ao contrário dos PLCs que são normalmente sistemas fechados e dependentes das aplicações de desenvolvimento das próprias marcas. Neste aspecto a norma IEC 61131 veio reduzir alguns dos problemas associados com as linguagens de programação proprietários.

Com o passar do tempo, os PC tornaram-se mais poderosos e resistentes. As redes de campo fizeram com que fosse possível a um PC funcionar como um controlador, mesmo que fisicamente distante do processo a controlar. Em maquinas onde existe a necessidade de controlar e adquirir dados, o PC foi o passo natural, transformando-se num PLC por software – SoftPLC. Em processos e ambiente de produção onde o PC funcionava como uma ponte entre o sistema de controlo e os sistemas SCADA, a evolução lógica foi a de atribuir mais funcionalidades ao PC tornando a interface mais simples e a implementação menos complexa, uma vez que esse sistema irá substituir vários.

De igual modo, também as redes de campo vieram revolucionar o chão de fábrica, onde os PLCs controlam os sensores e actuadores, através do envio de comandos para o barramento, tornando obsoletas as ligações analógicas. A utilização de protocolos standard, tipo TCP/IP sobre ethernet, veio permitir a interligação de todo o processo de fabrico, desde o chão de fábrica até a rede corporativa, de forma transparente e com um baixo custo.

Surge também nesta altura, a necessidade receber e enviar dados entre os processos de fabrico e os níveis de gestão e optimização de processos, para pode ser efectuada a manutenção e reconfiguração dos vários sistemas, de forma a responder mais eficientemente as necessidades actuais de

uma empresa bem como o seguimento de um produto ao longo da cadeia de produção, permitindo deste modo efectuar o historial do mesmo, desde lotes de matérias primas até aos operadores usados no fabrico desse mesmo produto. Surgem deste modo as MES ("*Manufacturing Execution Systems*").

Mas uma vez acabado o processo de fabrico, o processo de automação ainda contínua, sendo necessário transportar, armazenar e comercializar o produto. Neste ponto a introdução de tecnologias tipo código de barras e/ou RFIDs veio dinamizar o processo de identificação e localização, tornado os mesmos automáticos a medida que os produtos se deslocam pelo chão de fabrica.

Toda esta quantidade de dados relativos a processos e produtos tiveram de ser agregados em uma plataforma de gestão integrada, os ERP ("*Enterprise Resource Planning***"**), de modo a permitir um planeamento de acções a realizar e estratégias de negócio a seguir.

No seguimento dessa lógica, os sistemas de automação de processos fabris, quer sejam processos discretos ou contínuos, podem ser divididos em 3 grandes níveis (figura 2.1):

- Nível 1 – Tipicamente localizado no chão de fabrica, onde é feito controlo do sistema físico por actuação directa no mesmo.

- Nível 2 – Sistemas com o intuito de tornar o controlo do nível 1 mais simples e intuitivo para o operador, possibilitar a modificação de parâmetros de funcionamento durante o funcionamento do processo, fazer a aquisição de dados em tempo real do nível 1, para serem integrados em sistemas de interface homem maquina (HMI) tais como SCADA, consolas e para posterior envio de dados de produção para o nível superior.

- Nível 3 – Sistema de gestão de produção (MES e ERP), onde são geridos tempos de produção, encomendas, inventários, capacidade de produção, etc. [1] [2] [3]

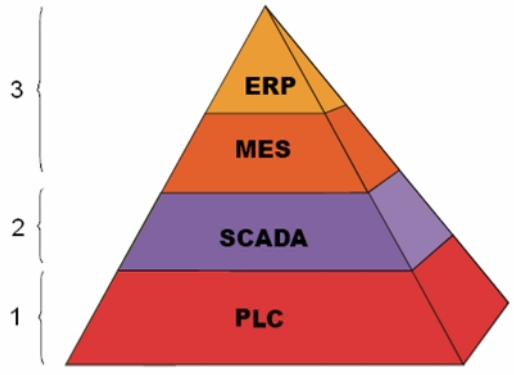


Figura 2.1 *- Pirâmide dos sistemas de automação actuais*

2.1 Arquitecturas

Os sistemas de automação são usados em todo o tipo de aplicações de controlo, desde processos de reduzidas dimensões até grandes linhas de produção, sendo os requisitos e arquitecturas associadas a cada uma dessas aplicações diferentes. A classificação das arquitecturas dos sistemas de automação são baseadas na distribuição dos sistemas de controlo usados. Desse modo existem três tipos de arquitecturas, a centralizada, descentralizada e distribuída.

Num sistema com uma arquitectura centralizada, o controlo é feito em um único dispositivo que corre o algoritmo um controlo, sendo o responsável pela leitura (sensores) e escrita (actuadores) das variáveis no processo.

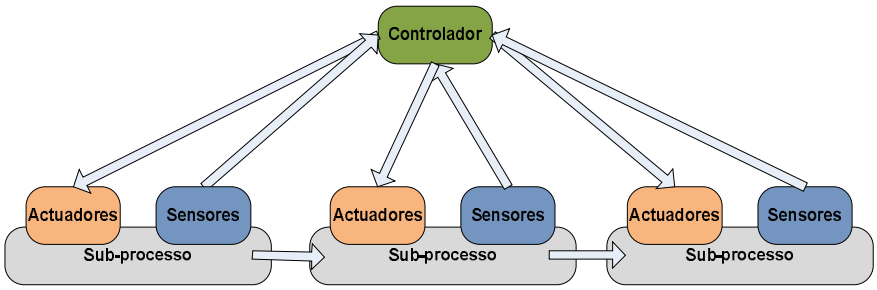


Figura 2.2 *- Exemplo de arquitectura de controlo centralizado*

Nos sistemas de controlo descentralizado, existem vários controladores distribuídos ao longo do processo, cada um responsável por uma parte do mesmo e interligados entre si por um barramento de dados para troca de informações.

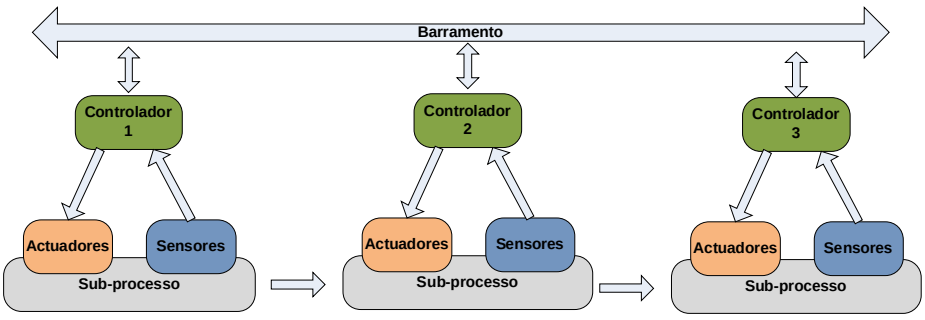


Figura 2.3 *- Exemplo de arquitectura de controlo descentralizada*

Num sistema de controlo distribuído, cada constituinte do sistema (sensor, actuador, interface, …) corre o seu algoritmo de controlo e tem a capacidade de ler e escrever dados do barramento.

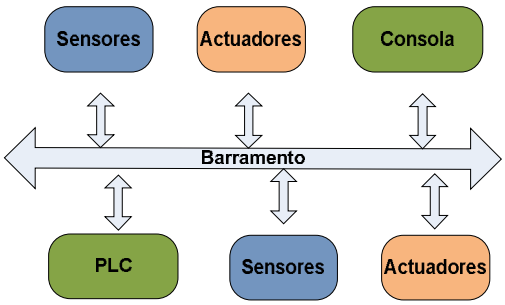


Figura 2.4 *- Exemplo de arquitectura de controlo distribuído*

2.2 Os Kit's

O sistema adquirido (Figura 2.5) funciona como um simulador físico de um chão de fábrica em miniatura, composto por 5 kit's miniatura interligados entre si.

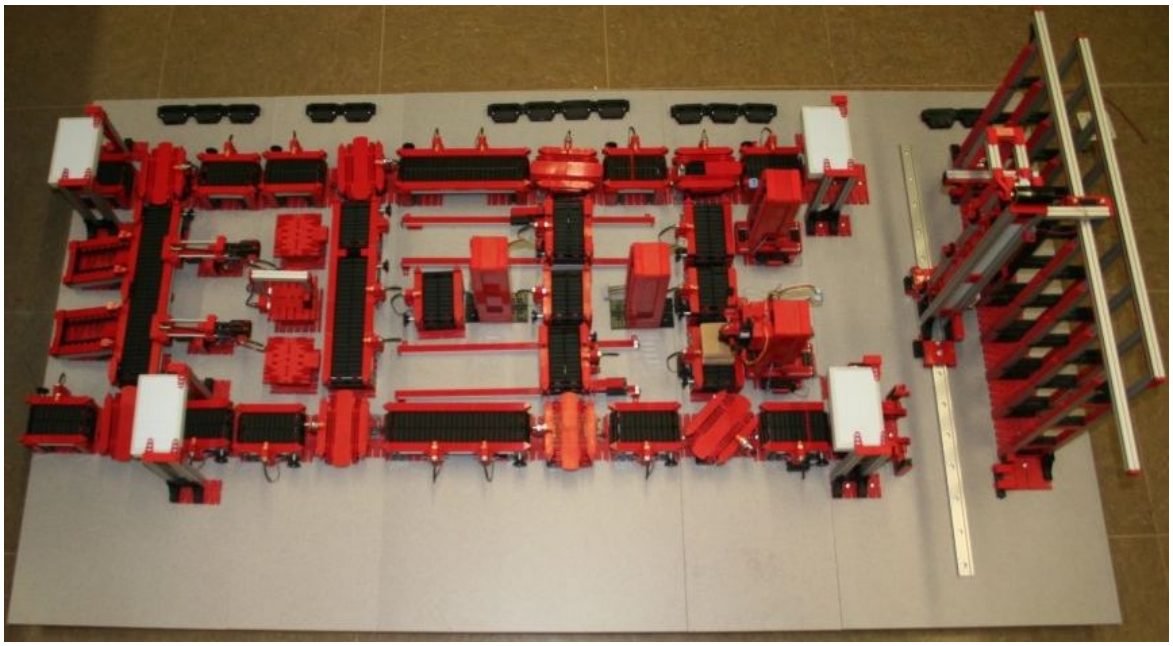


Figura 2.5 *- Vista geral do sistema completo*

Mais pormenorizadamente, o primeiro módulo, da direita para a esquerda da figura 2.5, mostra um armazém automático, com varias posições de armazenamento e leitores RFID nas entradas/saídas para leitura de informações contidas nas etiquetas das peças a trabalhar/trabalhadas. O segundo módulo é uma máquina de funcionamento em série, onde as peças são trabalhadas de

modo sequencial. O terceiro módulo é uma máquina de funcionamento em paralelo, onde existem vários recursos partilhados para transporte e maquinagem de peças. O quarto módulo é um manipulador robótico 3D com 3 postos de trabalho. O quinto e último módulo é um sistema de entrada / saída de materiais, onde os materiais são enviados para as diferentes saídas, dependo das informações lida das etiquetas RFID.

A seguinte tabela indica as principais características dos diferentes kits, tais como o número de entradas e saídas e número de periféricos de comunicação rs-232 e de *encoders*.

Tabela 2.1 - Características dos diferentes kit's

	Entradas	***Saídas***	***RS232***	***Encoder***	***Total I/Os***
ASRS	16	11	2	2	27
Serial	21	30	--	--	51
Paralel	25	32	--	--	57
3D portal	24	23	--	3	47
E/S	13	20	2	--	33

2.3 Requisitos do sistema de simulação

Pretende-se nesta dissertação projectar e desenvolver uma plataforma capaz de efectuar a interface entre o sistema de controlo e os kits, e de suportar um sistema de controlo e simulação, para isso é necessário que o sistema tenha as seguintes características:

- Configurável via web browser

- Capaz de gerar erros de funcionamento

- Receber e enviar comando modbus sobre TCP/IP

- Funcionar com os níveis de tensão requeridos pelos sistemas de automação

- Possibilidade de configuração de tempos de propagação dos sinais eléctricos entre o sistema de I/O (PLC, ilhas remotas, …) e o kit

- Permitir a interligação com *encoders* incrementais.

- Criação de saídas virtuais para por exemplo gerar um sinal para o sistema de controlo a indicar a finalização de determinada operação de uma máquina.

2.4 Estudo das alternativas de suporte ao sistema de controlo de simulação

Neste sub capitulo são dados a conhecer algumas das plataformas capazes de suportar o sistema de controlo de simulação. As principais soluções e respectivas características são referidas nos próximos tópicos.

- ICnova AP7000 Base da In-Circuit

Características:

- 140MHz Max. 200MHz
- 64MB SDRAM, 32Bit data bus
- 8MB Flash
- 2x UART
- 10/100MBit/s Ethernet,
- Regulador tensão integrado
- I2C
- Sistema operativo baseado em Linux
- 55 GPIOs
- Alimentação via USB ou conector de alimentação externa.

Figura 2.6 *- Placa ICnova AP7000*

Preço: 95 €

- NGW100 da Atmel

Características:

- 2x 10/100Mbit/s Ethernet
- 32MB SDRAM, 32Bit data bus
- 16MB Flash
- 2x USART
- I2C
- 63 GPIOS
- Sistema operativo baseado em Linux

Figura 2.7 *- Placa NGW100*

Preço: 91,45€

Capítulo 3

3. Tecnologias

Neste capítulo pretende-se dar a conhecer algumas das tecnologias mais usadas nos sistemas de automação, os vários protocolos de comunicação e suas possíveis aplicações no caso em estudo.

3.1 SCADA

Com crescimento industrial, a implementação de novos modelos de produção em série e o aumento de tamanho de máquinas a controlar, surge a necessidade de uma monitorização e controlo quer localmente, quer remotamente.

Esta necessidade de controlo e monitorização foi colmata pela introdução de interfaces homem maquina denominados de SCADA "*Supervisory Control And Data Acquisition*", esta interface tem por base um PC, onde por intermédio de software e de uma ou varias redes de comunicações, é efectua a monitorização de varias variáveis do processo fabril e o respectivo controlo sobre as mesmas, que este esteja localizado localmente ou remotamente. As leituras são tratadas e posteriormente disponibilizadas ao utilizador sob a forma de sinópticos representativos do processo, que evoluem com a alteração das variáveis físicas do processo. Estes sistemas permitem ainda, aos operadores humanos a interacção com o processo, para efectuar mudanças sempre que necessário. Criam também registos mais ou menos complexos das mudanças ocorridas no sistema, desde mudanças nos estados das variáveis até registos de alarmes, entradas de operadores. [4]

3.2 Protocolos de Comunicação

3.2.1 Modbus

O modbus é um protocolo de comunicação que assenta na camada 7 do modelo OSI (camada de aplicação), isto porque define regras para a organização e interpretação do dados. Este facto permite a comunicação entre cliente/servidor de dispositivos ligados em diferentes topologias de rede ou em diferentes tipos de barramentos de dados. Este é um protocolo aberto, criado em 1979 pela Modicon, sendo mantido actualmente pela "*Modbus organization*".

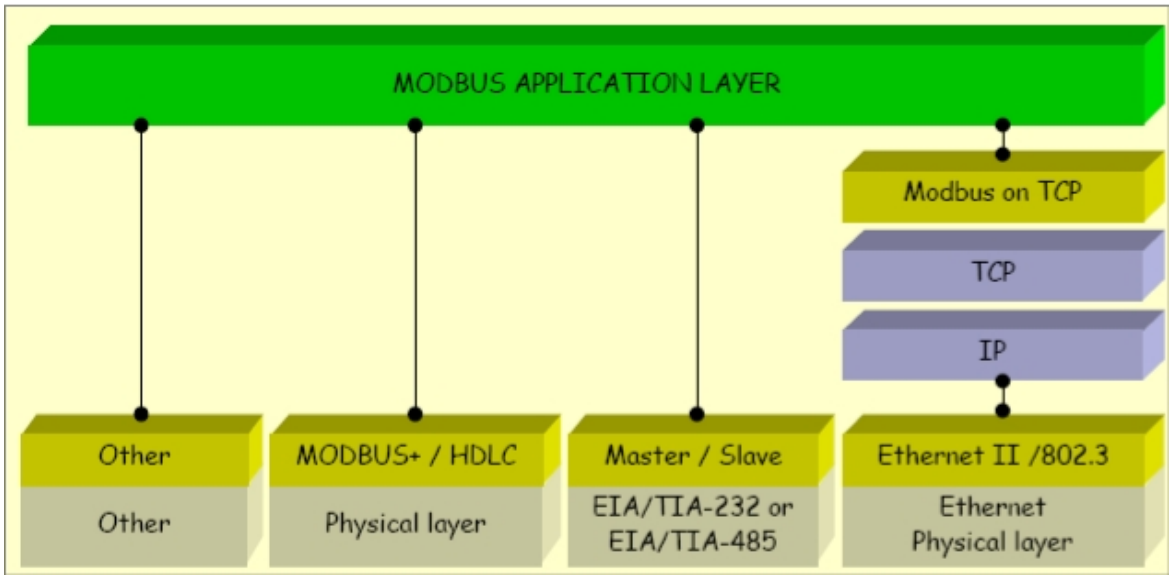


Figura 3.1 *- Modelo de referência OSI do protocolo Modbus [5]*

Actualmente, este protocolo esta implementado sobre:

- TCP/IP sobre Ethernet.
- Transmissão série assíncrona sobre uma vasta gama de meios físicos (cabo, rádio, fibra óptica).
- Modbus Plus

O modo de funcionamento do protocolo modbus assenta no esquema de cliente/servidor, onde o cliente efectua os pedidos, e o servidor responde de acordo com o pedido efectuado. Esta arquitectura pode também ser vista como uma arquitectura mestre-escravo, onde o mestre efectua os pedidos e o escravo apenas responde a estes. A figura 3.2 exemplifica o esquema de comunicação via mobus. [6]

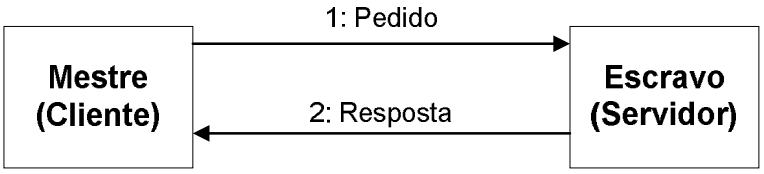


Figura 3.2 *- Esquema de comunicação mestre-escravo*

No esquema de comunicação mestre/escravo, é sempre o mestre que inicia as comunicações, podendo este enviar pedidos apenas a um escavo (*unicast*), ou a todos os escravos existentes na rede (*broadcast*), sendo que, neste caso nenhuma resposta deve ser retornada por parte dos escravos, uma vez que as mensagem de *broadcast* são necessariamente de escrita. Um escravo nunca inicia a comunicação com o mestre, nem comunica com outros escravos, apenas responde aos pedidos do mestre.

3.2.1.1 Modbus TCP/IP

O Modbus sobre TCP/IP é nada mais que uma simples trama Modbus integrada numa trama TCP/IP. Uma das vantagens da utilização do protocolo TCP ao invés do protocolo UDP, prende-se com o facto do protocolo TCP efectuar a correcção de erros, garantido assim a correcta entrega das mensagens modbus. O uso do Modbus sobre TCP/IP tem também a vantagem de estes protocolos serem compatível com uma das redes mais usadas e com maior desenvolvimento a nível mundial, a ethernet.

O grande entrave a utilização do Modbus TCP/IP para controlo crítico de sistemas, deve-se ao facto de este estar assente numa rede não determinística, isto é, uma rede onde não existem garantias temporais para a entrega de determinada mensagem.

3.2.2 Ethernet/IP

A Ethernet/IP é um protocolo de camada de aplicação industrial para aplicações em sistemas de automação industrial. Construída sobre os protocolos TCP/IP, esta interface utiliza hardware e software, amplamente usados em redes LAN e WAN, para definir um protocolo de camada de aplicação para a configuração, acesso e controle de dispositivos de automação industrial. O protocolo de camada de aplicação Ethernet/IP baseia-se no CIP (*Common Industrial Protocol*) largamente usado quer no DeviceNet ou no ControlNet, e já bastante desenvolvidos a nível industrial. Construída sobre este protocolo a Ethernet/IP oferece um sistema integrado transparente desde o "chão-de-fábrica" até a rede corporativa (figura 3.3).

Devido ao uso de componentes ethernet generalistas, este protocolo torna-se numa solução com uma elevada relação qualidade/preço para a implementação de sistemas de automação. As redes implementadas sobre Ethernet/IP, são desenhadas para ter uma elevada largura de banda (10/100 Mbit/s). Esta elevada largura de banda associada com o uso de *switch* de rede, para criar domínios de colisão, o uso de *routers* e *firewalls* para separar os dados de controlo do chão-de-fábrica, dos dados associados aos níveis superiores de controlo e gestão (MES / ERP), permitem obter uma rede com uma elevada performance e fiabilidade na transmissão de dados. [7] [8] [9]

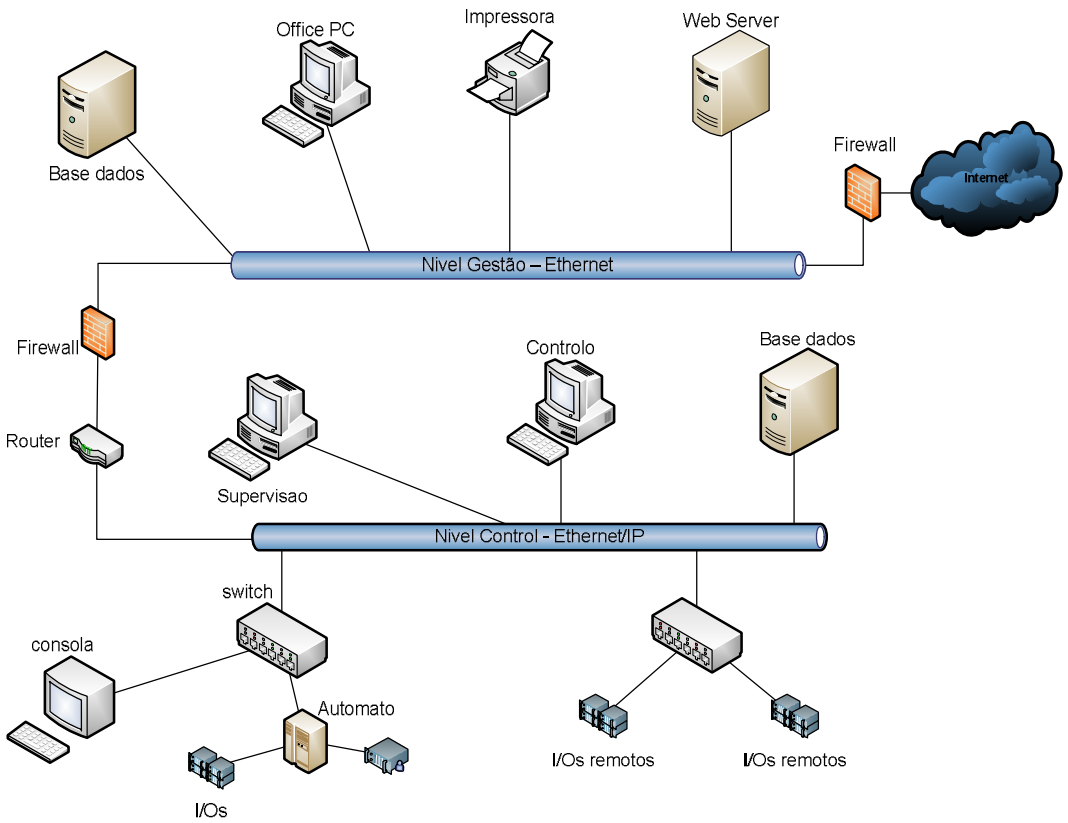


Figura 3.3 *- Exemplo simplificado da implementação de uma rede ethernet desde o chão-de-fábrica à rede corporativa*

Em termos de nível físico, as únicas diferenças entre este protocolo industrial e o protocolo geral, são a utilização de componentes (*routers*, *switch*) mais robustos, cabos blindados e fichas

com protecção, de modo a poderem resistir em ambientes industriais, onde poderão existir elevados níveis de ruído, poeiras, humidade, etc…

***Figura 3.4** - Exemplo de um switch industrial [10]*

***Figura 3.5** - Exemplo de fichas rj-45 para uso industrial [11]*

O problema com a implementação de sistemas de controlo com TCP/IP sobre ethernet é que esta não foi desenhada para operações que requerem elevada precisão temporal. Uma vez que a latência da rede aumenta com o aumento do tráfego na mesma, e o protocolo TCP/IP tem um elevado *overhead*. Outro problema é que qualquer dispositivo assente na rede pode fazer *broadcast*, o que pode levar a situações de falha de rede, e consequentemente a não entrega de outras mensagens. Uma solução para este problema é a utilização de tecnologias baseadas no IEEE 1588, que se baseiam na sincronização de relógio dos vários dispositivos da rede, tornando o envio de mensagens determinístico. Outras soluções têm como objectivo reduzir ao mínimo o *overhead* e utilizar algoritmos de roteamento baseados no endereço MAC. [12]

3.2.3 Common Industrial Protocol – CIP

Tradicionalmente as redes usadas em sistemas de automação estão optimizadas para obter o máximo de performance para determinada aplicação, normalmente para dispositivos, controlo, informação e segurança. Embora bem equipadas para as funcionalidades para as quais foram criadas, estas não foram desenvolvidas para integrar uma arquitectura corporativa. Uma vez que a eficiência, fiabilidade e a capacidade de gerar lucros estão normalmente dependentes de mais que uma destas características. Deste modo, os fabricantes viram-se obrigados a implementar varias redes diferentes, nenhumas das quais capaz de comunicar com a outra de forma nativa. Como resultado, a maior parte das redes corporativas são compostas por varias redes especializadas incapazes de comunicar entre si.

Actualmente, as especificações para a criação de redes de sistemas de automação, mudaram radicalmente, muito em parte, devido a rápida adopção de tecnologias baseadas na Internet. Companhias de todo o mundo, estão à procura das melhores maneiras de interligar os seus sistemas. A nova tendência da indústria é a de, aceder em qualquer hora e em qualquer local aos dados referentes a produção através de um sistema global de gestão da empresa.

Durante os últimos anos, houve um enorme crescimento na procura, por parte das companhias, de sistemas abertos capazes de interligar todos os processos da sua rede, desde o chão-de-fabrica até aos sistemas de gestão e planeamento. Contudo a expectativa de um sistema aberto foi uma desilusão. Os dispositivos, programas e processos usados nas várias camadas do modelo OSI tinham diferentes opções e não respeitavam qualquer standard. Deste modo, a interligação dessas redes iria necessitar de recursos e programação extra, e mesmo assim num era garantido que todas as lacunas existentes fossem cumpridas.

A chave principal para o avanço das comunicações e integração de redes é a utilização da camada de aplicação do modelo OSI. O CIP (*Common Industrial Protocol*) permite a completa integração dos sistemas de controlo, com os sistemas de gestão. Construído sobre uma única camada, este protocolo permite integrar o controlo de I/Os, a configuração de dispositivos e ainda a aquisição de dados através de múltiplas redes. Minimizando assim o custo em engenharia, instalação e tempo. [13]

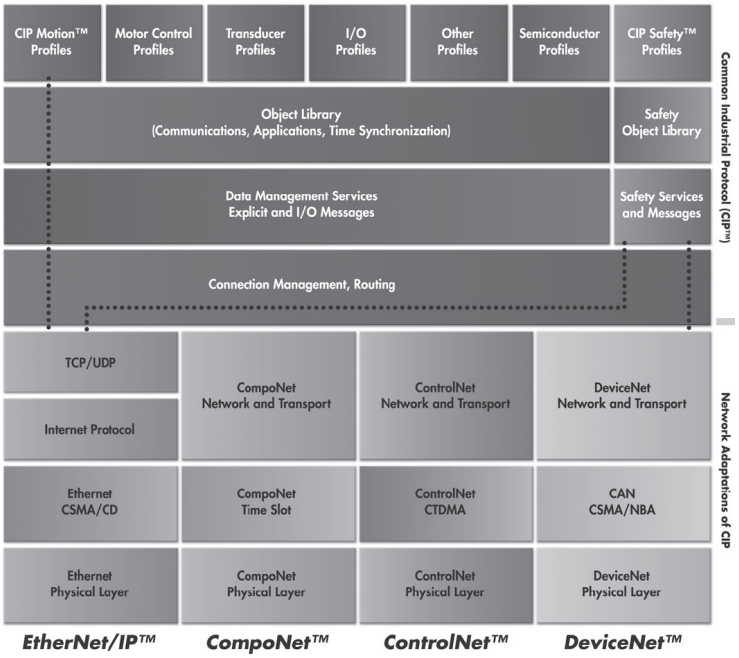


Figura 3.6 *- Implementação do CIP sobre vários protocolos de comunicação [13]*

De forma resumida, as grandes vantagens do CIP são:

- Possibilidade de implementação sobre vários meios físicos
- Funções especiais capazes de sincronizar várias aplicações e sistemas distribuídos.
- Funções dedicadas a segurança

3.3 SoftPLC's

A evolução dos computadores, nomeadamente em termos de velocidade de processamento e fiabilidade de funcionamento permitiu abrir um novo mercado na área de controlo, que até aqui era exclusiva dos PLCs.

Os SoftPlcs são aplicações de software que emulam o funcionamento de um PLC, com a capacidade de controlar IOs físicos reais. Estas aplicações são normalmente utilizadas em conjunto com sistemas operativos de tempo real, e permitem, para além de correrem as aplicações de controlo do processo, o uso de múltiplos protocolos de comunicação.

3.4 Tecnologias WEB

À medida que a *World Wide Web* se vai expandindo, a necessidade das empresas de expandir o seu negocio para este meio torna-se evidente, uma vez que permite às mesmas a interligação de todos os seus sistemas de negócios, mesmo que dispersos fisicamente, o acesso a dados e processos em qualquer parte do mundo, para além de permitir que potenciais clientes tomem contactos com os seus produtos.

A introdução de aplicações baseadas em tecnologias web face ao uso de software pré-instalado, é uma grande vantagem, uma vez que estas usam varias tecnologias standard (HTML, javascript, ajax, …), que permitem o desenvolvimento de aplicações para a maioria dos sistemas operativos existentes, sendo apenas necessária a utilização de um *web browser* para aceder e manipular a informação. Este tipo de tecnologias permite interagir com as aplicações de controlo e monitorização que estão correr em um determinado local, consultar bases de dados, etc. Sendo que todas estas funções são realizadas mediante pedidos a um servidor central, que se encarrega de aceder aos vários sistemas e recolher os dados relevantes. Outras vantagens do uso destas tecnologias são o facto de não ser necessário efectuar a actualização de todos os programas de acesso remoto existentes, quando se efectua alguma mudança no modo de funcionamento do sistema, sendo apenas necessário actualizar as aplicações existentes nos servidores. Existe também a vantagem de, como não é necessário proceder a instalação de um programa específico, evitar possíveis incompatibilidades com outros softwares existentes nos PCs ou mesmo incompatibilidades com o hardware dos mesmos.

Embora seja um sistema com enormes vantagens tem alguns inconvenientes, nomeadamente ao nível de segurança e performance, sendo que, o primeiro é o mais simples de ultrapassar mediante a encriptação dos dados transferidos e pela utilização de zonas de acesso restrito. O segundo inconveniente é o mais difícil de ultrapassar principalmente quando se esta ligado através da Internet, onde todas as comunicações são feitas utilizando um algoritmo de "melhor esforço" na entrega dos dados, não sendo deste modo garantidos os limites temporais.

3.4.1 CGI

O CGI do acrónimo em inglês “Common Gateway Interface” surgiu inicialmente com o intuito de estender as funcionalidades dos servidores. Permitindo gerar páginas instantaneamente quando requisitadas pelos utilizadores, ao invés de apenas carregar paginas estáticas previamente feitas.

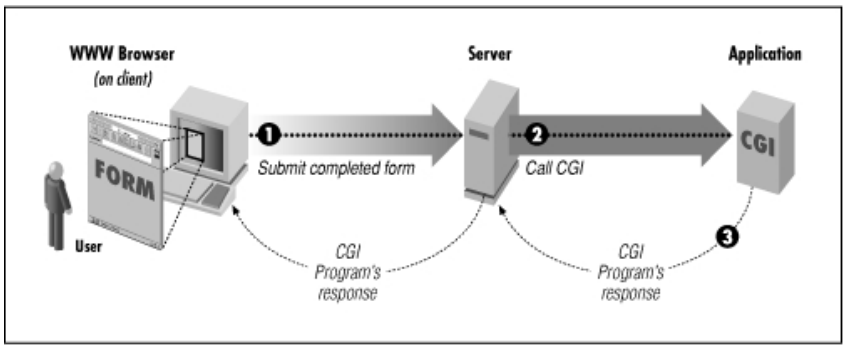


Figura 3.7 *- Diagrama de funcionamento do CGI [14]*

O CGI permite que seja executado programas que residem nos servidores a partir da *Web*. Deste modo, torna-se simples a interacção de utilizadores remotos com outras aplicações de existentes no servidor, que sejam bases de dados, sistemas de controlo de hardware, etc.

A utilização da linguagem C na programação dos *scripts* CGI, permite a criação de *scripts* de reduzido tamanho, mais optimizados em termos do uso de memória e de capacidade de processamento do servidor, ao contrario do uso de linguagem tipo PERL que necessitam de um maior poder computacional visto que tem que ser compilados e executados a cada pedido efectuado ao servidor. [15]

3.4.2 JavaScript

Durante muito tempo as pagina *web* foram criadas de forma estáticas, sendo que apenas eram usados os elementos HTML disponíveis, permitindo uma interacção mínima com os utilizadores. A necessidade das empresa criarem websites dinâmicos, interactivos e intuitivos capazes de oferecer ao utilizador uma boa experiência de navegação, e deste modo atraírem possíveis clientes para os seus produtos e/ou serviços, deram origem ao aparecimento do javascript que apesar do nome, não está relacionado com o java tradicional, apenas partilha alguma semelhança na sua sintaxe, sendo totalmente diferente no conceito e no uso.

O javascript é uma linguagem de programação interpretada, desenvolvida inicialmente pela Netscape em 1995, usada principalmente para o desenvolvimento de aplicações *Web* que são executadas pelo browser, do lado do cliente. O que significa que, qualquer código escrito em javascript é entregue juntamente com a página HTML e é executado dentro do browser do cliente em vez de correr directamente no servidor que disponibiliza a página.

Uma das mais frequentes utilizações do javascript é a de validação dos dados introduzidos por utilizadores em formulários. No caso de os dados não estarem correctamente inseridos, a submissão dos dados para o servidor não deverá prosseguir até que sejam correctamente inseridos. Esta acção é de grande importância, pois permite reduzir a carga nos servidores, passando a ser o *browser* o responsável pela validação dos dados e informação de erros de preenchimento. Deste modo o utilizador não tem que esperar que o servidor lhe valide os dados, tornando a experiência de navegação mais agradável e eficaz.

Porém existem algumas desvantagens no seu uso, das quais se destaca a adição de dezenas ou mesmo centenas de linhas de código a uma página *Web* tornando a sua manutenção extremamente complicada e de difícil compreensão. Esta desvantagem pode ser facilmente ultrapassada guardando as várias secções de código javascript num ficheiro com a extensão js e incluindo-o no cabeçalho de todas as páginas HTML que necessitem deste código. Desta maneira, o código javascript fica separado do conteúdo HTML da página propriamente dita. [16] [17]

3.4.3 AJAX

À medida que a *World Wide Web* se foi expandindo e a largura de banda das ligações aumentando, as pessoas tenderam a procurar *websites* com mais e melhores conteúdos, aplicações mais inteligentes e com respostas mais rápidas.

Até muito recentemente, à medida que a complexidade das aplicações *web* ia crescendo, os utilizadores começaram a deparar-se com sistemas lentos e com dificuldade de resposta, onde páginas inteiras tinham de ser recarregadas apenas para se actualizar um único elemento. A plataforma *Web* mostrava-se então frágil e pouco eficaz para as aplicações mais exigentes. De modo a ultrapassar estas dificuldades e possibilitar o desenvolvimento de páginas com melhor tempo de resposta, interactivas e dinâmicas, com uma maior eficácia, adoptou-se uma nova metodologia, o Ajax.

Este tornou-se popular em 2005 com o *Google Suggest*, o Ajax ou "*Asynchronous JavaScript and XML*" não é mais do que uma combinação de várias tecnologias standard já existentes como o javascript, o DHTML/HTML e o XML. O objecto XMLHttpRequest é o componente fundamental do Ajax e o que o torna tão útil. Ele proporciona um mecanismo para o cliente, através do javascript, trocar informação directamente com o servidor *Web*. A informação recebida pode ser processada em *background* e usada para dinamicamente actualizar elementos numa página *Web* sem a necessidade de recarregar toda a página. Desta forma, a informação circula em *background* e as páginas são actualizadas como se tratassem de aplicações *standalone* típicas, ao invés das aplicações *Web* tradicionalmente disponíveis até à altura.

A parte assíncrona do termo Ajax (figura 3.8) significa que o browser não vai esperar pela informação que vai ser devolvida pelo servidor, mas vai processá-la apenas quando esta for enviada pelo servidor. Esta é a parte crucial do Ajax, em que não temos de por colocar toda a aplicação *Web* à espera que os dados cheguem. Se a aplicação parar à espera da informação (aplicações tradicionais), essa aplicação será síncrona (figura 3.9). [18] [19]

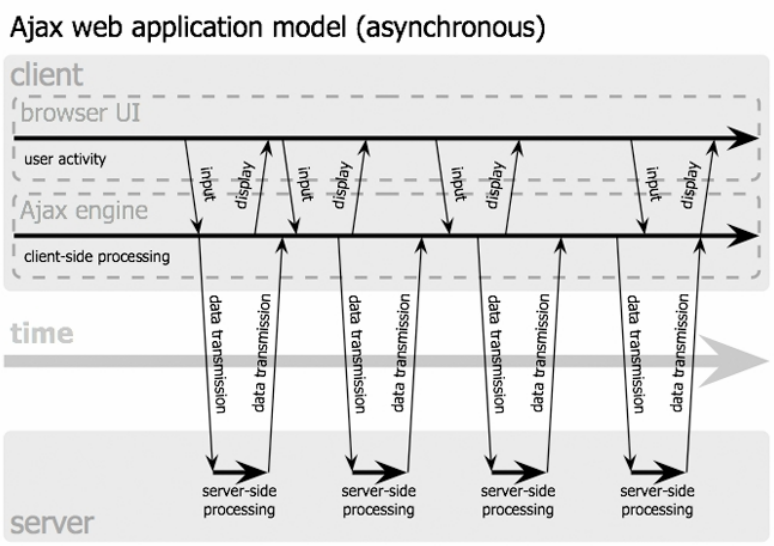


Figura 3.8 *- Modelo da interacção assíncrona das aplicações Ajax [19]*

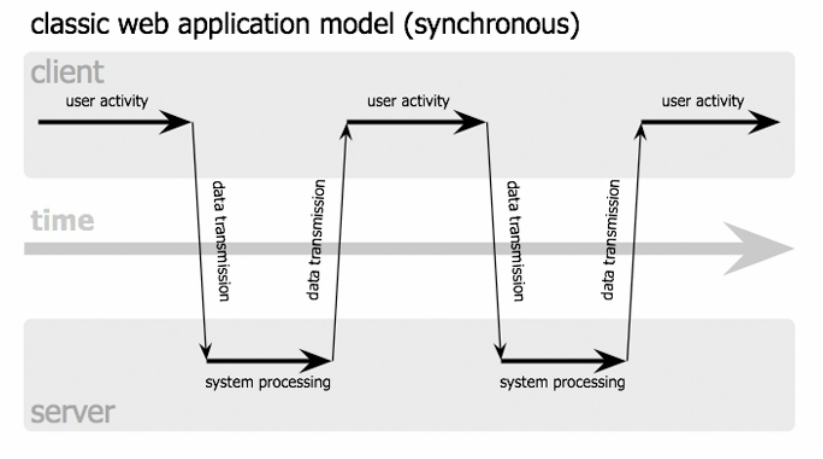


Figura 3.9 *- Modelo da interacção síncrona das aplicações Web tradicionais [19]*

3.4.4 PHP

Antes do aparecimento do PHP, a criação de páginas *Web* dinâmicas passava, principalmente, pela escrita de *scripts* CGI, o que significava escrever bastante código em C e compilar cada vez que se fazia uma alteração à aplicação, o que dependendo da dimensão e complexidade da aplicação poderia tornar-se uma tarefa bastante penosa em termos de tempo de desenvolvimento.

É devido a esta falha do CGI que surge o PHP, um acrónimo recursivo para "PHP: Hypertext Preprocessor". Esta nova tecnologia é uma linguagem de programação interpretada, orientada a objectos, livre e utilizada para gerar conteúdo dinâmico em páginas *Web* de uma maneira relativamente rápida. A sintaxe do código PHP é em tudo semelhante à do C/C++ e do JAVA e permite que as aplicações desenvolvidas façam tudo o uma aplicação CGI podia fazer até agora, tal como recolher dados de formulários e interagir com bases de dados. Os blocos de código PHP são embutidos directamente no código HTML e são executados no servidor quando este recebe o pedido de um cliente para carregar a página.

Quando alguém visita uma página concebida com esta tecnologia, primeiramente o servidor *Web* processa o código PHP, depois analisa quais as partes que deve mostrar aos visitantes (conteúdos, imagens) ocultando os blocos do código PHP depois de interpretados e executados. São nestes blocos onde se executam as operações como por exemplo, o acesso a bases de dados, cálculos numéricos e operações sobre ficheiros. No fim desta sequência, no servidor, o código resultante é traduzido para um ficheiro HTML que é a página que vai ser enviada para o *browser* cliente.

O PHP é considerado uma das linguagens mais fáceis de aprender e de utilizar para o desenvolvimento de aplicações *Web* dinâmicas. A simplicidade e facilidade de desenvolvimento do PHP, juntamente com uma grande comunidade e repositórios de bibliotecas PHP de código aberto, fazem desta linguagem a favorita de programadores e de *Web* designers a nível mundial. Para além de ser compatível com a maior parte dos sistemas operativos existentes para servidores *web*. [20] [21]

3.4.5 JQuery

JQuery é uma biblioteca JavaScript poderosa e extremamente leve que permite os programadores e os *Web* designers, adicionar elementos dinâmicos e interactivos às suas páginas, atenuar as inconsistências dos vários browsers e reduzindo fortemente o tempo de desenvolvimento das aplicações promovendo a produtividade.

Criado por *John Resig*, o jQuery é um projecto *open-source* que promove uma maneira diferente de escrever código em Javascript e que proporciona um instrumento essencial para fornecer uma compatibilidade multi-plataforma e simplificar a maneira de como se percorrem os vários elementos dos documentos HTML, a manipulação de eventos, o executar de animações e a simplificação das interacções Ajax com uma página *Web*. [22]

3.5 Sistemas Embebidos

Um sistema embebido é computador criado especificamente para realizar um pequeno conjunto de acções predeterminadas. Tem como principais características o baixo custo, requerem poucos recursos em termos de RAM, ROM ou outros dispositivos de I/Os em comparação com os necessários pelos PCs, normalmente correm sistemas operativos de tempo real, ou extensões de tempo real, e são tipicamente, dispositivos com reduzidos consumos em termos de energia.

De facto, nos primórdios dos sistemas embebidos, não era usado qualquer sistema operativo, estando o desenvolvimento do software para esses sistemas reservado as entidades fabricantes. Mas com o passar do tempo, tornou-se evidente a necessidade de utilização de um sistema operativo capaz de realizar múltiplas tarefas, múltiplos processos, gerir memoria, comunicações, etc. Desse modo, as empresas começaram a desenvolver módulos de software capazes de realizar as tarefas necessárias, aparecendo assim os primeiros sistemas operativos para sistemas embebidos. Actualmente existem dezenas de sistemas operativos embebidos, dos quais se destacam o Microsoft Windows CE e varias versões *baseadas em* Linux. [23] [24] [25]

3.5.1 Linux Embebido

A escolha de um sistema operativo embebido baseado em Linux, para além das vantagens de ser um sistema *open source*, está também abrangido pela licença GPL, logo é distribuído de forma gratuita, ao contrário das versões proprietárias, e embora existam varias empresas que desenvolvem sistemas operativos embebidos, todas usam os mesmos módulos básicos (Kernel) e livrarias, logo, com relativo baixo custo, é possível mudar de empresa fornecedora.

Os sistemas operativos embebidos baseados em Linux são sistemas monolíticos, isto é, existe uma clara distinção entre o *user space* e o *kernel space*, logo, quando um programa é executado a partir do *user space*, este normalmente não tem acesso ao hardware, tendo o programa de fazer chamadas a funções de sistema para aceder ao mesmo. Deste modo, qualquer falha no programa apenas irá afectar o comportamento do mesmo, deixando o resto do sistema intacto. [24]

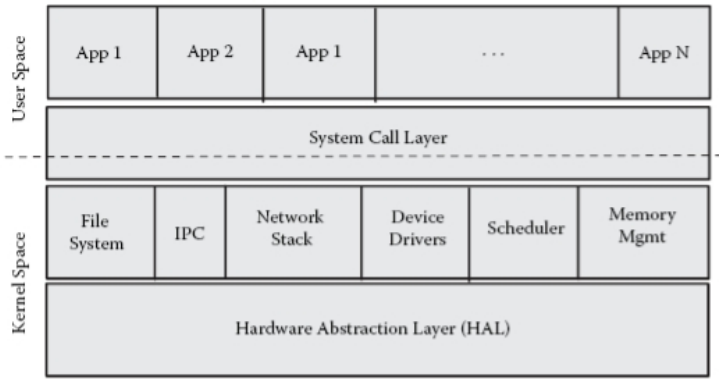


Figura 3.10 *- Arquitectura de um Kernel monolítico [23]*

3.5.2 Sistemas embebidos de tempo real

Os sistemas embebidos de tempo real são sistemas que reagem dinamicamente as mudanças de estado de determinadas variáveis. Nestes sistemas, o tempo é o principal factor, sendo que a dependência deste pode ser de importância critica (*hard real time*), no caso de sistemas que possam causar danos físicos ou ambientais, ou de reduzida dependência (*soft real time*), no caso de sistemas onde os custos relativos a falhas seja reduzido. [24]

Capítulo 4

4. Abordagem ao problema

Neste capítulo serão abordadas as várias soluções propostas para o sistema de controlo de simulação dos kits miniatura, bem como as várias soluções de controlo e configuração do sistema de simulação.

Para suportar o sistema de controlo de simulação foi escolhido a solução da InCircuit, a placa IcNova AP7000 Base. A escolha desta plataforma em detrimento da solução da Atmel deveu-se principalmente aos custos envolvidos na altura da compra, que eram bastante diferentes dos custos actuais (75€ para a placa ICNova e 135€ para a placa NGW100, preços em Fevereiro 2008), pesaram ainda as dimensões das mesmas, sendo que a placa ICNova é muito mais compacta e simétrica que a sua homologa, o que permite criar juntamente com os circuitos de isolamento uma solução mais simples e compacta.

4.1 Projecto Software

Para o projecto do sistema de controlo e simulação são propostas 2 arquitecturas de software distintas (arquitectura distribuída e arquitectura centralizada).

Em ambas as arquitecturas são utilizados servidores *Web* com suporte para *scripts* CGI, sendo que é através deste que se irá configurar e gerir todo o sistema de interface, nomeadamente configurar o sistema, configurar tipos de funcionamento para determinados pinos ou conjunto deles. Os *scripts* CGI comunicam com o programa principal (*Kit Supervision and simulation control*) efectuando as alterações necessárias ao funcionamento do sistema e lendo o estado actual do mesmo para posterior visualização pelos utilizadores.

4.1.1 Sistema de controlo distribuído

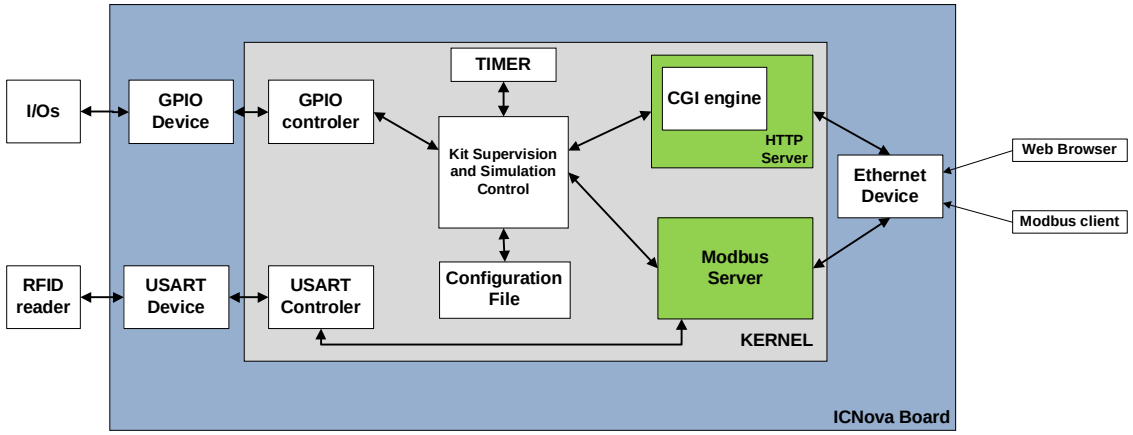


Figura 4.1 *- Arquitectura de controlo de simulação distribuída*

Nesta arquitectura, existem vários programas a correrem paralelamente, sendo que, cada um dos programas, *USART controler, GPIO controler*, tem como função efectuar a interface entre o programa principal (*Kit Supervision and simulation control*) e os respectivos drivers do Kernel responsáveis pelo acesso físico ao hardware, respectivamente *USART Device e GPIO Device*. Estes programas efectuam a ligação ao programa principal através de uma lógica de comunicação servidor cliente, sendo que, apenas comunicam com o programa principal quando existem dados relevantes para a correcta execução do mesmo.

Todos os dados requisitados via web browser, serão processados pelos scripts CGI e posteriormente enviados para o programa principal, o qual irá alterar as variáveis de funcionamento de acordo com os dados recebidos.

No caso das comunicações via USART, e como estas não alteram qualquer parâmetro no programa principal (apenas interessam para os níveis de gestão de produção), são encaminhadas directamente para o servidor modbus para este poder enviar as mesmas para os sistemas de gestão de dados.

O programa principal será o responsável pela gestão dos eventos recebidos, tendo este acesso aos timers do sistema e acesso aos ficheiros de configuração, onde serão guardadas variáveis de funcionamento e de simulação.

4.1.2 Sistema de controlo Centralizado

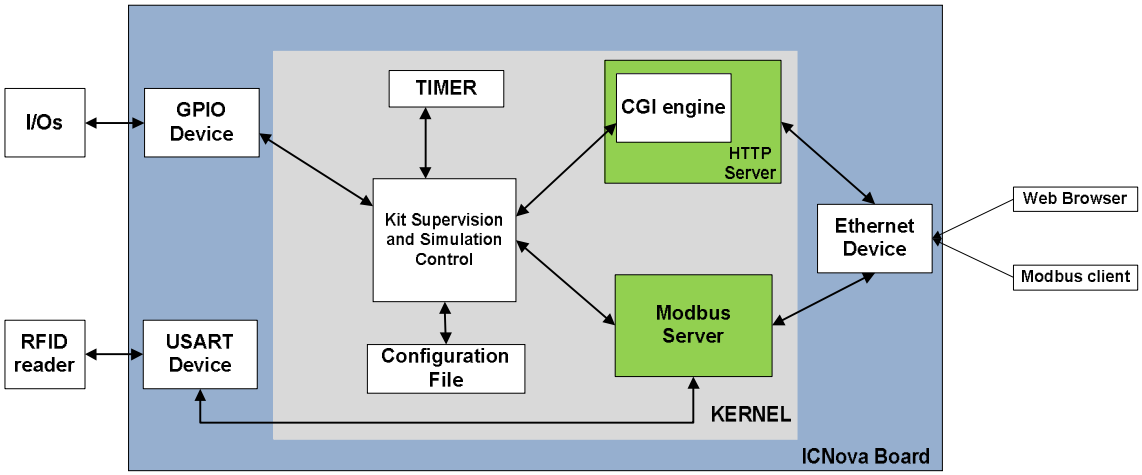


Figura 4.2 *- Arquitectura de controlo de simulação centralizada*

Nesta arquitectura de software apresentada, o programa principal (*Kit Supervision and simulation control*) é o responsável pela execução completa do sistema de simulação e controlo. Para efectuar esse controlo, existem duas hipóteses de implementação do sistema de gestão dos GPIOS, por *polling* ou por eventos (interrupções).

Para o primeiro caso (*polling*), o programa principal irá verificar de forma cíclica se existem alterações nos estados dos pinos de GPIO e comparar com os eventos previamente configurados via CGI.

No segundo caso (eventos), o programa principal receberá uma notificação a avisar a mudança de estado do pino GPIO, após a qual irá ser executada a rotina de atendimento de interrupções configurada para esse tipo de eventos, e dentro desta verificado quais as acções a realizar.

Tal como na arquitectura anterior, servidor modbus comunica directamente com o dispositivo USART para envio e recepção de dados para as etiquetas RFID.

4.2 Projecto Hardware

Devido as diferenças dos níveis de tensão entre o sistema de controlo de simulação e os vários sistemas utilizados para a automação de processos, torna-se necessário a criação de um sistema de interface entre estes dois sistemas.

A figura 4.3 demonstra o esquema funcional requerido para o sistema, o qual terá de ser capaz de observar, receber e enviar sinais lógicos para qualquer um dos sistemas existentes, nomeadamente interceptar e actuar nas várias entradas e saídas dos mesmos.

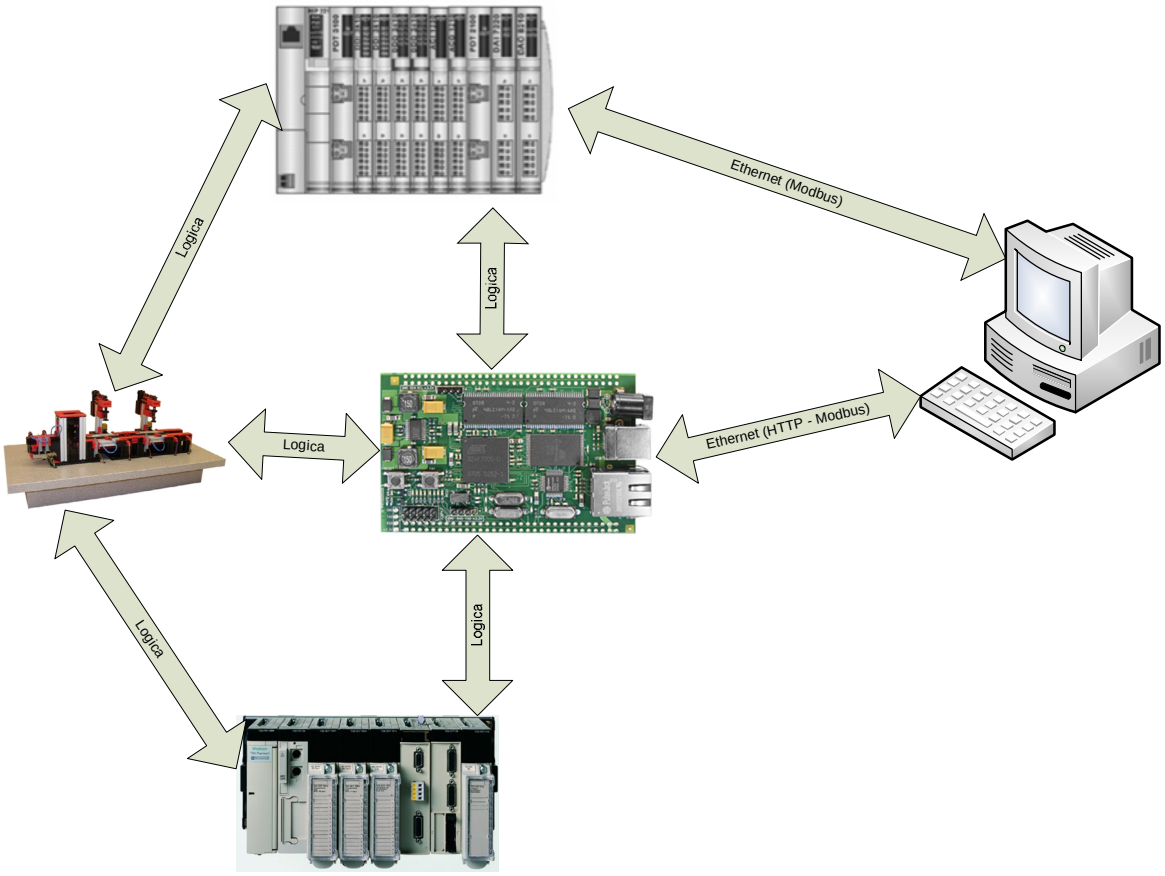


Figura 4.3 *- Esquema funcional para o sistema proposto*

4.2.1 Requisitos do sistema de interface

O sistema de interface deverá ter as seguintes características:

- Efectuar o isolamento quer ao nível de sinais, quer ao nível de alimentações e massas.
- 40 Entradas directas para o sistema de controlo de simulação.
- 48 Saídas.
- Dimensões reduzidas (não superiores a 10x15 cm)

4.2.2 Características dos vários sistemas

As tabelas 4.1 e 4.2 resumem as características eléctricas de cada um dos sistemas.

Tabela 4.1 - Características eléctricas dos kits

	Alimentação [V]	***Sinais [V]***	***Corrente [mA]***
Saídas	24	0 ou 24	---
Entradas	24	24	20
RS-232	12	12	---

Tabela 4.2 - Características eléctricas do microprocessador

	Alimentação [V]	***Sinais [V]***	***Corrente max[mA]***
Saídas	3,3	0 ou 3,3	9
Entradas	3,3	0 ou 3,3	9
RS-232	3,3	3,3	---

Características físicas da placa de controlo

- 55 Pinos de IOs genéricos (GPIOs).
- 2 portas USART
- 10/100MBit/s Ethernet
- Alimentação via USB ou através do conector de alimentação.
- Corrente máxima do conversor: 2A

4.2.3 Dimensionamento dos componentes de interface e isolamento

Devido aos níveis de tensão que tipicamente existem nos sistemas de automação (24V), e os níveis de tensão a que o microcontrolador funciona (3,3V), torna-se necessária a criação de um circuito que permita reduzir os níveis de tensão e que, permita ainda, isolar electricamente ambas as partes do sistema.

4.2.3.1 Isolamento Entradas

No caso das entradas, e tendo em conta o requisito de que, todas têm que estar ligadas directamente a placa de controlo de simulação (deste modo o SO da placa é capaz de detectar os eventos nos pinos sem ter que estar a gastar tempo de ciclo a verificar varias latch), serão necessários 40 I/Os. Para efectuar o isolamento entre os diferentes níveis de tensão utilizados, optou-se pelo uso de optoacopladores, de modo a garantir um completo isolamento eléctrico entre as partes.

O optoacoplador escolhido foi o HCPL-2730 com as seguintes características:

- 2 Canais
- Elevado CTR ≈ 2000%
- Baixa corrente de entrada
- Saída de -0,5V a 7V
- V_F =1,3V @ 1,6mA

Para garantir que o optoacoplador entra em funcionamento quando lhe é aplicado um sinal de 24V a entrada, é necessário dimensionar as resistências R1 e R2 (figura 4.4) de modo a obter a corrente requerida pelo diodo para entrar em condução.

$$R1 = R2 = \frac{V_{cc} - V_F}{I_F} = \frac{24 - 1{,}3}{1{,}6mA} = 14K2\,\Omega$$

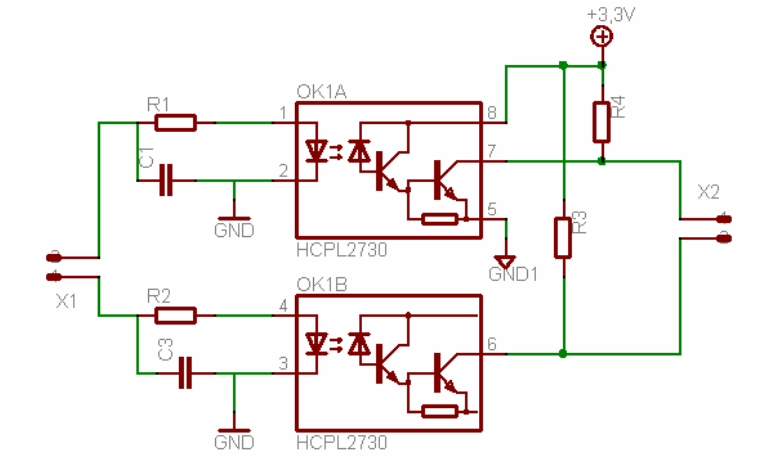


Figura 4.4 *- Esquema de ligações para o HCPL2730*

A alimentação do lado dos 24V é garantida pelos próprios actuadores do PLC/Kit. Do lado do sistema de controlo de simulação, a alimentação é garantida pelo conversor integrado na própria placa, que por sua vez esta ligado ou via USB a um PC ou a uma fonte de alimentação isolada da anterior.

Depois de dimensionado os componentes para um dos optoacopladores, torna-se necessário replicar o mesmo, de modo a obter as 40 entradas directas para o sistema, devidamente isoladas. Sendo que o total de custos associados a esta montagem é de 77,16€. Na tabela 4.3 estão descriminados as quantidades e preços unitários de cada um dos componentes necessários.

Tabela 4.3 - Custo dos componentes do circuito de isolamento de entradas

Componente	***Quantidade***	***Preço unitário [€]***	***Total [€]***
HCPL-2730	20	1,48	29,6
Conector Wago 8	5	3,16	15,8
Conector Wago 2	1	0,84	0,84
Conector PC104	2	13,62	27,24
Resistência	80	0,005	0,4
Condensador	40	0,082	3,28

Este conjunto de componentes é uma escolha equilibrada, uma vez que se consegue obter o isolamento eléctrico das entradas com um relativo baixo custo e respeitando as dimensões máximas permitidas. Com esta solução é possível colocar todas as 40 entradas em uma única placa de circuito impresso de reduzidas dimensões. O esquema mecânico referente a placa de entrada projectada encontra-se no Anexo 2.

4.2.3.2 Isolamento Saídas

Para o isolamento das saídas, e tendo em conta os requisitos para o sistema de interface (40 entradas + 48 saídas) e a quantidade de I/Os disponíveis na placa *Icnova* (55 GPIOs), verifica-se que não existem I/Os suficiente para cobrir os requisitos previamente indicados.

Deste modo, torna-se evidente a necessidade de multiplicar o número de I/Os disponíveis. Uma vez que as entradas do sistema têm que ser directas, sobram as saídas para serem multiplexadas. Assim, para efectuar a multiplexação podem ser usados *flip-flop* de 8 bits, permitindo deste modo, obter as 48 saídas requeridas com apenas 6 *flip-flop* e utilizando apenas 14 linhas de I/O da placa *Icnova* (8 linhas de dados mais 6 linhas para selecção dos integrados). Para além da multiplexação das saídas, é também necessário efectuar o isolamento eléctrico entre ambas as partes. Para efectuar as funções referidas, poderão ser utilizados os seguintes componentes:

◦ Octal Flip-Flop 74ACT574 com as seguintes características:

- 8 Entradas/Saídas
- Entradas e saídas em lados opostos do chip para facilitar o layout
- Compatível com os níveis lógicos requeridos (3,3V)
- 24 mA por saída

◦ Optoacoplador HCPL3020 com as seguintes características:

- Até 200mA corrente de saída
- I_F = 8mA
- V_F = 1,5V
- Saída entre 10V e 30V

De modo a garantir que o optoacoplador entra em funcionamento quando lhe é aplicada um sinal de 3,3V, será necessário dimensionar a resistência R1 (figura 4.5).

$$R1 = \frac{V_{cc} - V_F}{I_F} = \frac{3{,}3 - 1{,}5}{8mA} = 225\Omega$$

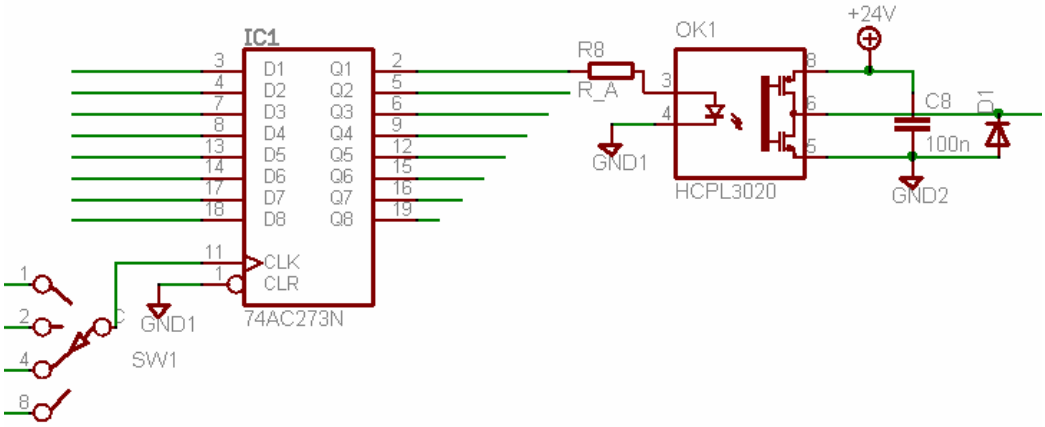


Figura 4.5 *- Esquema de ligações para o HCPL3020 e 74ACT574*

Note-se que na saída do optoacoplador é ligado um diodo de roda livre, para evitar sobretensões nos terminais do mesmo, quando este é desligado.

O custo associado a esta montagem é de 189,14€ no total. Na tabela 4.4 estão descriminados as quantidades e preços unitários de cada um dos componentes necessários.

Tabela 4.4 - Custo dos componentes do circuito de isolamento de saída a optoacopladores simples

Componente	***Quantidade***	***Preço unitário [€]***	***Total [€]***
HCPL-3020	48	1,25	60
Conector Wago 8	6	3,16	18,96
Conector Wago 2	1	0,84	0,84
Conector PC104	6	13,62	81,72
Resistência	48	0,005	0,24
Diodo	48	0,155	6,2
“Rotary switch”	6	2,65	15,9
74ACT574	6	0,88	5,28

Este conjunto de componentes apresenta uma relação custo/complexidade reduzida, uma vez que torna possível o isolamento eléctrico das saídas com um mínimo de componentes necessários, e garante ainda que, as saídas estão efectivamente ligadas ao nível lógico requerido, uma vez que o optoacoplador escolhido garante a tensão Vcc na saída quando existe sinal de entrada e garante a ligação a massa quando se desliga o sinal de entrada.

Outra solução para as saídas passa pela utilização de optoacopladores para a efectuar o isolamento entre a placa de controlo e o PLC/kit e posteriormente o uso de transístores para fornecer a corrente necessária para alimentar os componentes do kit. Para isso poderão ser usados os seguintes componentes:

◦ Optoacoplador 4N35 com as seguintes características:

- CTR = 100%
- V_F =1,3V @ 8mA
- Saída até 50V
- I_{Cmax} = 50mA

◦ Array de transístores ULN2801A com as seguintes características:

- 8 Canais
- Corrente saída até 500mA
- Tensão saída até 50V
- Diodo de roda livre por canal

A montagem referente apenas um sinal de saída é a apresentada na figura 4.6.

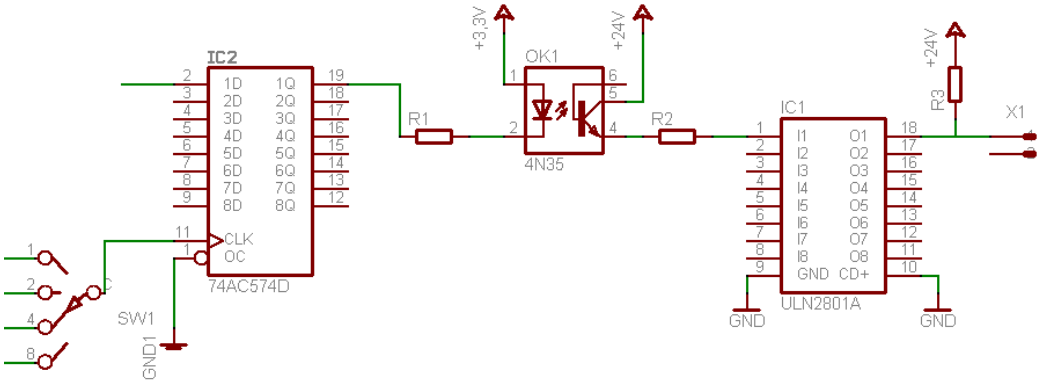


Figura 4.6 *- Esquema de ligações para o ULN2801A e 74ACT574*

Para o correcto funcionamento dos vários integrados é necessário dimensionar as resistências R1, R2 e R3.

$$R1 = \frac{V_{cc} - V_F}{I_F} = \frac{3{,}3 - 1{,}3}{8mA} = 225\Omega$$

$$R2 = \frac{V_{cc}}{I_B} = \frac{24}{0{,}85mA} = 28K4\Omega$$

$$R3 = \frac{V_{cc}}{I_c} = \frac{24}{120mA} = 200\Omega$$

O custo associado a esta montagem é de 150,66€ no total. Na tabela 4.5 estão descriminados as quantidades e preços unitários de cada um dos componentes necessários.

Tabela 4.5 - Custo dos componentes do circuito de isolamento de saída a transístores

Componente	***Quantidade***	***Preço unitário [€]***	***Total [€]***
4N35	48	0,43	20,64
Conector Wago 8	6	3,16	18,96
Conector Wago 2	1	0,84	0,84
Conector PC104	6	13,62	81,72
Resistência	144	0,005	0,72
“Rotary switch”	6	2,65	15,9
74ACT574	6	0,88	5,28
ULN2801A	6	1,10	6,60

Este conjunto de componentes representa a melhor relação custo/funcionalidade, uma vez que efectua a função requerida, a de isolar electricamente as saídas, com o menor custo. Mas aumenta a complexidade e numero de componentes a colocar na placa de circuito impresso.

Em ambas as soluções propostas, as alimentações das saídas da placa de interface, serão asseguradas por uma fonte de 24V ligada a um conector existente. Para a alimentação do circuito dos *flip-flop* poder-se-á usar uma alimentação externa de 3,3V devidamente isolada da fonte de 24V, ou usar a alimentação fornecida pelo conversor integrado na placa.

Devido ao elevado numero de saídas e ao facto de para cada saída ser necessário um optoacoplador, o que totaliza 48 integrados, optou-se por dividir essas mesmas saídas por 3 placas de circuito impresso, cada uma com 16 saídas, ligadas da forma apresentada na figura 4.7. A transmissão dos sinais entre os vários andares de placas de circuito impresso será feito através de um barramento tipo PC104.

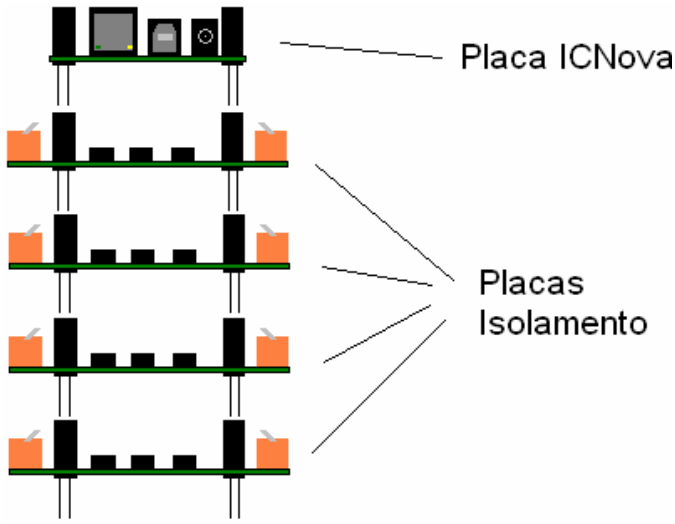


***Figura 4.7** - Esquema representativo da solução de isolamento proposta*

Assim sendo, optou-se por desenhar 3 placas de saídas genéricas (exactamente iguais), sendo necessário para isso a utilização de um método que permita seleccionar qual dos vários sinais de controlo disponíveis é que será o responsável pela activação de determinado *flip-flop*. Para isso utilizou-se um "*rotary switch*" de 1 pólo para 6 posições, ligado às 6 linhas de selecção vindas da placa ICNova e com o pólo ligado ao clk do *flip-flop*. Por actuação no "*rotary switch*" torna-se possível escolher qual a linha de selecção é que será associada a determinado *flip-flop* e consequentemente qual a linha de selecção responsável por determinado conjunto de saídas.

Esta solução de dividir as saídas por varias placas permite que o conjunto final, composto por 1x placa de entradas, 3x placas de saídas e 1x placa ICNova, fique com umas dimensões reduzidas (AxCxL=10x10x8cm), tornando a sua manipulação fácil e sendo possível a sua colocação por baixo do suporte dos kits.

Conclusão:

Após a análise dos vários prós e contras das soluções existentes para o circuito de saídas optou-se por usar o esquema de saídas a optoacopladores (Figura 4.5), uma vez que deste modo é possível manter as dimensões da placa no mínimo possível, e mantêm a complexidade das mesmas a níveis aceitáveis e garante ainda a correcta aplicação dos sinais nas saídas.

Note-se ainda que a razão de o custo do isolamento das saídas ser superior das entradas, deve-se ao facto de ser necessário dividir as saídas por várias placas (3x), o que se traduz em uma necessidade de utilizar vários componentes extra, nomeadamente os conectores PC104 e os "Rotary switch", que trazem um custo acrescido as soluções.

O custo total associado a esta solução é de 361,3€, estando incluído 1 placa de isolamento de entradas, 3 placas de isolamento de saídas e 1 placa ICNova AP7000.

O esquema mecânico da placa de saídas desenvolvida encontra-se no Anexo 2.

Capítulo 5

5. Implementação e Teste

Depois de estudadas as várias soluções, para o controlo e supervisão, optou-se pela arquitectura descrita na figura 5.1.

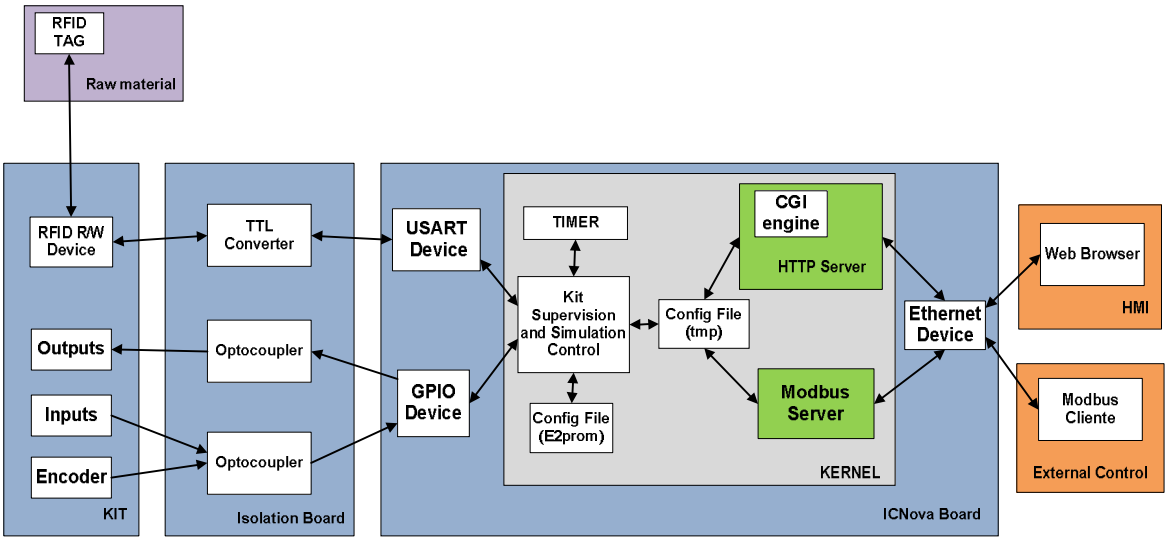


Figura 5.1 *- Esquema geral da solução proposta*

Em detalhe, o sistema proposto é composto por vários kits com as características referidas no capítulo 2, uma placa de isolamento, para efectuar o nivelamento de tensões entre o kit e/ou PLC/IOs remotos e a placa ICnova, que é a responsável pelo controlo de simulação. Esta placa tem um sistema operativo baseado num *kernel* de Linux, e tem para além dos vários sistemas inerentes a um sistema operativo tem um servidor *web* integrado com suporte para CGI, um driver para a gestão dos IOs e outro driver para a gestão das USART. Foi também implementado um servidor de modbus para envio e recepção de dados referentes aos *encoders*.

Na implementação do sistema de controlo de simulação para integrar na placa ICnova, optou-se pela arquitectura centralizada, onde um único processo é responsável pela gestão de todo o sistema, o "Kit supervision and control".

5.1 Implementação do software de controlo de simulação

Para a criação das várias rotinas foi necessário recorrer a um *Cross-compiler*, uma vez que as aplicações estavam a ser desenvolvidas numa plataforma computacional (X86) diferente da plataforma em causa (ARM) que, para além de uma arquitectura diferente, dispõem de recursos de hardware mais limitativos, nomeadamente ao nível de memoria não volátil para armazenamento da aplicação. Por outro lado a existência de um sistema operativo proporciona funcionalidades para multitarefas, gestão de memória e permite que ao programador desenvolver as suas aplicações com independência do hardware subjacente. [29]

5.1.1 Rotina de controlo

Para efectuar o controlo do sistema de simulação foi criado um único programa que é o responsável pela gestão das varias funções, tais como obter configurações de funcionamento a partir dos scripts CGI, gestão de modos de funcionamento, tempos de atraso de sinais, efectuar a contagem de voltas dos *encoders* (com detecção de direcção) e envio das mesmas para um cliente modbus.

O processo de controlo de simulação segue o diagrama de sequência apresentado na figura 5.2. Neste processo, depois de iniciados os vários sistemas (registo dos GPIOS, leitura de dados e configurações iniciais) com sucesso, é activado o *timer* que será o responsável por ciclicamente gerar uma interrupção onde será verificado, se existe novos pedidos feitos pelo CGI e incrementar os temporizadores registados. Noutra zona do programa, que está em constante verificação, são comparadas as definições dos eventos registados com os estados dos pinos. No caso de existir um evento associado a determinado pino, são iniciados os temporizadores associados, nos quais ao fim do tempo requerido irão activar a respectiva saída. Note-se que caso não exista qualquer evento registado para determinado pino de entrada e ocorra uma mudança de estado do mesmo, a saída associada será actuada de acordo com o estado actual (quando o pino de entrada é activado, o correspondente pino de saída também é activado e vice-versa).

Esta aplicação também é usada para a interface com os *encoders*, sendo o ciclo de verificação explicado na seguinte sub secção (5.1.2).

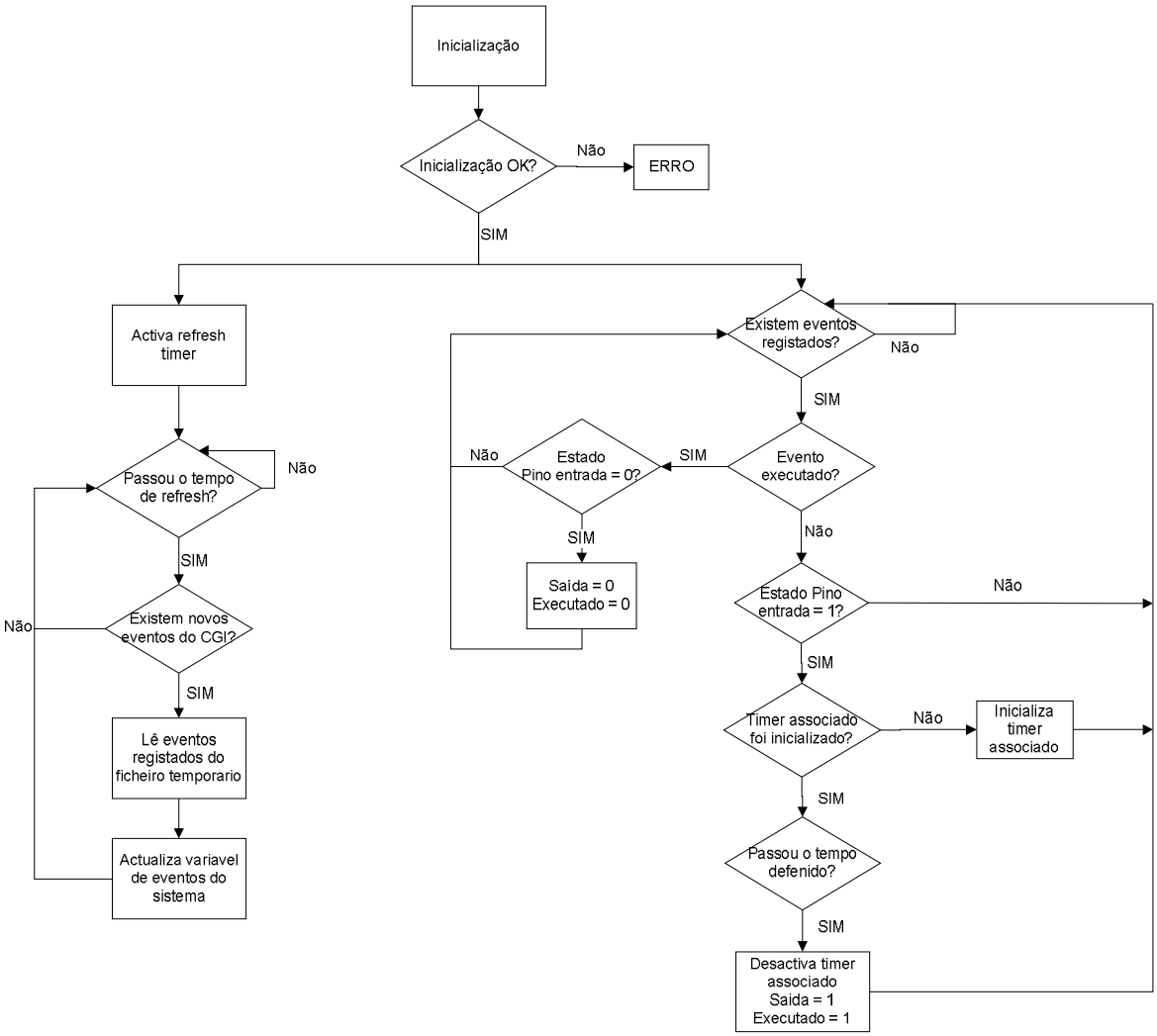


***Figura 5.2** - Diagrama de sequência do processo de controlo de simulação*

Durante a execução deste algoritmo de controlo, podem ser requisitados novos eventos para os pinos através de uma página *web*. Podendo os pinos ser configurados de 3 modos distintos:

- **Modo observação** – em que os pinos apenas podem ser lidos, este modo funciona na observação de qualquer sinal quer seja I/O do PLC ou do kit.

- **Modo *encoder*** – Para a contagem de impulsos vindos de *encoders*, onde é necessário alocar 2 pinos contíguos para efectuar a contagem.

- **Modo de interface** – Podem ser atribuídas varias funções a conjuntos de pinos, por exemplo, usar um par de pinos (uma entrada e uma saída) para gerar atrasos de propagação de sinais. Dentro deste modo poderão haver vários outros sub modos de funcionamento:

  - Normal – em que o atraso dos sinais pode ser:

    - Dado por uma formula (constante, gauss, exponencial, …);

    - Dependentes do numero de pinos activos ou inactivos;

- Dado por um evento num dos pinos.

- Avaria – onde se pretende que os sinais sejam:

  - Atrasados de forma aleatória (*random*) ou mediante uma formula não linear (passa-alto ou passa-baixo);

  - Mantidos sempre ao nível lógico baixo ou alto, mesmo quando ocorrem alterações de estado dos pinos.

Todos estes modos de funcionamento referentes aos pinos são configuráveis via página *web*.

5.1.2 Rotina de aquisição para encoders

Para a aquisição de sinais vindos dos *encoders*, nomeadamente, o número de voltas e o sentido de rotação, foi necessário estudar a máquina de estados associada ao funcionamento de um *encoder* (Figura 5.3) para poder desenvolver um algoritmo capaz de realizar as funções requeridas.

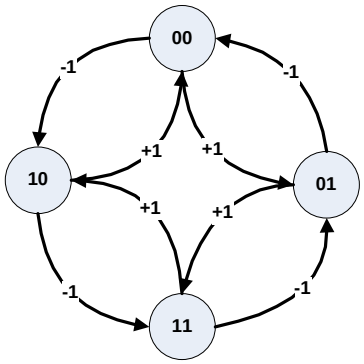


Figura 5.3 *- Máquina de estados do funcionamento de um encoder incremental*

Depois de estudado o funcionamento do *encoder* para as várias situações, foi criado um algoritmo que segue o diagrama de sequência descrito na figura 5.4.

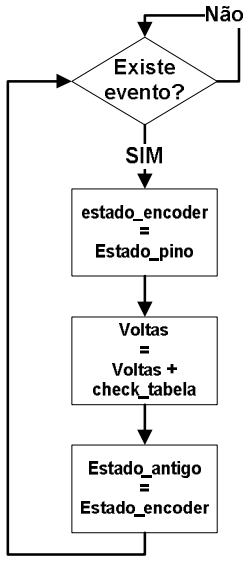


Figura 5.4 *- Diagrama de sequencia do algoritmo associado ao encoder*

Este algoritmo verifica se existe algum *encoder* registado no sistema, e no caso afirmativo, vai ver quais os pinos associados e guarda os seus estados como estado_encoder, de seguida, vai comparar o estado actual com o estado antigo e verificar numa tabela se tem que incrementar ou decrementar o numero de voltas. Finalmente guarda o estado actual como o estado antigo para a próxima iteração.

5.1.3 Paginas *web* para configuração do sistema

Uma vez que a placa usada tem um servidor *web* integrado com suporte para HTTP e CGI, usou-se esta funcionalidade a criação de uma interface de configuração do sistema de controlo de simulação. Foram criadas varias páginas com os vários modos de configuração existentes para o sistema. Tendo sido utilizada a biblioteca jQuery para reduzir a complexidade de implementação e manter a compatibilidade entre browsers das chamas assíncronas causadas pelas escolhas feitas pelo utilizador.

Nesta página (Figura 5.5) é possível escolher um dos três modos de funcionamento dos pinos. Ao efectuar a escolha, é carregado o formulário referente a este modo.

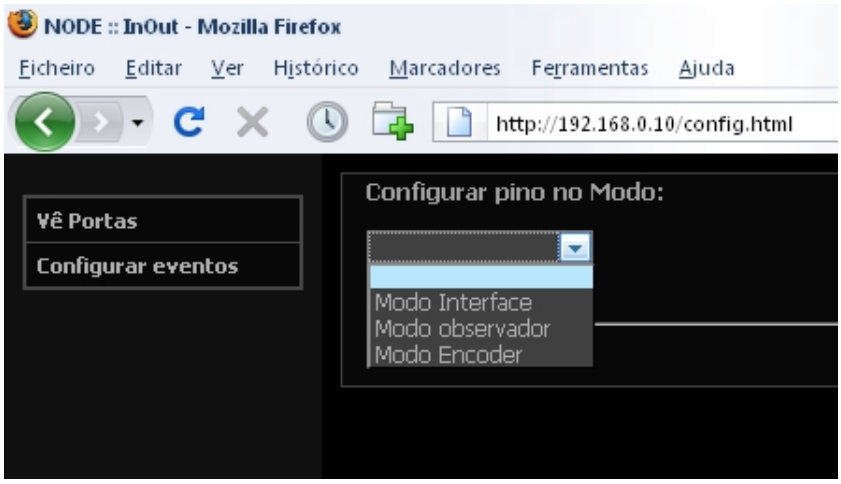


Figura 5.5 *- Pagina Web para escolha do modo de funcionamento*

Uma vez escolhido o modo de funcionamento é possível configurar varias funções (Figura5.6) para associar entre os pinos de entrada e de saída.

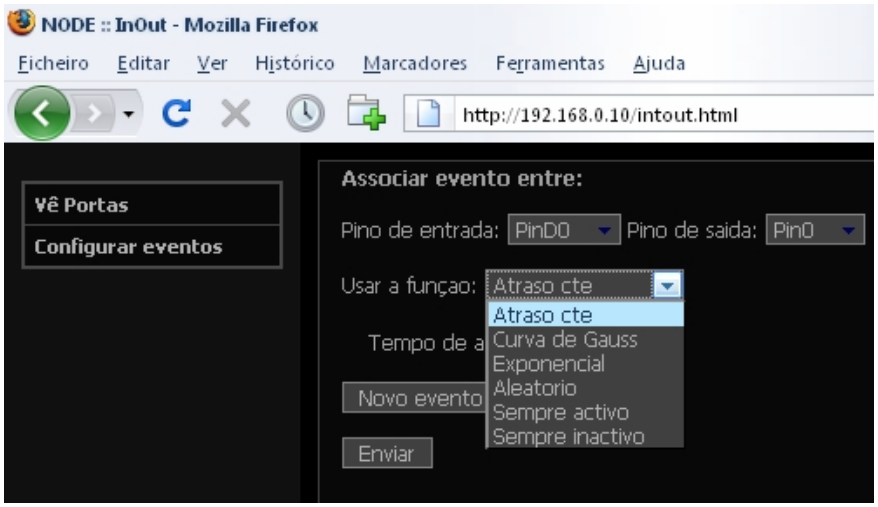


Figura 5.6 *- Pagina de configuração de eventos*

Depois de escolhidas as configurações para os pinos, o script CGI irá escrever para um ficheiro de texto temporário as configurações para o evento acabado de registar. Este ficheiro temporário será depois lido pelo programa "Kit supervision and control" que irá actualizar a sua lista de eventos e reagir aquando do acontecimento do evento associado.

Através da escolha do modo de observador, ou recorrendo ao menu lateral, é possível visualizar o estado das varias portas, quer estas sejam entradas (Port A) ou saídas (PortC Port D). A página apresentada na figura 5.7 mostra essa funcionalidade. Quando a imagem associada ao pino é verde, quer dizer que este está ao nível lógico 0, o que representa, no caso das saídas que a mesma esta desactivada. No caso da entradas a imagem verde indica que o pino associado esta a ser actuado externamente.

Nesta página podemos escolher a frequência com que a página será actualizada. É também possível activar ou desactivar as saídas "clicando" nas imagens referentes. Todas estas funções de actualização e mudança dos estados dos pinos de saída são realizadas recorrendo a pedidos assíncronos ao servidor via XMLHttpRequest. Deste modo, a página mantém-se funcional mesmo que o servidor demore mais tempo a efectuar a acção requisitada, sendo possível realizar outras funções.

Figura 5.7 *- Pagina web do modo de observador*

5.2 Implementação do software de controlo do kit

Para a validação de resultados, foi implementado um algoritmo de controlo com base na norma IEC 61131-3, capaz de controlar uma das partes do kit (Figura 5.8). Este algoritmo foi realizado, recorrendo ao ambiente de desenvolvimento Isagraf da companhia *ICS Triplex ISaGRAF Inc*, que permite a criação de algoritmos de controlo de acordo com a norma atrás referida. Esta companhia disponibiliza ainda uma aplicação de *Runtime*, que permite emular o funcionamento de um PLC, tornando possível deste modo interagir directamente com o hardware a controlar. Esta interacção é feita através da utilização de um driver de modbus/TCP, o qual permite mapear variáveis remotas em variáveis do próprio sistema, tornado o acesso a essas mesmas variáveis transparente para o programador. Para além destes sistemas, foi ainda utilizada uma ilha de IOs remotos da Schneider Electric, o Advantys STB, que comunica com o sistema de controlo via modbus/TCP, e é o responsável pela leitura dos dados e actuação nas variáveis físicas do processo.

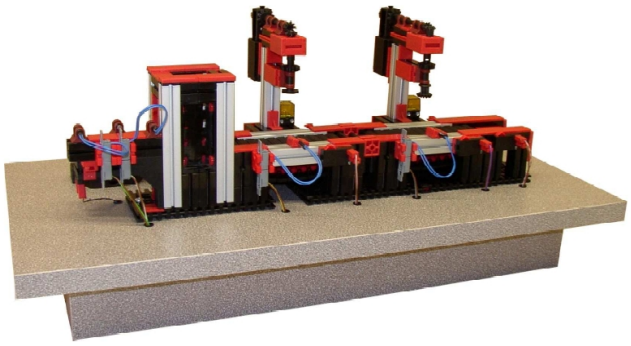


Figura 5.8 *- Kit usado na validação da solução*

O kit utilizado para testes, figura 5.8 é um módulo de maquinagem sequencial (duas maquinas) e um armazém de peças por maquinar. Para retirar as peças do armazém existe um braço extensível que coloca uma peça de cada vez no tapete associado a primeira máquina. Quando a peça chega a posição da máquina 1 é realizada a operação, que, quando terminada é colocada a peça em espera junto do tapete associado a maquina 2. Quando a maquina 2 estiver livre, inicia-se o processo de posicionamento da peça para junto da máquina 2, onde será realizada a operação associada. Terminada a operação, a peça e enviada para fora do sistema, terminando assim o ciclo de funcionamento.

No ambiente de desenvolvimento, foi criado um algoritmo de controlo utilizando as linguagens SFC (“*Sequential Function Chart”*) e ST (“*Structured Text”*). O algoritmo criado foi dividido em 3 módulos independentes, armazém, maquina 1 e maquina2, de modo a permitir o funcionamento independente dos mesmos, quando existem 2 sistemas contíguos livres e existem peças para processar.

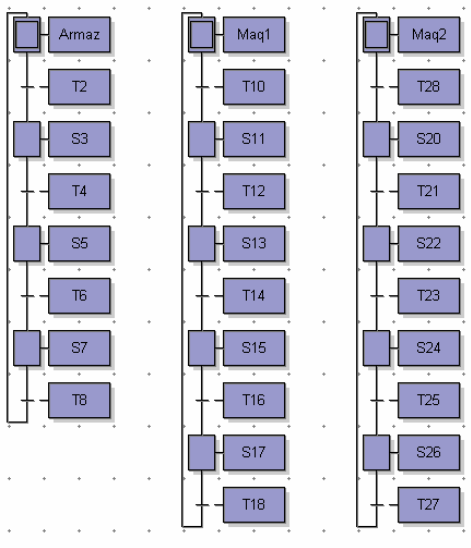


Figura 5.9 *- SFC do software de controlo do kit*

Para verificar se a aplicação de controlo funcionava correctamente, ligou-se o kit directamente a ilha STB e iniciou-se o *Runtime* com a aplicação desenvolvida. O esquema usado é o apresentado na figura 5.10.

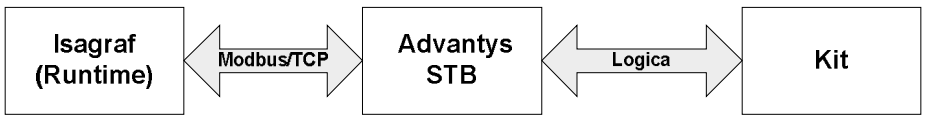


Figura 5.10 *- Esquema implementado para de teste da aplicação de controlo*

Deste modo pode-se verificar que a aplicação de controlo do kit funcionava correctamente.

5.3 Implementação da aplicação de gestão de dados do processo

Para simular este sistema foi criada uma aplicação em Delphi, capaz interagir com o processo, isto é agendar o número de peças a criar, verificar o tempo de ciclo de cada máquina, mostrar o número peças trabalhadas pelo sistema. A interacção com o processo de controlo, é feita através de um servidor de modbus, no qual o sistema de controlo irá ler e escrever nas várias variáveis correspondentes.

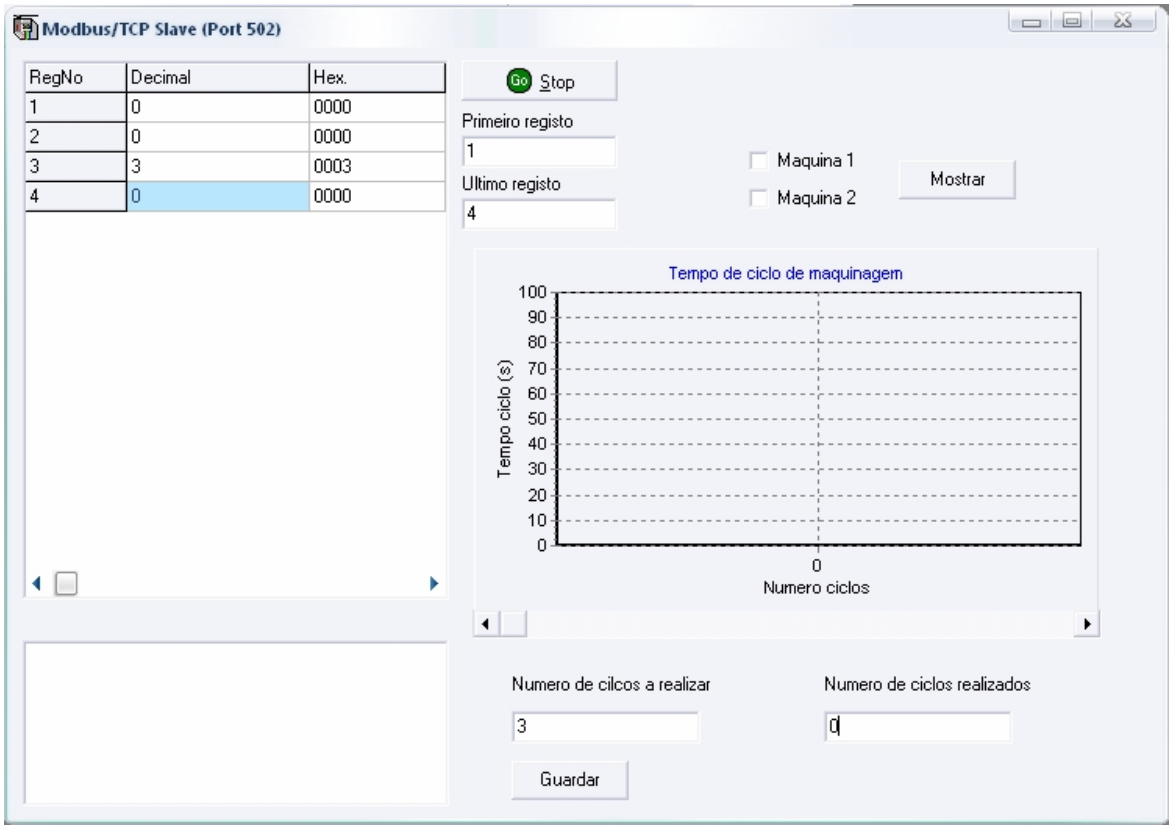


Figura 5.11 *- Aplicação de gestão e controlo do processo*

Note-se que a comunicação de dados entre a aplicação e o sistema de controlo, realizou-se sobre modbus/TCP. Deste modo foi necessário dotar a aplicação desenvolvida em Delphi de funções capazes de comunicar sobre Modbus/TCP, foi necessário utilizar uma livraria específica (delphimodbus-1.0.0), que permite a criação de mestres e escravos para comunicar via Modbus. Nesta aplicação implementou-se um escravo, de modo a que fosse o sistema de controlo o responsável pela leitura e escrita das variáveis.

5.4 Teste Software de controlo de simulação

Após a implementação do algoritmo de controlo de simulação, torna-se necessário validar o mesmo, de modo a verificar quais os limites de funcionamento associados aos vários sistemas. Os sistemas testados foram, o temporizador, a frequência de ciclo de *polling* e a frequência de aquisição para o *encoder*.

5.4.1 Teste temporizador

O temporizador de sistema, é o responsável pela temporização de todo o sistema de simulação, nomeadamente a actualização dos tempos de atraso de cada evento registado e a verificação de novos eventos registados via *web browser*. A implementação deste foi feita recorrendo a uma função do sistema operativo, o *ualarm()*, que é uma função que conta um determinado numero de micro-segundos, após os quais gera uma interrupção no sistema operativo, que irá desencadear a execução da função responsável por essa interrupção. Na função de atendimento da interrupção, serão incrementados os vários temporizadores de *software* registados.

As medições do tempo de atraso do temporizador foram realizadas, alternando ciclicamente o estado de um pino de saída, sendo possível deste modo, verificar qual a frequência a que este esta a comutar e consequentemente o período de funcionamento.

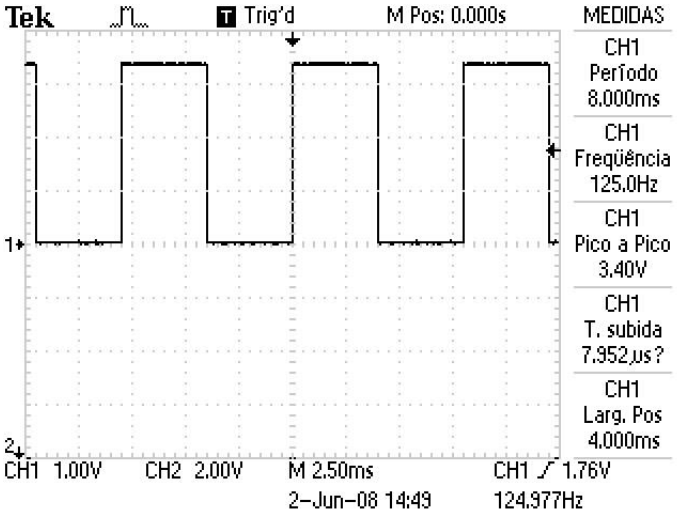


Figura 5.12 *- Atraso da função ualarm para 1ms*

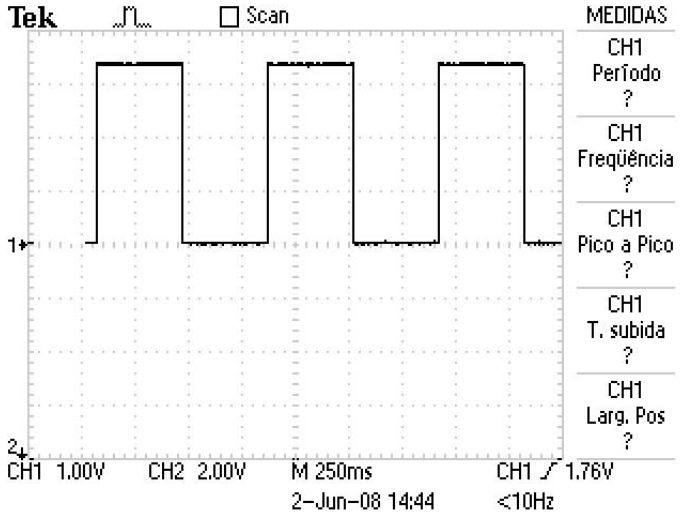


Figura 5.13 *- Atraso da função ualarm para 10ms*

Como se pode observar pela figura 5.12 e figura 5.13, o atraso de atendimento da função mantém-se constante (4ms por cada ms). Estes valores não são críticos, visto que o sistema não necessita de uma elevada resolução temporal, tipicamente 100ms, o que, configurando a função ualarm para 10ms (40ms reais), é mais que suficiente para a geração de impulsos para os temporizadores. Observou-se ainda que estes tempos não sofrem de influências causadas pelo *jitter*, mantendo-se constantes em qualquer situação de carga.

5.4.2 Teste Ciclo de *Polling*

Para a medição da frequência do ciclo de *polling* do programa do "Sistema de Supervisão dos Sinais de Controlo" foi utilizado o mesmo método do ponto anterior, onde, dentro do ciclo *principal* foi colocado um pino a comutar de estado. Deste modo foi possível medir a frequência máxima e mínima a que o sistema funciona.

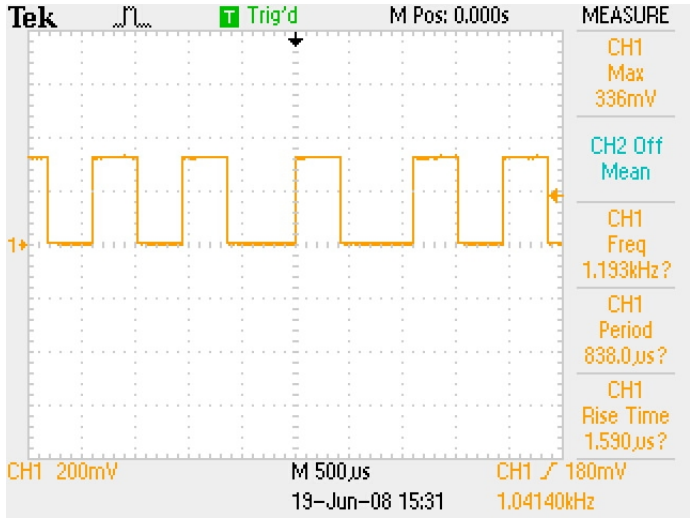


Figura 5.14 *- Frequência de ciclo do programa de controlo de simulação*

No caso apresentado na figura 5.14, a frequência de ciclo é afectada por uma variação aleatória (*jitter*), causada, na sua maioria pelo facto de o programa estar a correr sobre um sistema operativo, que tem que fazer a gestão de vários processos e de vários periféricos que geram interrupções prioritárias em relação ao programa em questão. Os testes de frequência de ciclo foram feitos com vários cenários de funcionamento:

- Apenas com os serviços básicos a funcionar, isto é, servidor httpd e programa principal sem eventos registados.
- Serviços básicos mais modo *encoder*.
- Serviços básicos mais modo intercepção de eventos.
- Serviços básicos mais modo *encoder* e modo intercepção de eventos.
- Todos os anteriores e *flood* da porta ethernet.

Após estes testes verificou-se que o tempo de ciclo variava aleatoriamente entre 1,2KHz e 800Hz, estando o sistema mais próximo desta ultima quando testado com todos os módulos em funcionamento.

5.4.3 Teste da frequência de aquisição para o *encoder*

Para o teste de aquisição de rotações do *encoder* foi utilizado um motor acoplado directamente ao *encoder*, para deste modo poder avaliar o comportamento da rotina de contagem para diferentes frequências de rotação. A tabela 5.1 mostra as medições feitas e respectivos erros.

Tabela 5.1 - Estimativa do erro de contagem do encoder

Tensão motor (V)	***Frequência encoder (Hz)***	***Impulsos lidos micro***	***Frequência micro (Hz)***	***Erro Abs (Hz)***	***Erro (%)***
3	142	16774	140	2,2	1,6
3	142	16639	139	3,3	2,4
3	142	16639	139	3,3	2,4
3	142	16974	141	0,6	0,4
5	280	34112	284	4,3	1,5
5	280	33342	278	2,1	0,8
5	280	34110	284	4,3	1,5
5	270	32834	274	3,6	1,3
5	270	32681	272	2,3	0,9
8	512	61132	509	2,6	0,5
8	512	60523	504	7,6	1,5
8	512	60628	505	6,8	1,3
9	600	67890	566	34,3	5,7
9	600	68139	568	32,2	5,4
12	856	91930	766	89,9	10,5
12	856	92373	770	86,2	10,1

Note-se que as frequências lidas do *encoder* apenas representam o sinal de um dos canais, e que o número de impulsos lidos pelo programa representam qualquer mudança no estado lógico (mudança de flanco) do sinal em qualquer um dos canais, ao longo de 30 segundos de rotação.

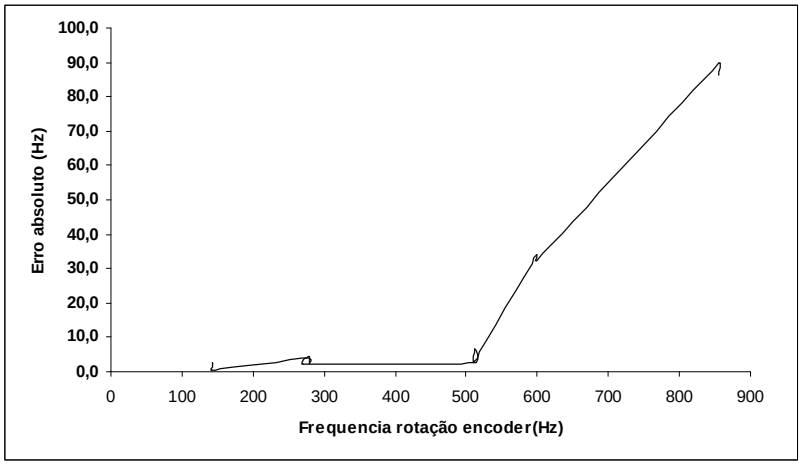


Figura 5.15 *-Gráfico da relação frequência rotação/Erro absoluto medição*

Como se pode verificar pela tabela 5.1 e figura 5.15, a partir dos 500Hz, o programa começa a perder impulsos. Isto deve-se ao facto de a frequência de rotação passar de metade da frequência de verificação (frequência do ciclo de *polling*) do sistema.

Convertendo a frequência do sinal do *encoder* para rotações por minuto pode-se verificar que o sistema nunca poderá controlar sistemas com velocidades superiores a 600 RPM (500HZ), isto no caso do uso de *encodes* de 100 impulsos por rotação.

Se a resolução dos *encoders* for superior terão de ser usadas velocidades mais reduzidas (razão inversa), uma vez que o programa terá mais impulsos para contabilizar em igual período de tempo, para a mesma velocidade de rotação.

5.5 Teste geral do sistema

Uma vez criados todos os módulos de hardware e software, passou-se ao teste da solução no sistema real. Para isso montou-se o sistema com o esquema descrito na figura 5.16.

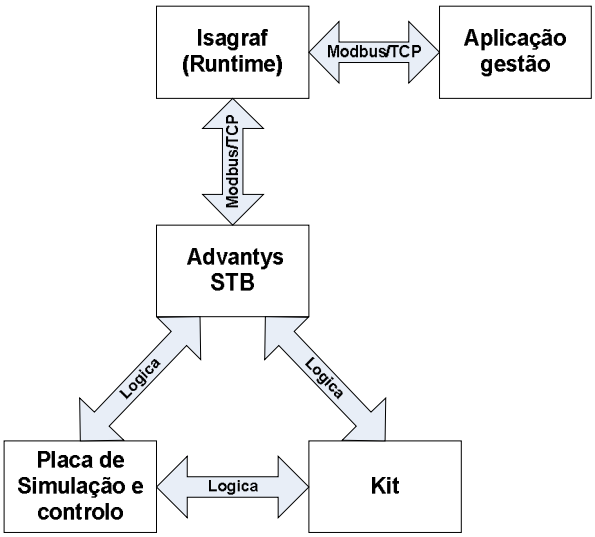


Figura 5.16 *- Esquema implementado para de teste geral da solução*

Para efectuar os testes colocou-se a aplicação de controlo a correr no *runtime* do Isagraf, interceptaram-se 2 sinais (uma entrada e uma saída) de modo a verificar se o sistema respondia de acordo com o pré-configurado.

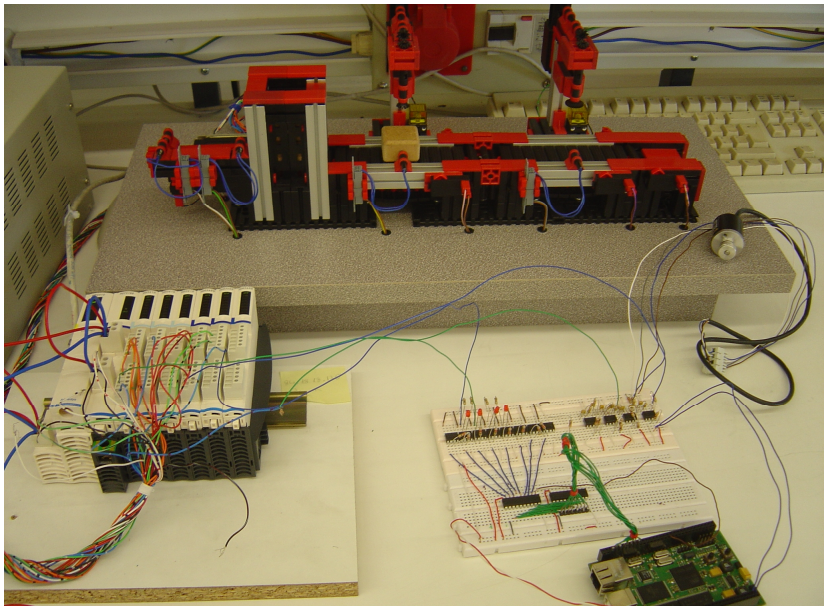


Figura 5.17 *- Montagem do circuito para teste*

Capítulo 6

6. Conclusões

Este capítulo conclui a tese com a apresentação de todo o trabalho realizado ao longo desta dissertação, sendo deixadas no final algumas perspectivas de desenvolvimento.

6.1 Trabalho realizado

Esta dissertação apresentou a análise e projecto de um sistema de interface para integrar com os kits didácticos adquiridos para o laboratório de ensino de automação da FEUP. Foram discutidas algumas das plataformas capazes de suportar esse mesmo sistema, as diferentes tecnologias existentes para a criação de um sistema de gestão e controlo baseado na *web* e algumas das arquitecturas de controlo utilizadas nos sistemas de automação.

Foi implementado um pequeno sistema que serve de prova de conceito para as ideias que se pretende implementar. Foram identificadas algumas falhas na implementação produzida, a maior parte das quais relaciona-se com o facto de as bibliotecas de software existentes para o funcionamento da placa escolhida, não estarem completamente desenvolvidas ou apresentarem alguns "*bugs*" de desenvolvimento.

Os aspectos mais relevantes relativamente a solução desenvolvida são:

- Implementação de um sistema capaz de atrasar a propagação dos sinais de controlo;
- Geração de sinais virtuais;
- Leitura e escrita de de I/Os;
- Tratamento de *encoders;*
- Estudo e desenvolvimento de uma plataforma de isolamento para os sinais do "Sistema de Supervisão dos Sinais de Controlo";
- Implementação e respectivo "*port*" de um cliente modbus para a plataforma usada;
- Estudo da tecnologias *web* passíveis de ser usadas no "Sistema de Supervisão dos Sinais de Controlo";

- Implementação do algoritmo de controlo de uma parte do kit.

Todas estas soluções foram desenvolvidas com sucesso, tendo-se verificado que o sistema funcionava correctamente nas várias situações de funcionamento requeridas.

Conseguiu-se comprovar as vantagens da utilização de um sistema de gestão e configuração baseado em tecnologias *web*, tendo-se verificado que a plataforma *web* se mantêm funcional em qualquer browser (Internet Explorer, Firefox, Safari) e em vários sistemas operativos (Windows XP, Ubuntu 7.01 e Leopard 10), o que permite uma total independência da plataforma usada para a configuração do "Sistema de Supervisão dos Sinais de Controlo".

Pode-se observar nos capítulos anteriores que, o número de entradas e saídas da placa de controlo de simulação são respectivamente 40 e 48, o que leva a que, no caso dos kits com um total de IOs superiores a 40, não sendo possível a intercepção de todos os sinais. Isto deve-se ao facto de que por cada sinal interceptado, serem necessárias duas interfaces da placa de controlo de simulação (uma entrada e uma saída).

O código desenvolvido para a aplicação de controlo de simulação, deixa em aberto a possibilidade de adicionar mais funcionalidades para eventos, uma vez que estes são registados mediante o recurso a uma tabela e a uma lista de tipos de eventos.

6.2 Desenvolvimentos futuros

Para desenvolvimentos futuros poder-se-á implementar o ciclo de detecção de mudança do estado dos pinos por interrupções, o que permitiria, no caso do uso dos *encoders* uma medição mais precisa, uma vez que deste modo não se esta dependente da frequência do ciclo de verificação para a contagem de impulsos.

Seria também interessante poder utilizar a placa como se de uma ilha de IOs remotos se tratasse, fazendo uso do cliente modbus para receber e enviar comandos para a actuação nos IOs.

Para finalizar, seria interessante estudar os limites de funcionamento dos vários kits, e, em conjunto com um cliente e um servidor Modbus TCP a correr na placa, verificar se os comandos enviados pelos controladores são adequados ao correcto funcionamento do kit. Este modo poderá também ser desenvolvido utilizando o modo de intercepção de sinais, no qual o sistema irá verificar se os sinais eléctricos enviados para o kit são adequados ao seu correcto funcionamento.

Referências Bibliográficas

[1] Matias Harjula, "*SoftPLC vs. PLC*". Disponível em http://www.automationit.hut.fi/file.php?id=813. Consultado em 13/04/2008.

[2] Constantino Seixas Filho, "*A automação nos anos 2000 uma análise das novas fronteiras da automação*". Disponível em http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/Conai2000Automacao.pdf. Consultado em 20/02/2008.

[3] Cynthia M. Hollenbeck, "e-Automation: Endless Possibilities"

[4] "*What is SCADA?*". Disponível em http://www.tech-faq.com/scada.shtml. Consultada em 12/06/2008

[5] http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Consultado em 7/06/2008

[6] http://ethernet.industrial-networking.com/protocols/modbus.asp. Consultado em 7/06/2008

[7] http://ethernet.industrial-networking.com/protocols/ethernetip.asp

[8] http://www.siemon.com/br/whitepapers/ethernet-ip.asp

[9] http://www.ab.com/en/epub/catalogs/12762/2181376/214372/1810894/3404056/index.html

[10] http://www.industrial-ethernet.ie/industrial-solutions.html

[11] http://www.bb-europe.com/images/product_images/Receptacles.jpg

[12] Montague, Jim 2003. "Ethernet Hits Real-Time...Really, Control Engineering", Disponível em http://www.controleng.com/article/CA339683.html

[13] http://www.odva.org

[14] Shishir Gundavaram, "CGI Programming on the World Wide Web" – O'REILLY

[15] Daniel J. Berlin, Ken Hunt, Shuman Ghosemajum*der*, "CGI Programming Unleashed"

[16] http://ezinearticles.com/?JavaScript-for-Web-Design---Advantages-and-nDisadvantages&id=645013

[17] http://www.expertrating.com/courseware/JavaScriptCourse/JavaScript-Introduction-1.asp

[18] "Ajax Bible" by Steven Holzner : John Wiley & Sons

[19] http://www.adaptivepath.com/ideas/essays/archives/000385.php

[20] Gilmore, W. Jason (2006), "*Beginning PHP and MySQL 5: From Novice to Professiona*", Apress

[21] David Powers, "*PHP Solutions: Dynamic Web Design Made Easy*", Apress (2007)

[22] http://www.jquery.com

[23] P. Raghavan, Amol Lad, Sriram Neelakandan, "Embedded Linux System Design And Development", 2006

[24] Christopher Hallinan, "*Embedded Linux Primer: A Practical, Real-World Approach*", September 18, 2006

[25] Craig Hollabaugh, "*Embedded Linux®: Hardware, Software, and Interfacing*", March 07, 2002

[26] Richard Zurawski, "Embedded systems Handbook".

[27] "*Real-Time and Embedded Computing Systems and Applications*", 9th International Conference, RTCSA 2003, Tainan, Taiwan, February 18-20, 2003.

[28] http://www.upto11.net/generic_wiki.php?q=embedded_system

[29] "*Using Linux in Embedded Systems and Smart Devices*" www.linuxdevices.com

Anexo 1 – Guia de uso da placa ICnova

O documento que se segue, descreve algumas das instruções para utilizar os GPIOs (General Purpose inputs / outputs, Port-Pins), a interface web..

Para seguir este tutorial é necessária conhecer alguns dos comandos linux/unix, bem como a utilização do editor de consola "vi".

Principais características:

Porta USB configurada para NETWORK com o endereço 192.168.93.1

Porta ethernet configurada para DHCP

Porta Série (TTL 3,3V) para acesso a consola ou as mensagens de BOOT

Pode ser alimentada via porta USB ou através do jack ao lado da entrada USB. Note-se que para alimentar a board via jack, deve-se apenas usar tensões entre 5-10V (o valor imprimido na board é incorrecto e pode levar a destruição da mesma)

Para acesso a consola pode ser usado qualquer um das seguintes interfaces:

USART – para aceder a consola via este interface, é necessária a utilização de um adaptador de tensão, uma vez que a board comunica com o nível 3,3V. O PC deve ser configurado para 11520-8-#-1.

USB – Em sistemas Windows é necessário um driver, que ira transformar a porta usb em uma porta série, passando a ser usadas as configurações referidas no ponto anterior. Em sistemas Linux/Unix, após a ligação do cabo USB, é necessário configurar a interface ethx com endereço na mesma subnet da placa

su ifconfig ethx 192.168.93.2 netmask 255.255.255.0

Posteriormente, o acesso a consola é feito via telnet no ip 192.168.93.1

Ethernet – Para utilizar esta interface basta ligar a placa a um dispositivo com capacidade de servidor de DHCP.

Posteriormente para acesso a consola, basta utilizar um cliente telnet no ip atribuído a placa.

Após ter sido estabelecida a ligação, é necessário autenticar, para isso basta utilizar como username ***default***, sem qualquer password.

Para iniciar o trabalho é necessário estar ligado como root, para isso execute o comando **su**.

Para configurar a interface ethernet com ip fixo é necessário editar o ficheiro */etc/network/interfaces* e alterar:

auto eth0

iface eth0 inet dhcp

Para

iface eth0 inet static

address 192.168.1.100

netmask 255.255.255.0

As definições de funcionamento da placa são carregadas durante o arranque do ficheiro */etc/inittab*.

A placa vem com um servidor httpd, sendo que os ficheiros estão localizados na pasta */var/www*. Sempre que for enviado um novo ficheiro para a pasta é necessário dar as correctas permissões.

O servidor httpd vem também com suporte para cgi, para usar este tipo de ficheiros é necessário criar a pasta */var/www/cgi-bin* e colocar lá, todos os ficheiros previamente compilados para placa, uma vez que a compilação deste servidor não tem suporte para perl ou php.

As configurações deste servidor encontram-se em */etc/httpd.conf*.

O driver GPIO

Nesta secção será explicada a forma de aceder aos gpios através do driver fornecido no kernel. De seguida encontram-se os passos a dar para criar e configurar um novo objecto gpio:

Criar uma nova subdirectoria em */config/gpio* com o comando *mkdir*. Se tudo correr bem, dentro da recente criada directoria iram aparecer 4 ficheiros:

- gpio_id: selecciona a porta a ser usada. Para a porta A escreve-se 0, para a porta B escreve-se 1, etc.

- pin_mask: Selecciona os pinos da porta configurada pelo gpio_id a serem usados. Para isso escreve-se um ‘1’ na posição referente aos pinos que se querem usar. Por defeito, todos os pinos estão configurados como entradas.
- oe_mask: define os pinos como entrada ou saída. Colocando ‘1’ para definir como saída ou ‘0’ como entrada.
- Enable: activa o objecto gpio criado. Para activar o gpio basta escrever ‘1’, sendo que em caso de sucesso um ficheiro gpiox irá ser criado na pasta */dev*.

Seleccionar a porta a ser usada com o comando **echo 0 > /config/gpio/xxx/gpio_id** (selecciona a porta A, para usar a porta B usar echo 1 ….)

Seleccionar os pinos da porta a serem usados com o comando **echo 0x0000003F > /config/gpio/xxx/pin_mask** #usa pinos 0-7

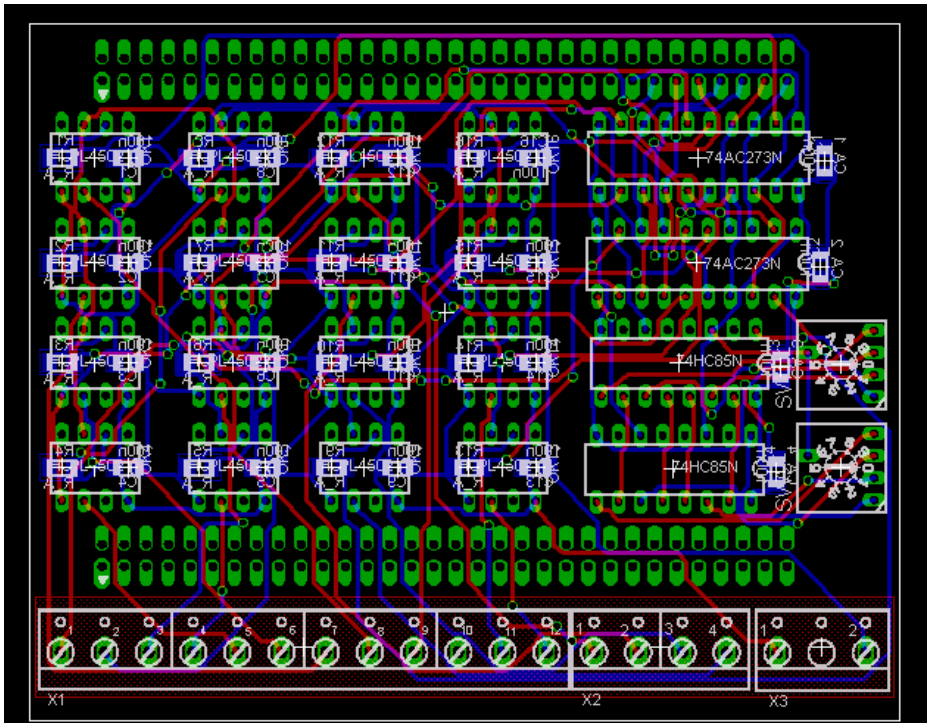
Selecciona os pinos a serem usados com entradas com o comando **Echo 0x0000000F > /config/gpio/xxx/oe_mask** #pinos 0-3 como saídas

Activar o gpio configurado com o comando **echo 1 > /config/gpio/xxx/enable**

Depois de activado, é possível ler e escrever do gpio mediante o uso das funções do sistema operativo open(), read(), write().

Anexo 2 – Circuitos do sistema de interface

Esquema físico da placa de saídas



Esquema físico da placa de entradas