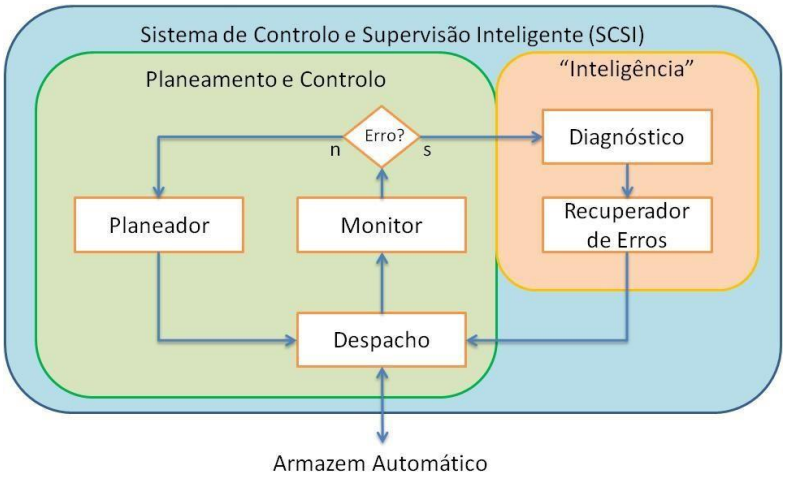
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Supervisão Inteligente 2014 / 2015

Relatório Trabalho Prático 1

Modulo DESPACHO



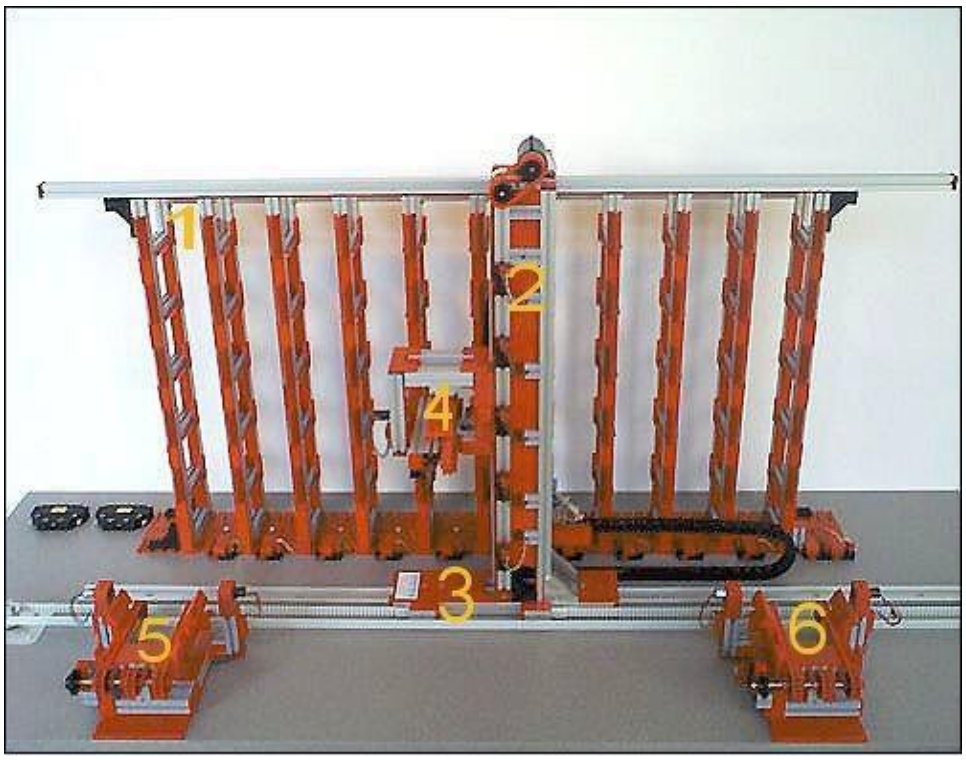
Projeto e Implementação de um Supervisor Inteligente

João Aires #34051
João Barata Oliveira #31559

Arquitetura proposta

Visão geral do Problema

Criar um projeto de supervisão inteligente para dar suporte ao armazém robótico. Primeira fase do trabalho: desenvolver o módulo despacho. Este módulo atua sobre o mundo real (seguindo um plano previamente definido). O despacho dispõe basicamente de operações de ativação de atuadores, capazes de garantir a materialização (no mundo real) das decisões do supervisor inteligente (e.g. acionar tapetes rolantes). É também neste módulo que é recebida toda a informação sensorial. Basicamente este é o único módulo que tem contacto com o Hardware.



Visão geral da Solução

a	Funções de interação com hardware **Hardware.cpp**	C++
b	Ponte entre linguagens de alto e baixo nível **Hardware.dll**	DLL
c	Classes e thread para garantir acesso otimizado ao hardware **Harware.java; BufferData.java; Updater.java**	JAVA
d	Funções de cumprimento de tarefas sobre plataforma do armazém **GotoPosition.java; PutGet.java**	JAVA
e	Classes de otimização de comandos para realizar diferentes tarefas, **Command.java; Dispatcher.java; Action.java**	JAVA
f	Interface gráfica de demonstração **Test_Gui.java**	JAVA

Criação da DLL hardware.dll responsável pela comunicação entre as funções C++ e Java

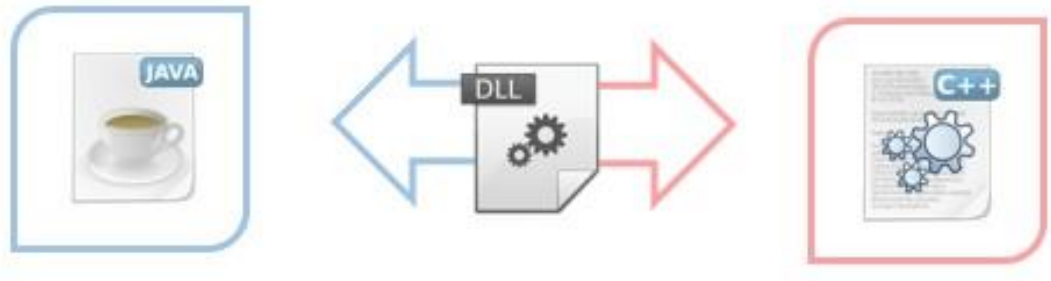
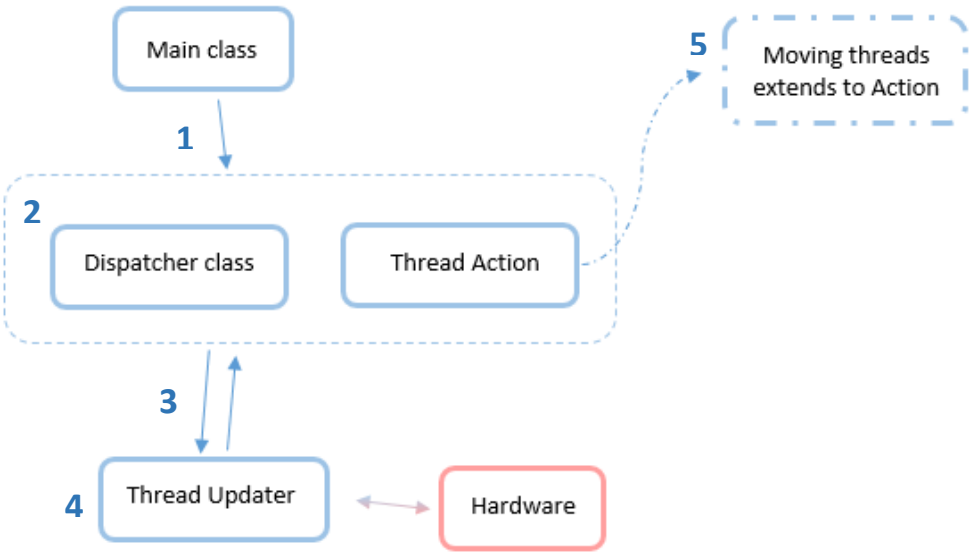


Diagrama da arquitetura de threads usada no módulo dispatcher



1. Foi criada uma classe chamada “Command.java” que transforma todas as ordens vindas da “Main” para o “Dispatcher” em objetos para melhorar a sua execução.

2. A classe “Main”(Trab1.java), invoca todo o módulo de Dispatcher. Módulo esse constituído por uma classe específica “Dispatcher.java” com a obrigação de implementar ordens vindas da classe Main.

3. Foi criada uma classe chamada “BufferData.java” para dar suporte aos pedidos de escrita e leitura entre as várias threads como as “extends Action”.

4. A thread “Updater.java” é o elemento responsável pela leitura/escrita no hardware. Recebe os pedidos de escrita e dá acesso aos dados dos sensores, apenas escreve num sensor, se o valor do pedido feito for diferente do atual.

5. São criadas threads “extends Action” com tarefas predefinidas de deslocar a plataforma nos eixos x,y e z.