**Departamento de Engenharia Eletrotécnica**

**Secção de Robótica e Manufatura**

**Relatório Primeiro Trabalho**

**Cadeira de Supervisão Inteligente**

Outubro 2015

Flávio Silva nº37671

João Ralo nº32146

REGENTE: Luis Manuel Camarinha de Matos

DOCENTE DAS PRÁTICAS: Tiago Cardoso

INSTITUIÇÃO: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Índice

[1. Introdução 1](#_Toc432259108)

[2. Implementação do módulo de Despacho 2](#_Toc432259109)

[3. Desenvolvimento da solução 3](#_Toc432259110)

[3.1. Setbit/Getbit 3](#_Toc432259111)

[3.2. Readports 3](#_Toc432259112)

[3.3. Security 4](#_Toc432259113)

[3.4. Move 4](#_Toc432259114)

[3.5. Interface gráfica do utilizador 5](#_Toc432259115)

[4. Conclusão 6](#_Toc432259116)

# Tabela de Figuras

[Figura 1 – Kit Armazém 1](#_Toc432258936)

[Figura 2 – Arquitectura do sistema 1](#_Toc432258937)

[Figura 3 - Representação do algoritmo implementado 2](#_Toc432258938)

[Figura 4 - Interface gráfica do utilizador 5](#_Toc432258939)

# Introdução

No âmbito da cadeira de Supervisão Inteligente, foi proposto aos alunos a elaboração de um trabalho que visa abordar os conceitos base associados ao desenvolvimento de um Sistema de Controlo e Supervisão Inteligente. Este irá ser implementado num *kit* robótico de um armazém automático existente no laboratório com 5 prateleiras de altura e 10 de largura (**Figura 1**).

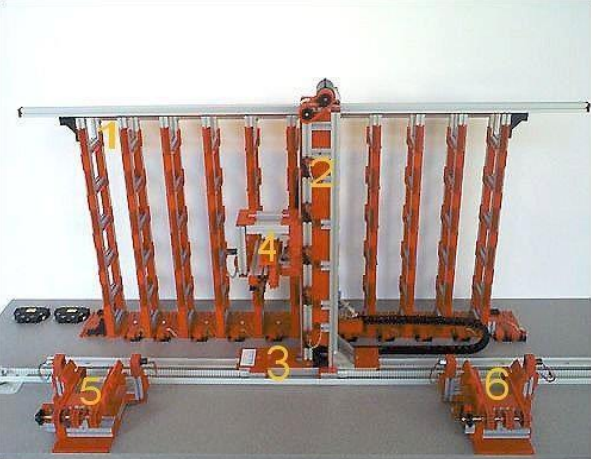


Figura 1 – Kit Armazém

O trabalho consiste na implementação de uma arquitetura constituída por cinco módulos diferentes, nomeadamente: planeador, monitor, despacho, diagnóstico e recuperador de erros (**Figura 2**). O projeto é dividido em duas partes. Primeiro vai ser abordado apenas o módulo de despacho.

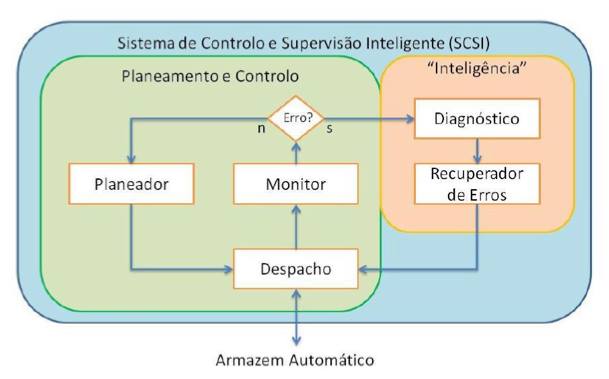


Figura 2 – Arquitetura do sistema

# Implementação do módulo de Despacho

Este módulo é o único que comunica com o *hardware*. Este possui 6 portos de 8 bits cada, onde do porto 0 ao 3 se pode ler informação sensorial do armazém. Nos dois portos restantes atua-se sobre os motores para o elevador se mover entre prateleiras e estações de mercadoria. Nesta primeira abordagem, o Despacho é implementado como uma simples aplicação de consola em linguagem C e recebe comandos diretamente do utilizador. O programa segue o seguinte algoritmo:

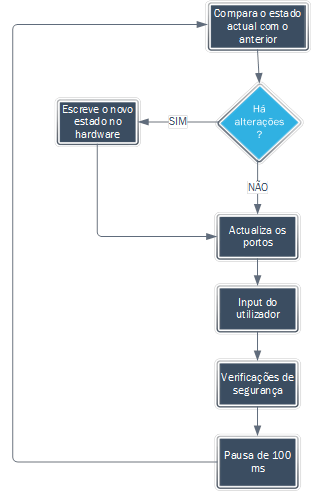


Figura 3 - Representação do algoritmo implementado

Numa fase posterior do projeto, este módulo sofrerá algumas alterações de modo a receber ordens dos outros módulos da arquitetura SCSI (planeador e recuperador de erros).

# Desenvolvimento da solução

## Getbit/Setbit

São duas funções auxiliares que permitem manipular e aceder ao valor de cada bit em cada porto. Isto permite que o elevador faça movimentos na diagonal (p.e. ligando o bit de cima e o da direita) porque atualizando apenas o valor dos bits necessários o restante não se altera, e permite saber a posição atual do robot.

int getBit(int var, int nBit) {

int result = var & (1 << nBit);

if(result != 0){

return 1;

}else {

return 0;

}

}

void setBit(int &var, int nBit, bool value) {

int mask\_on = 1 << nBit;

int mask\_off = 0xff - mask\_on;

if (value == TRUE) {

var = var | mask\_on;

}else {

var = var & mask\_off;

}

}

## Readports

Esta função verifica, porto a porto, se alguma coisa se alterou nos entre o último estado do sistema e o presente. Caso hajam alterações escreve-as no *hardware*.

void read\_ports(int ports[6], int last\_ports[6]) {

int i;

for (i = 0;i < 6;i++) {

if (ports[i] != last\_ports[i]) {

out\_port(i, ports[i]);

}

ports[i] = ReadDigitalU8(i);

last\_ports[i] = ports[i];

}

}

No fim atualiza o estado do porto para que se possa repetir processo.

## Security

Esta função verifica os sensores que dos extremos laterais, superior, inferior e limites interno e externo do elevador de modo a não colocar em risco a integridade do *hardware.*

void security(int ports[6]) {

//protege fora

if (getBit(ports[1], 2) == 0) {

setBit(ports[4], 3, FALSE);

}

//protege dentro

if (getBit(ports[1], 4) == 0) {

setBit(ports[4], 4, FALSE);

}

//protege esquerda

if (getBit(ports[0], 0) == 0) {

setBit(ports[4], 1, FALSE);

}

//protege direita

if (getBit(ports[1], 1) == 0) {

setBit(ports[4], 0, FALSE);

}

//protege cima

if (getBit(ports[1], 5) == 0) {

setBit(ports[4], 5, FALSE);

}

//protege baixo

if (getBit(ports[2], 6) == 0) {

setBit(ports[4], 6, FALSE);

}

}

Quando deteta que algum dos limites foi atingido desliga o bit do motor correspondente ao movimento que está a ocorrer e essa informação é depois passada ao hardware na função *readports()*.

## Move

A função *move* é um *switch* que recebe o *input* do utilizador e o valor dos portos para alterá-los consoante o input do utilizador.

void move(int ports[6], char aux) {

switch (aux) {

case 'w'://cima

setBit(ports[4], 6, FALSE);

if (getBit(ports[4], 5)) {

setBit(ports[4], 5, FALSE);

}

else {

setBit(ports[4], 5, TRUE);

}

break;

Aqui só está representada uma parte da função porque é muito extensa e repetitiva para descriminar na íntegra neste relatório.

## Interface gráfica do utilizador

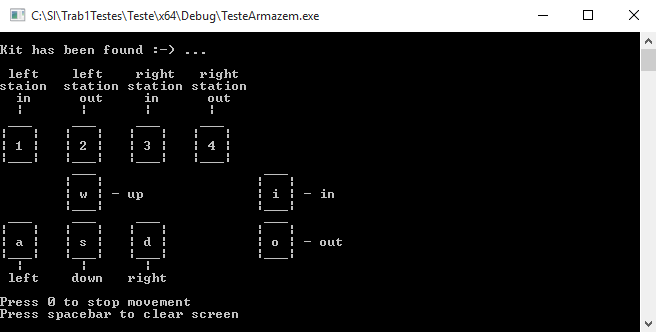


Figura 4 - Interface gráfica do utilizador

Foi concebida a interface gráfica representada na Figura 4. Para testar o trabalho desenvolvido. Como pedido, trata-se de uma simples consola em que são impressos no ecrã os comandos para comandar o módulo de despacho.

# Conclusão

Devido a uma explicação prévia do trabalho por parte do docente nas aulas práticas foi relativamente fácil implementar esta parte do projeto. O facto de ser possível, e fomentado pelo docente, testar a solução diretamente no *software* contribui como um fator motivador para o desenvolvimento deste projeto.

Como foi pedido, foi criado um “modelo do mundo real” através de um vetor de 6 inteiros, em que cada posição do vetor representa um porto do *hardware*. Infelizmente, devido a limitações de acesso a este, não foi possível testar o projeto no modelo físico presente no laboratório após a implementação deste modelo.

É de salientar ainda que, para uma utilização facilitada do programa de teste do armazém, existiu preocupação em apresentar ao utilizador uma interface fácil de utilizar, equivalente a um jogo de computador onde, por exemplo, se utilizam as teclas clássicas ‘WASD’ para mover o elevador nas direções cima, esquerda, baixo ou direita, respetivamente.

Em suma, os autores consideram que foram satisfeitos os requisitos propostos pela docência.