Pedro Miguel Pinto Xavier Rodrigues

Licenciado em Ciências da Engenharia Eletrotécnica

e de Computadores

Gestão e reprogramação de controladores, em tempo de execução, suportadas por IOPT-Tools

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador:	Doutor Filipe de Carvalho Moutinho, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Co-orientador:	Doutor Rogério Alexandre Botelho Campos Rebelo, Investigador, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente:	Prof. Doutor Fernando José Almeida Vieira do Coito
Arguente:	Prof. Doutora Anikó Katalin Horváth da Costa
Vogal:	Prof. Doutor Filipe de Carvalho Moutinho

Setembro, 2018

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Gestão e reprogramação de controladores, em tempo de execução, suportadas por IOPT-Tools

Para a minha família e amigos

Agradecimentos

Esta dissertação foi marcada por importantes apoios e incentivos provenientes de inúmeras pessoas, sem as quais não teria sido realizada e completada. Como tal quero agradecer a todos aqueles que de certa maneira se envolveram e ajudaram a alcança este objetivo.

Em primeiro lugar, ao Professor Doutor Filipe Moutinho e ao Professor Doutor Rogério Rebelo, orientador e coorientador desta dissertação, respetivamente, que se demonstraram sempre disponíveis para comunicar, rever e orientar todo o trabalho desenvolvido nesta dissertação.

Em seguida, o Engenheiro Fernando Pereira, que facultou o trabalho que desenvolveu ao longo dos anos e se demonstrou sempre disponível para apoiar na implementação e concretização deste projeto.

À coordenadora de curso Professora Helena Fino que ao longo do tempo se disponibilizou, aconselhando e orientando no sentido de me incentivar a atingir os objetivos.

Um muito obrigado a todos os docentes, amigos e colegas da instituição Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa, que contribuíram para o meu percurso. Em especial aos meus amigos e companheiros, Thomas Noronha, Ricardo Mota, Alexandre Caetano, Duarte Bragadesto, João Mouco, Francisco Carrola, João Pires, Tiago Carrasqueira e Carolina Lagartinho de Oliveira.

Um agradecimento enorme aos meus pais e à minha família. Hoje não estaria neste lugar sem eles foram e sempre serão um apoio enorme em toda a minha vida, deram-me um impulso nos momentos menos favoráveis e fizeram todos os esforços possíveis para estarem sempre presentes. Em especial à mãe, Luísa Rodrigues. e ao pai, João Rodrigues.

Por fim, não quero deixar de agradecer a uma pessoa especial que me acompanhou nos últimos anos, Filipa Aires Marcos, que me ouviu e aconselhou em todos os momentos bons e maus.

Resumo

A constante evolução da indústria leva a que, nos dias de hoje, se encoraje a automatização de processos e a personalização de produtos finais, bem como o melhoramento dos mesmos para a satisfação do consumidor final.

Desta forma, propõe-se, neste trabalho, um sistema capaz de gerir a execução e reprogramação de controladores, em tempo de execução, bem como a monitorização dos mesmos. Proporcionando, deste modo, reprogramação de máquinas e sistemas por forma a corrigir falhas, efetuar atualizações e facilitar a produção de produtos personalizados.

O sistema proposto permite que sistemas externos (humanos, nuvens ou outros dispositivos) possam controlar, reprogramar e supervisionar os controladores em execução. Esta interação pode ocorrer através de uma interface gráfica Web, no caso de humanos, ou através de mensagens, que permitem por exemplo a reprogramação e a paragem de controladores, bem como a aquisição e envio de informação.

O sistema implementado utiliza as ferramentas IOPT-Tools, nomeadamente o gerador de código *C* e o *debugger* (que permite a monitorização por humanos). As IOPT-Tools permitem ainda a receção dos modelos e a geração de código para diversas plataformas, possibilitando que sistemas externos possam transmitir modelos independentes de plataforma.

Palavras-chave: Gestor de execução, reprogramação, IOPT-Tools, modelos IOPT, automatização de processos

Abstract

The constant evolution of the industry means that today, the process automatization and the customization of final products are encouraged to be delivered by the producer, as well as the improvement of those products for the satisfaction of the final consumer.

In this way, it is proposed, in this work, a system capable of managing the execution and reprogramming of controllers, at runtime, as well as their monitoring. Enabling this way systems and machines reprogramming to correct failures, to make updates and facilitate the product customization.

The proposed system allows external systems (humans, clouds or other devices) to control, reprogram and supervise the running controllers. This interaction can occur through a graphical user interface, in case of humans, or through messages, which allows, for example, the reprogramming and stopping of controllers, as well as the acquisition and sending of data.

The implemented system uses the IOPT-Tools, namely the C-code generator and the debugger (which allows human monitoring). The IOPT-Tools also allow the reception of models and the generation of code for several platforms, allowing external systems to transmit controller models that are independent of the final platform.

Keywords: execution manager, reprogramming, IOPT-Tools, IOPT models, process automatization

Índice Geral

Índice de Tabelas

Índice de Figuras

Lista de Acrónimos

CPS – Cyber-Physical System

DS-Pnets – Data-flow, Signals and Petri-nets

EMF – Eclipse Modeling Framework

EWS – Embedded Web Server

FFT – Fast Fourier Transform

FPGA – Field-programmable gate array

GALS – Globally-Asynchronous Locally-Synchronous

GPRS – General Packet Radio Services

GRES – Group on Reconfigurable and Embedded Systems

GSM – Global System for Mobile communication

HTTP – Hypertext Transfer Protocol

IL – Instruction List

IOPT – Input-Output Place-Transition

IOPT-nets - Input-Output Place-Transition nets

IOPT-Tools – Input-Output Place-Transition - Tools

IOPT-Flow – Input-Output Place-Transition - Flow

IoT – Internet of Thing

MIS – Modelos de Interpretação de Sinais

MOF – Meta Object Facility

OPC UA – Open Platform Communications Unified Architecture

OTA – Over the air

OTAP – Over the air Programming

PHP – PHP: Hypertext Preprocessor

PLC – Programmable Logic Controllers

PNML – Petri Net Markup Language

RdP – Rede de Petri

RFID – Radio-Frequency Identification

SDK – Kits de Desenvolvimento de Software

USB –Universal Serial Bus

VHDL – VHSIC Hardware Description Language

WSN – Wireless Sensor Network

XML – eXtensible Markup Language

1 Introdução

Neste primeiro capítulo é apresentado o enquadramento da presente dissertação, identificado um problema e apresentadas as motivações e objetivos para a sua resolução. Por fim é descrita a estrutura do documento, onde são apresentados os capítulos e um pequeno sumário de cada um.

1.1 Enquadramento e motivação

Hoje em dia vivemos na era da quarta revolução industrial, denominada indústria 4.0. Enquanto que as anteriores revoluções industriais implementaram os conceitos de máquinas mecânicas movidas pelo movimento de água ou vapor, produção em massa com o uso de energia elétrica e automação com o recurso a componentes eletrónicos, respetivamente [1], esta revolução foca-se essencialmente na indústria inteligente e conectividade de dispositivos e máquinas com recurso à *Internet of Things* (IoT), computação em nuvens e *Big Data*.

A indústria 4.0 torna possível a capacidade de recolha e partilha de informação inerentes às máquinas e à respetiva produção para realizar decisões estratégicas de produção e manutenção em tempo de execução. A recolha de informação é realizada por componentes de leitura de sensores que posteriormente transmitem essa informação para um servidor de armazenamento centralizando-a num único local, podendo ser uma nuvem local ou remota da fábrica [2]. Esta inúmera informação centralizada e inserida em forma de nuvem, denominada *Big Data* [3], possibilita realizar monitorização e partilha de dados com outras máquinas ou indústrias fora do ambiente fabril.

Esta revolução não acarreta apenas o fator de recolha e partilha de informação da fábrica, mas também a informação do lado do cliente, que requisita/compra os produtos desenvolvidos. Desta forma o cliente passou a poder personalizar os produtos, mudando a visão da produção,

tornando-a a pedido em vez de produção em massa. Para realizar esta personalização é necessário adaptar a produção industrial, alterando o comportamento e operações de cada máquina, em tempo real.

A informação centralizada e o processamento da mesma, em tempo real, permite a otimização de técnicas tanto ao nível da manutenção como a nível encómio. Ao nível da manutenção, o conhecimento do estado da máquina durante o seu tempo de execução possibilita a manutenção atempada bem como a otimização do processo de produção envitando a perda de recursos. A nível económico, o aproveitamento da informação poderá evitar a produção em excesso.

Ao longo dos anos têm se vindo a desenvolver componentes capazes de captarem informação sobre o estado do sistema usando sensores e transmitindo essa informação por uma rede privada até ao local de armazenamento. As *Wireless Sensor Network* (WSN) [4] definem-se como uma rede de nós em forma de teia. Estes nós têm como finalidade adquirir a informação do meio e transmiti-la para o exterior de forma a facultar uma imagem dos sistemas em execução. Cada nó da rede tem a capacidade de ser reprogramado em tempo real, porém a única tarefa capaz de executar é a leitura do meio em redor, não tendo a capacidade de alterar outros sistemas. Por outro lado, existem dispositivos industriais que controlam o funcionamento de máquinas, nomeadamente PLCs (Programmable Logic Controllers), estes dispositivos podem ser programados para executarem uma determinada tarefa de controlo ou monitorização [5].

Tendo em conta estes fatores, a possibilidade de criar sistemas capazes de controlar, adquirir e partilhar dados de máquinas e outros sistemas, bem como de se reprogramar, em tempo de execução, revela-se um desafio e uma motivação, podendo ser um contributo para a evolução industrial.

1.2 Âmbito e objetivos

O principal objetivo deste trabalho é propor/desenvolver um sistema capaz de gerir a execução, reprogramação e monitorização, em tempo de execução, de controladores para sistemas embutidos e de automação. Pretende-se que este sistema possa receber os modelos dos controladores, gerar automaticamente código, compilar e executar esse código na plataforma associada. Este sistema deverá permitir a reprogramação de controladores que se encontram em execução, bem como a interação com sistemas remotos, por forma a:

- Receber modelos de controladores;
- Receber instruções de gestão da execução dos modelos/controladores;
- Receber informação para afetar as entradas dos controladores;
- Fornecer informação do sistema que permita a monitorização remota.

Este trabalho tem ainda como objetivo concluir sobre a adequabilidade de ferramentas de automatização de projeto para suportar este tipo de sistemas, nomeadamente um conjunto de ferramentas denominadas *Input-Output Place-Transition-Tools* (IOPT-Tools), e propor, uma nova ferramenta integrada nas mesmas. As IOPT-Tools permitem a especificação de sistemas de automação e sistemas embutidos usando um ambiente gráfico que permite criar, simular e verificar propriedades de modelos *Input-Output Place-Transition-nets* (IOPT-nets), bem como a geração automática de código *C*, *Instruction List* (IL), *JavaScript* e *VHSIC Hardware Description Language* (VHDL), que suporta a implementação em plataformas heterogéneas.

1.3 Estrutura da dissertação

Nesta secção é apresentada a estrutura do documento referindo brevemente o conteúdo de cada capítulo. A presente dissertação encontra-se segmentada em seis capítulos: Introdução, Estado de Arte, Gestor de execução e reprogramação, Implementação do gestor, Validação e Conclusões e Trabalhos futuros.

No Capítulo 1–Introdução, é realizado o enquadramento e motivação do trabalho e feita uma introdução ao problema. Em seguida são apresentados os objetivos do trabalho e por fim é apresentada a estrutura do documento.

No Capítulo 2 – Estado de Arte, é feita uma revisão sobre projetos e trabalhos científicos na área estudada. Temas como, *Wireless Sensor Network*, *Over-the-air Programming*, interfaces gráficas para a indústria e laboratórios remotos são abordadas neste capítulo. São ainda apresentados formalismos *Input-Output Place-Transition* e ferramentas associadas que irão dar suporte ao desenvolvimento da presente dissertação.

No Capítulo 3 – Gestor de execução e reprogramação, é apresentada a arquitetura do sistema proposto para a resolução do problema encontrado. Explicando a sua aplicação, funcionalidades e a comunicação entre diferentes sistemas, bem como os requisitos para a correta implementação.

No Capítulo 4 – Implementação do gestor, descreve-se a implementação do gestor de execução e reprogramação, que foi parcialmente suportada pelas ferramentas IOPT-Tools. São explicadas as adaptações efetuadas nas ferramentas de modo a permitir o correto funcionamento da implementação bem como as funções adicionais para a comunicação entre sistemas externos, utilizadores e plataformas de desenvolvimento.

No Capítulo 5 – Validação, é apresentada a abordagem utilizada para a validação da implementação realizada, bem como os modelos usados para teste, resultados obtidos.

No Capítulo 6 – Conclusões e Trabalhos futuros, são apresentadas as conclusões e propostos alguns trabalhos futuros associados a esta dissertação.

2 Estado de Arte

Neste segundo capítulo é apresentado o Estado de Arte, sendo apresentados conceitos sobre estudos e trabalhos desenvolvidos na área estudada, bem como os formalismos de modelação e as ferramentas IOPT, que são utilizadas como suporte à presente dissertação.

2.1 Projetos e trabalhos científicos relacionados

Nos últimos anos têm surgido diversas tentativas de modelação e implementação de sistemas capazes de promover a otimização em sistemas industriais. Nesta secção apresentam-se um conjunto de projetos e trabalhos científicos relacionados com a temática que servem de estudo para a elaboração e desenvolvimento desta dissertação.

É introduzido o conceito de *over-the-air* (OTA) [6] que utiliza comunicação sem-fios para a distribuição de software até aos dispositivos e o conceito de *Wireless Sensor Network* (WSN). Esta contém nós de sensores para monitorização de condições física e ambientais para posterior análise e processamento de informação, transmitindo todos os dados relevantes por meio sem-fios. São apresentados alguns dispositivos que juntam as vantagens da tecnologia OTA com os dispositivos WSN projetam soluções para o mercado *Internet of Things* (IoT).

2.1.1 Over-the-air

O *over-the-air* refere qualquer tipo de transmissão por via sem-fios sendo habitualmente usada na distribuição de atualizações de software, pacotes de dados e transmissão de rádio ou televisão. Com os novos conceitos de WSN e IoT, onde podem existir dezenas ou centenas de dipositivos ligados entre si; o OTA foi aplicado a sistemas [7][8] com capacidades móveis tais como 4G, 3G e GPRS e protocolos sem licença, tal como o WiFi, para o envio de pacotes de

dados remotamente até aos dispositivos moveis ou estáticos na rede, ou seja, sem necessitar a deslocação ao local ou recolha dos dispositivos para a transferência de informações.

2.1.2 Wireless Sensor Network

Uma rede de sensores sem fios (WSN – Wireless Sensor Network) [4] consiste numa rede de comunicação sem fios entre dispositivos autónomos, distribuídos num ambiente, por exemplo numa fábrica, que contêm nós de sensores para monitorização de condições físicas e ambientais, sendo que as leituras dos dados destes sensores são transmitidas via sem fios, para futura análise e processamento. A WSN contêm três componentes principais para o seu funcionamento: nós, gateway e software [9]. Os nós são dispositivos físicos, que se encontram ligados a sensores e entre si, formando uma rede na qual a informação é transmitida sem fios entre cada nó até chegar ao gateway. O gateway realiza a conversão da comunicação sem fios, proveniente dos nós, para uma comunicação com fio, conectando toda a informação num servidor ou computador. Por fim o software, executado nos servidores ou computador dedicado para a tarefa, realiza o processamento e análise da informação alocada com o objetivo de obtenção de dados estatísticos, otimizações entre outros.

O processo de implementação de uma WSN consiste, normalmente, em três etapas: na primeira etapa, cada nó realiza uma difusão do seu estado para os restantes nós existentes no espaço e recebe o estado dos mesmos por forma a detetá-los; seguidamente, na segunda etapa, os nós são organizados e é estabelecida uma rede de acordo com a topologia; por fim, na terceira etapa, são estruturados os itinerários que a transmissão de dados irá percorre entre cada nó até ao destino final (Figura 2.1) A fonte de energia usada, pela rede de nós de sensores, é frequentemente um conjunto de baterias que por sua vez limitam e reduzem a distância máxima de transmissão.

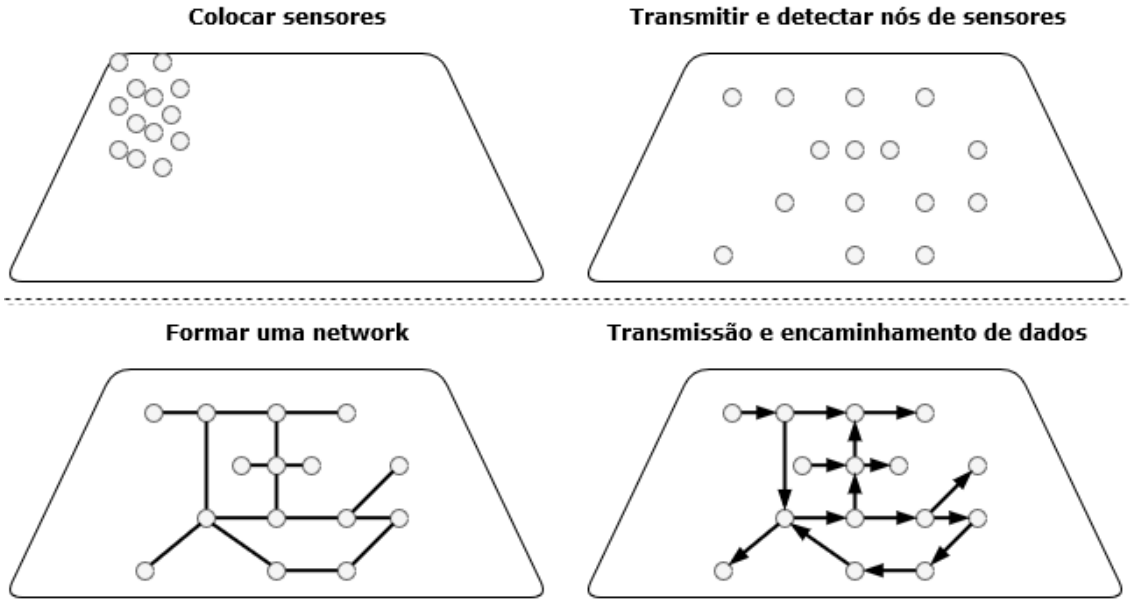

Figura 2.1 -Formação da rede WSNs. Adaptado de [9].

Os nós de uma WSN são capazes de se auto-organizar na rede, o que melhora significativamente a robustez da rede, resultando num conjunto de malhas inteligentes e adaptativas. Por forma a exercerem esta capacidade é necessário que o primeiro nó da malha monitorize os vizinhos e meça a intensidade do sinal e só posteriormente é que seleciona o vizinho apropriado para a sincronização, enviando um pedido de junção. Em seguida, o nó vizinho entrega o pedido ao gateway, que solicita e atribui recursos ao nó. Os nós da rede podem ser atribuídos com dois ou mais caminhos de transmissão para melhorar e otimizar a segurança da rede.

Técnica de redução de consumo de energia na transmissão de pacotes de dados

Uma vez que os nós dos sensores sem fios usam maioritariamente baterias internas, deverão ter baixo consumo de energia de forma a manterem-se ligados à rede durante longos períodos. É necessária uma gestão de energia e de recursos para a banda de comunicação, capacidade de armazenamento de dados e transmissão de dados por fim a estender o tempo de vida de cada nó na rede. A energia consumida na transmissão de dados é superior à energia de processamento de dados na rede [9].

Para impedir os problemas anteriormente mencionados, são usadas técnicas de agregação de dados. Este processo consiste na integração de diversas cópias de informação numa única copia, que é eficaz e capaz de responder às necessidades do utilizador. A integração de agregação de dados introduz benefícios tanto na economia de energia bem como na obtenção de informações precisas, sendo desenvolvidos para a realização de remoção de grandes quantidades de informação redundante, a fim de minimizar a quantidade de transmissão e economizar energia.

Problemas e desafios na reconfiguração de uma WSN

Existem ainda problemas associados à reconfiguração de uma WSN, nomeadamente, espaço de memória limitado e o consumo de energia. Segundo [10], não existe nenhum sistema operativo otimizado para a implementação ideal dos problemas referidos anteriormente, porém cada sistema operativo existente (TinyOS [11], Sensor Operating System [12], Contiki [13] e Mantis [14]) possuí características que ajudam num determinado aspeto da implementação de uma nova atualização/reconfiguração. A conclusão retirada é que todos os sistemas mencionados se focam no tempo reduzido de acesso à memória local do nó para reduzir o consumo de energia e sobrecarga de dados no sistema. A avaliação efetuada a cada sistema operativo permitiu entender que o sistema TinyOS encontra-se mais otimizado para uma pequena atualização enquanto que o Contiki e Sensor Operating System são melhores para uma reconfiguração total. Deste forma, os autores de [10], propõem o uso de um motor de decisão capaz de selecionar o melhor sistema para a implementação da reconfiguração desejada basada na lógica fuzzy.

2.1.3 OTA em WSN

Protocolos OTA têm vindo a ser propostos para WSN [15], existindo já diversos produtos no mercado que utilizam programação *over-the-air* em redes de sensores sem fios. A Libelium [16] é uma empresa que projeta e fabrica plataformas e kits de desenvolvimento de software (SDK) usando as WSN em conjunto com o OTA, para que empresas de consultoria, engenharia e sistemas integrados possam entregar soluções viáveis para o mercado das IoT. Podem-se encontrar diversos produtos com características e necessidades específicas, alguns destes produtos são o *Waspmote kit*, o *Waspmote Plug & Sense!* e o *Meshlium*.

O *Waspmote kit* é uma placa de desenvolvimento onde o utilizador tem acesso a todos os componentes de hardware e pode manipula-los, adicionando novos tipos de sensores ou embutir nos projetos pessoais. Esta placa contém uma variedade de protocolos de comunicação como Zigbee, WiFi, GSM, GPRS, 3G, 4G, Bluetooth e RFID. Todos estes protocolos de comunicação são suportados, necessitando apenas de adquirir o módulo de comunicação e inserir na placa, porém apenas os módulos 4G, 3G, GPRS e WiFi são capazes de utilizar o OTA como meio de reprogramação sem fios. Contém ainda duas camadas de criptografia para segurança da comunicação que é transmitida pela rede. Uma vez que este produto é uma placa de desenvolvimento, encontra-se preparada para suportar diversos tipos de sensores e módulos de comunicação industriais como RS-485 e RS-232.

Por outro lado, o *Waspmote Plug & Sense!* foi projetado para facilitar a implementação e respetiva manutenção, possibilitando que o utilizador ignore as componentes eletrónicas e se foque nos serviços e nas aplicações que possam ser desenvolvidas. O dispositivo contém um invólucro robusto impermeável com conectores externos para conectar sensores, um painel solar para recarregar o dispositivo e aumentar a autonomia, uma antena de comunicação e uma porta *Universal Serial Bus* (USB) para reprogramação.

Por último o *Meshlium* é um gateway IoT que tem capacidade de incluir um módulo de Bluetooth e WiFi para procura de aplicações, contém ainda a capacidade para quatro diferentes tipos de comunicações radio: WiFi 2.4 GHz, 4G/3G/GPRS/GSM e Zigbee.

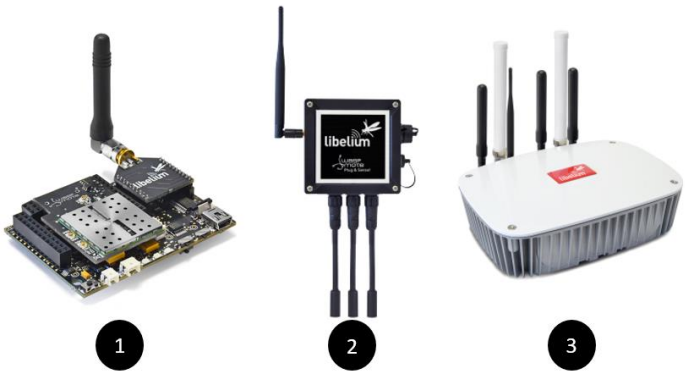

Figura 2.2 - Produtos comercializado por Libelium: 1 - Waspmote kit; 2 - Waspmote Plug & Sense!; 3 – Meshlium.

2.1.4 Interface para a indústria 4.0 permitindo, remotamente, a reconfiguração dinâmica

Uma vez que nos últimos anos têm aparecido diversas tecnologias, como sensores e atuadores inteligentes, no mundo industrial, promovendo a evolução da indústria 4.0, revelou-se necessário o desenvolvimento de interligações entre estes componentes, os sistemas e a informação relevante para o ser humano. Nesta subsecção é apresentado um artigo [17] publicado por S. Bougouffa, K. Meßmer, S. Cha, E. Trunzer e B. Vogel-Heuser, onde se propõe um conceito para permitir o acesso remoto a sistemas de produção automatizada pelo uso de uma interface com o intuito de permitir uma reconfiguração dinâmica por meio de serviços web.

Este conceito pretende a implementação de código para PLC diretamente nos *Programmable Logic Controllers* (PLC) [5], componentes responsáveis pelo controlo de sensores e atuadores de forma a desempenhar uma tarefa no sistema de produção. Para tal é necessário a implementação de um sistema *middleware* contendo uma interface responsável por transmitir e controlar o controlador em execução no PLC, bem como uma interface para a ligação de um dispositivo externo, de forma a realizar as reconfigurações e as reprogramações desejadas. Na Figura 2.3 encontra-se uma ilustração da visão geral deste conceito.

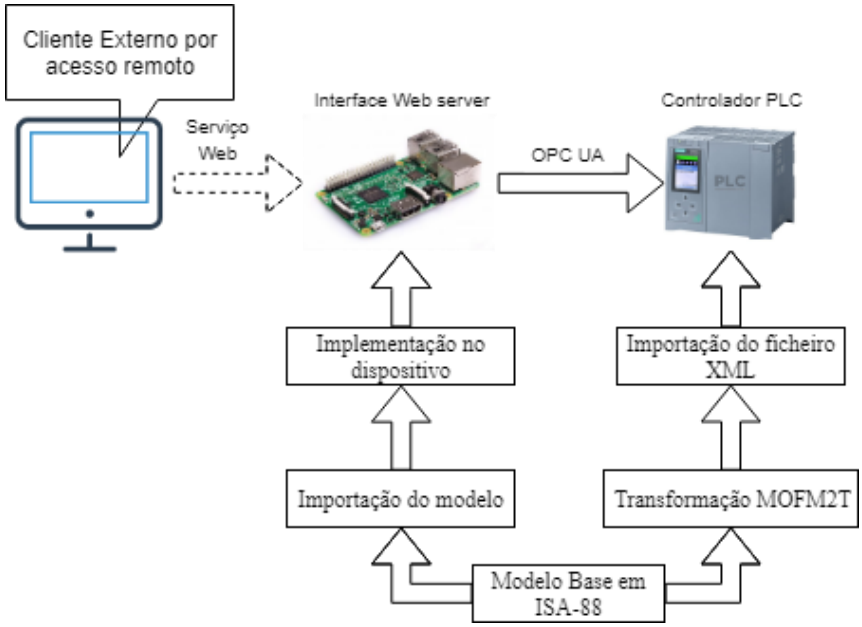

Figura 2.3 – Conceito base adaptado de [17].

Esta abordagem permite a reconfiguração do modelo base existente na fábrica possibilitando modificações no modelo e adição de extensões. Estas características facilitam a abordagem de técnicos de manutenção permitindo realizar alterações e ajustes apenas no modelo sem necessidade de deslocação.

O conceito persente no artigo apresenta pontos de contacto com a proposta apresentada neste documento, nomeadamente: o acesso remoto a um servidor com interface web; o visionamento do estado atual do modelo que se encontra em execução; a possibilidade de modificar o comportamento do modelo e ordenar a sua implementação; a reprogramação de controladores autonomamente. No entanto existem diferenças entre ambas as abordagens, nesta dissertação pretende-se: uma comunicação com sistemas externos sem intervenção humana, promovendo a comunicação entre sistemas; o *debug* de modelos em tempo de execução; o controlo remoto do estado de execução do controlador (iniciado, parado, pausado); a obtenção de informação da plataforma de execução e transmissão da mesma para sistemas externos.

Reconfiguração dinâmica num sistema de produção automatizada

Neste conceito o primeiro passo para alcançar a reconfiguração dinâmica é o desenvolvimento de um modelo base com uma descrição específica e única do modelo. O uso de *Meta Object Facility* (MOF) [18] permite definir a estrutura da arquitetura de forma a que cada elemento no modelo corresponda a uma camada no modelo. Esta estrutura do modelo é baseada no padrão ISA88 [19] para a modelação do meta-modelo, e é utilizada a ferramenta *Eclipse Modeling Framework* (EMF) [20] para a implementação do editor.

Após a modelação do modelo base, usando o editor ISA88, é possível gerar o código para o controlador PLC usando o MOF. Devido ao uso do MOF o modelo transferido encontra-se em versão de texto, nomeadamente PLCopen XML [21], permitindo uma reconfiguração dinâmica do código base do modelo sempre que este é carregado no PLC.

Interface para indústria 4.0

Este conceito pretende a implementação de fórmulas dinâmicas que permitem um planeamento flexível de processos usando o servidor *Open Platform Communications Unified Architecture* (OPC UA) [22] juntamente com serviços web no middleware. O uso destas duas tecnologias permite ao middleware aceder através da rede ao PLC, obtendo informação do sistema e usando a interface OPC UA para controlar o PLC com instruções para iniciar, pausar, reiniciar e parar operações. O middleware comunica com o PLC sendo um cliente para ler e escrever nas variáveis, para além da comunicação com PLC este dispositivo guarda os eventos ocorridos e apresenta uma interface gráfica.

Esta interface é o meio de comunicação com utilizadores externos e o PLC, com métodos/funções para *execução*, *histórico* e *instância*, usando pedidos *HyperText Transfer Protocol* (HTTP) com os parâmetros necessários.

O método de *execução* desencadeia funções de controlo no sistema de produção automatizada com dois tipos de modos, manual e automática. No modo manual somente uma instrução é invocada enquanto que no modo automático um conjunto de instruções são realizadas sequencialmente, sendo apenas necessário invocar a instrução principal porque o middleware se responsabiliza pela execução de todas as instruções associadas.

O método de *histórico* apresenta ao utilizador o estado do sistema no presente e no passado à realização do pedido. Por último, o método de *instância* devolve ao utilizador informação geral sobre o sistema, configurações e serviços disponíveis.

2.1.5 Laboratório remoto

Com o desenvolvimento de novos programas, a necessidade de testar os mesmos no mundo real revelou-se um problema logístico. Este problema deve-se ao elevado número de testes a efetuar em comparação com o reduzido número de plataformas e tempo disponível. Para resolver este problema surge o conceito de sistema denominado por Laboratório remoto, que possibilita a validação e visualização, no mundo real, dos modelos desenvolvidos.

Atualmente existem diversos laboratórios com estas capacidades. Um exemplo de implementação deste conceito é o *NetLab* [23].Este sistema foi desenvolvido para os estudantes poderem testar os seus projetos proporcionando análises de dados experimentais e de desempenho no

mundo real. Os utilizadores têm acesso a uma interface *web* baseada em *LabVIEW* [24] em conjunto com um *Embedded Web Server* (EWS) que por sua vez é responsável pela programação de microcontroladores para a execução de testes.

Para interagir com o laboratório remoto, os utilizadores acedem remotamente ao servidor web, através de um computador com acesso à internet. O servidor web que estabelece a conexão ao laboratório que tem um módulo baseado em EWS, microcontroladores, sensores e actuadores (Figura 2.4).

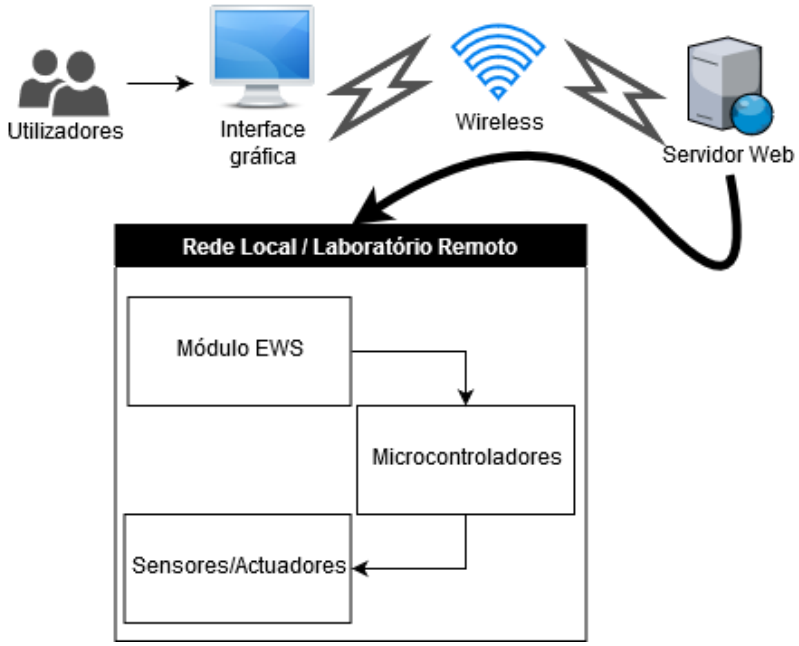

Figura 2.4 - Arquitetura do laboratório remoto. Adaptado de [25].

Em [25] é apresentado um exemplo de aplicação do EWS e um módulo de experimentação para a validação do *NetLab*. É utilizado um Raspberry Pi contendo um servidor web, o Linux Arduino IDE e ainda a plataforma Arduino Uno, que por sua vez executa o controlador recebido.

Para realizar a programação do sistema é transferido um ficheiro *hex* desde a interface gráfica em *LabVIEW* para o Raspberry Pi, que usando a ferramenta Linux Arduino IDE transforma o ficheiro *hex* num executável reconhecido pelo Arduino Uno para a respetiva programação. (Figura 2.5)

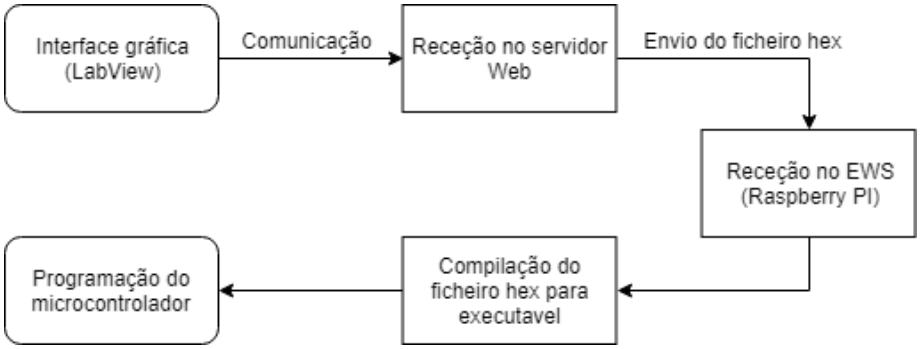

Figura 2.5 - Diagrama de fluxo para a programação do microcontrolador.

Do ponto de vista do utilizador o procedimento para a execução do controlador desenvolvido necessita somente de credenciais reconhecidas pela base de dados e a colocação do modelo para execução. Após este passo o servidor encarrega-se de realizar a transmissão entre sistemas e apresentar a emissão vídeo em tempo real do controlador em estado de execução.

2.2 Formalismos de modelação e ferramentas IOPT

2.2.1 Rede de Petri

A rede de Petri (RdP), apresentada por Carl Adam Petri em 1962, é um formalismo de modelação matemático e gráfico que suporta a especificação de sistemas simples ou complexos, através de modelos que fornecem a visualização explícita de concorrência, conflitos, exclusão mútua e sincronização.

Uma rede de Petri é um grafo composto por dois tipos de nós (lugares e transições) ligados por arcos, como ilustrado na Figura 2.6. Os lugares, representados geralmente por uma circunferência, contêm o número de marcas de um dado lugar e correspondem a um estado do sistema. As transições, representadas por uma barra ou quadrado, contêm a função de disparo e após a execução da função é realizada a destruição das marcas no lugar de origem e criadas uma nova marca no lugar de destino. Por fim, os arcos realizam a ligação entre os lugares e as transições. Um arco não pode ligar entre dois nós do mesmo tipo (dois lugares ou duas transições). As RdP são frequentemente utilizadas na modelação, simulação e verificação de sistemas, no entanto também podem ser usadas para suportar a implementação de sistemas, como é o caso das redes de Petri IOPT.

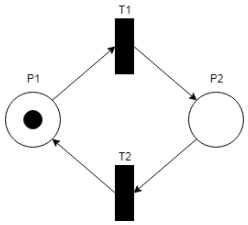

Figura 2.6 - Exemplo de uma rede de Petri

2.2.2 IOPT-nets

As IOPT-nets (Input-Output Place-Transition nets) [26] estendem a RdP lugar-transição com um conjunto de características para suportar a interação entre o controlador e o ambiente. Esta extensão em conjunto com uma semântica de execução bem definida permite especificar sistemas de automação e sistemas embutidos, bem como sistemas distribuídos globalmente-assíncronos localmente-síncronos (GALS) [27].

As novas extensões desenvolvidas proporcionam as seguintes características:

- As entradas (sinais e eventos) do sistema podem ser associadas às transições;
- As saídas (sinais e eventos) do sistema podem ser associadas às transições e aos lugares;
- Definição das prioridades nas transições;
- Os lugares possuem um atributo que indica o número máximo de marcas;
- Dois tipos de valores para os sinais de entrada e de saída (inteiro e booleano);
- Uma especificação explicita para um conjunto de transições com conflitos;
- Uma especificação explicita para um conjunto de transições síncronas;
- Arcos de teste.

As IOPT-nets permitem a sua especificação em expressões algébricas, variáveis, e funções nas transições (denominadas guardas), bem como condições nas ações de entrada associadas a cada lugar da RdP.

Uma rede IOPT pode ser definida por:

$$N = (P, T, A, TA, M , weight, weightTest, priority, isg, ie, oe, oce)$$

Onde:

- P é um conjunto finito de lugares;
- T é um conjunto finito de transições (disjunto de P);
- A é um conjunto de arcos, tal que $A \subseteq ((P \times T) \cup (T \times P))$;
- TA é um conjunto de arcos de teste, tal que $TA \subseteq (P \times T)$;
- M é a função de marcação: $M: P \rightarrow N_0$;
- Peso dos arcos, weight: $A \rightarrow N_0$;
- Peso dos arcos de teste, weightTest: $TA \rightarrow N_0$;

- A prioridade (priority) é uma função parcial que aplica transições a valores inteiros maiores que zero, priority: $T \rightarrow N_0$;
- isg é uma função parcial de guarda de sinal de entrada (IS - um conjunto finito de sinais de entrada) que aplica transições a expressões booleanas: $isg: T \rightarrow BE$, onde $\forall eb \in isg(T)$, $Var(eb) \subseteq IS$. BE é o conjunto de expressões booleanas. Var(E) retorna o conjunto de variáveis numa dada expressão E;
- ie é uma função parcial de eventos de entrada que aplica transições a eventos de entrada (IE - um conjunto finito de eventos de entrada):

  $ee: T \rightarrow IE;$

- oe é uma função parcial de eventos de saída que aplica transições a eventos de saída (OE - um conjunto finito de eventos de saída):

  $ee: T \rightarrow OE;$

- osc é uma função para um sinal de saída (onde OS é um conjunto finito de sinais de saída), de lugares para conjuntos de regras:

$osc: P \rightarrow P(RULES),$ onde $RULES \subseteq (BES \times OS \times N_0)$, $BES \subseteq BE$ e $\forall e \in BES, Var(e) \subseteq ML$, ML é o conjunto de identificadores para cada marcação de lugar, após um estado de execução cada marcação de lugar tem associado um identificador, que é usado quando o código gerado é executado.

A semântica de execução das IOPT-nets refere que uma transição dispara quando a mesma se encontra habilitada e a condição de entrada externa é verdadeira, ou seja, tanto o evento de entrada como a função de disparo necessitam de ser verdadeiros.

O uso de arcos de teste e prioridades associadas às transições proporcionam resoluções de conflitos na modelação do controlador.

As IOPT-nets são suportadas por dois ambientes de desenvolvimento de automatização de projeto, as IOPT-Tools e IOPT-Flow, ambos disponíveis e de acesso livre.

2.2.3 Documento PNML

A *Petri Net Markup Language* (PNML) é uma linguagem de descrição de RdP baseada em XML (eXtensible Markup Language) que especifica a sintaxe/características dos diferentes atributos da RdP através de etiquetas. O PNML rege-se por três princípios: legibilidade, universalidade e mutualidade [28]:

- A **legibilidade** define que o documento seja legível e editável tanto por um humano como por uma máquina usando um editor de texto convencional. Isto é garantido pelo uso de XML.
- A **universalidade** permite abranger qualquer versão de RdP independentemente da versão ou extensão usada.

- A **mutualidade**, permite extrair o máximo de informação da RdP, usando uma convenção que define o conjunto de etiquetas, independentemente de a rede ser conhecida ou desconhecida.

2.2.4 IOPT-Tools

As IOPT-Tools [29] são um conjunto de ferramentas interativas num ambiente de desenvolvimento gráfico (Figura 2.7). Estas ferramentas possibilitam a edição de modelos IOPT, a simulação e verificação de propriedades destes modelos, através da geração e análise do espaço de estados do sistema. Permite ainda a implementação de controladores embutidos através da geração automática de código.

AS IOPT-Tools dispõem de geradores automáticos de código necessários para a implementação de controladores em sistemas embutidos, possibilitando a geração de código C [30], de código de descrição de hardware em VHDL [31], código IL [32], *JavaScript* e de *Simulink System Block*. Possui ainda uma ferramenta de *debugger* com acesso remoto, via HTTP, ao hardware do controlador onde se encontra em execução o código, previamente, gerado automaticamente, capaz de afetar e verificar as saídas/entradas/variáveis/registos do sistema em tempo real.

Destaca-se deste modo as características principais das IOPT-Tools:

- Interface gráfica da edição de modelos RdP IOPT;
- Validação/simulação de modelos;
- Monitorização remota do sistema em tempo de execução;
- Geração automática de código.

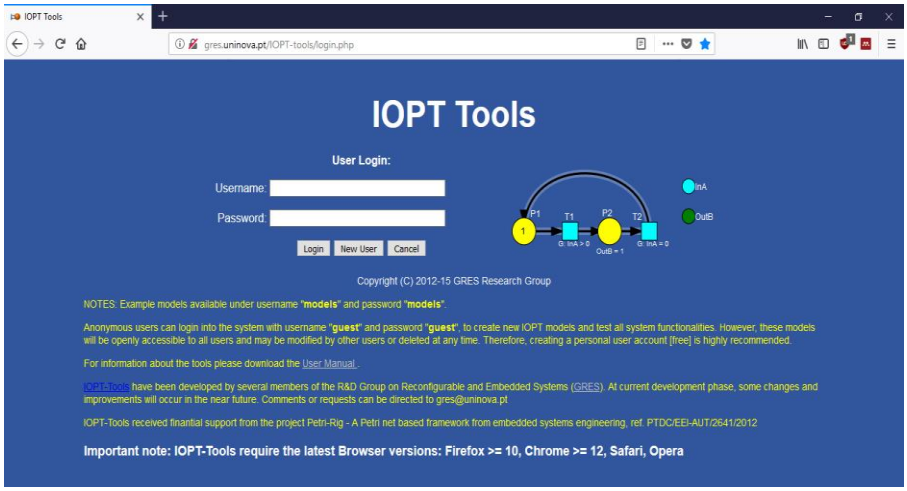

Figura 2.7 - Website da ferramenta IOPT-Tools desenvolvida pelo GRES.

Estas ferramentas e o respetivo manual [33], encontram-se disponíveis gratuitamente no website do *Group on Reconfigurable and Embedded Systems* (GRES): *http://gres.uninova.pt/IOPT-tools/login.php.*

2.2.4.1 Interface gráfica da edição de modelos RdP IOPT

Uma vez que as IOPT-Tools suportam uma extensão da classe de redes de Petri Lugar-Transição, IOPT-nets, o ambiente gráfico de edição contém um painel para o desenvolvimento do modelo IOPT e um caixa de ferramentas contendo o design não só para lugares, marcas, transições e arcos, mas também blocos para os sinais de entrada e saída, blocos para os eventos de entrada e saída e os arcos de teste, entre outros, como ilustrado na Figura 2.8.

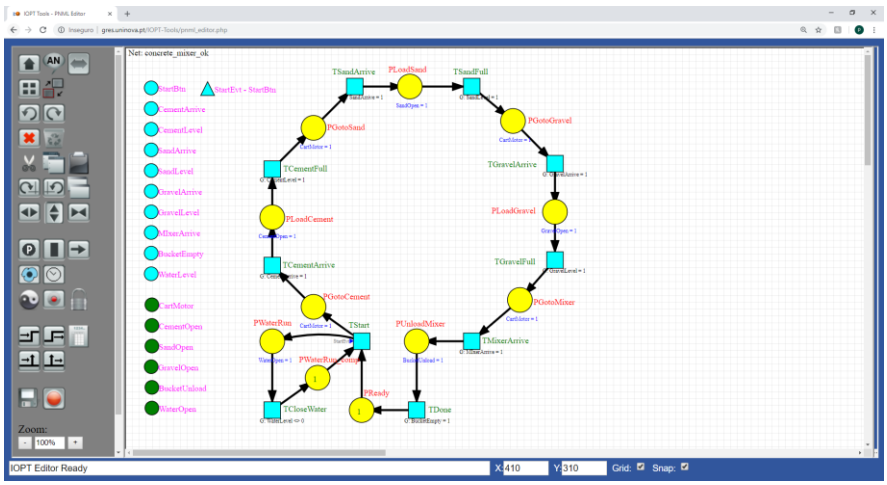

Figura 2.8 - Editor gráfico das IOTP-Tools.

Após adicionar o sinal ao modelo IOPT são apresentadas as propriedades inerentes ao sinal, nomeadamente, o nome do sinal, o tipo de valor (booleano ou inteiro), o valor mínimo e máximo

e por último o número da porta física da qual será realizada a leitura ou afetação do sinal na plataforma de desenvolvimento (Ver Figura 2.9). Propriedades semelhantes são apresentadas quando é escolhido um evento (Ver Figura 2.10). Um evento encontra-se diretamente relacionado com um sinal e este ocorre quando ocorre uma determinada alteração ao sinal.

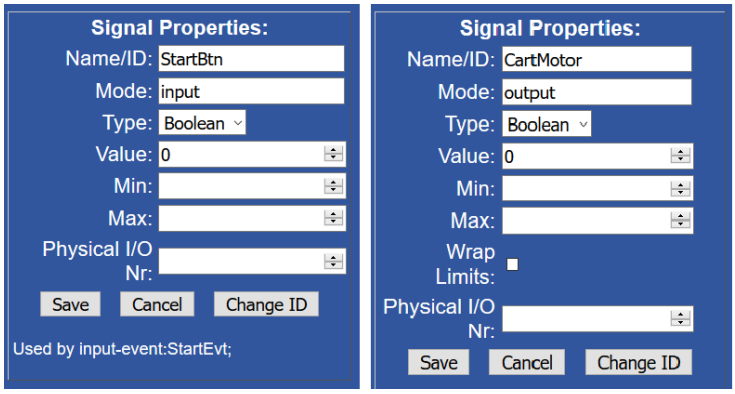

Figura 2.9 - Editor de propriedades de sinais de entrada e saída.

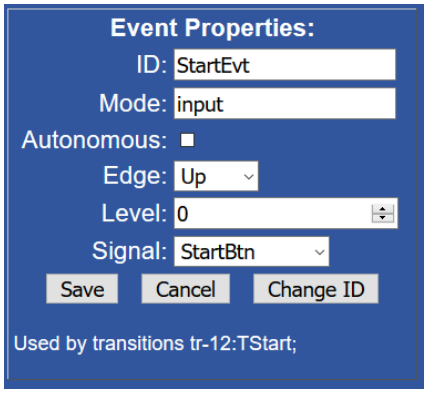

Figura 2.10 - Editor de propriedades de eventos de entrada e saída.

2.2.4.2 Validação/simulação de modelos

As IOPT-Tools possuem uma ferramenta de simulação e uma ferramenta de geração de espaço de estados de forma a verificar/confirmar o esperado e correto funcionamento do controlador sem a necessidade de usar plataformas de teste físicas.

A ferramenta de simulação (Ver Figura 2.11) é composta por uma coluna de controlos, na lateral esquerda, que contêm as funcionalidades de: iniciar, parar, verificar passo a passo a execução do controlador e uma funcionalidade de extração do histórico de execução do controlador, bem como a visualização a mesma nos instantes de tempo anteriores. No centro da ferramenta é apresentado o modelo IOPT, utilizando a cor vermelha para os lugares e laranja para as transições torna-se fácil a visualização das transições habilitadas e dos lugares que as habilitam. Por fim, na

lateral direita são apresentados todos os lugares, sinais e eventos bem como os respetivos valores para cada instante de tempo durante a execução da simulação do modelo.

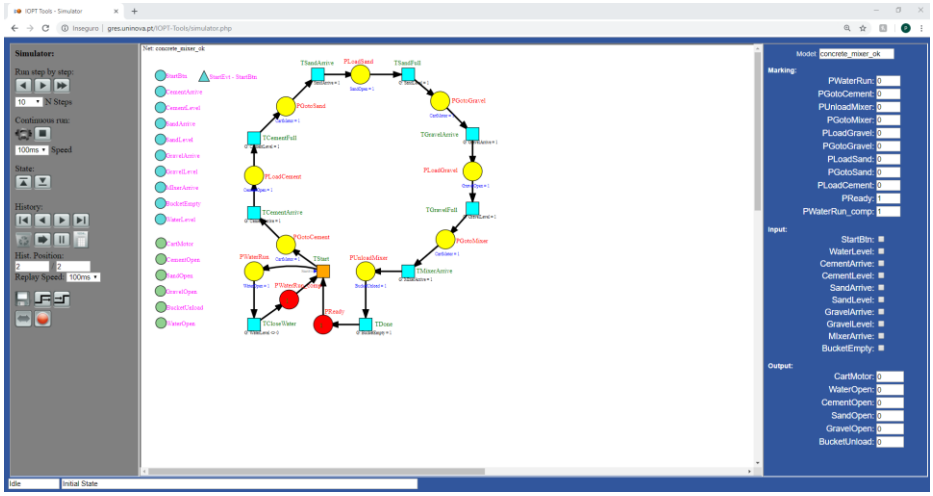

Figura 2.11 - Simulador gráfico das IOPT-Tools.

A ferramenta de geração de espaço de estados (Ver Figura 2.12), permite aos utilizadores verificarem o funcionamento do modelo IOPT para todas as possíveis de sequências de estados do controlador. A partir das marcas iniciais em cada lugar é validada a possibilidade de existência de deadlocks (um estado de bloqueio do controlador), conflitos e estados inválidos, ilustrando textualmente, Figura 2.13, como também graficamente, Figura 2.14.

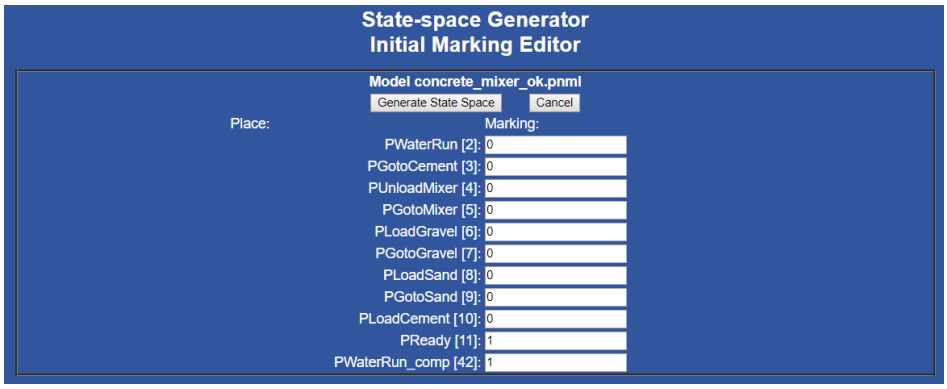

Figura 2.12 - Ferramenta para a geração de espaço de estados embutida nas IOPT-Tools.

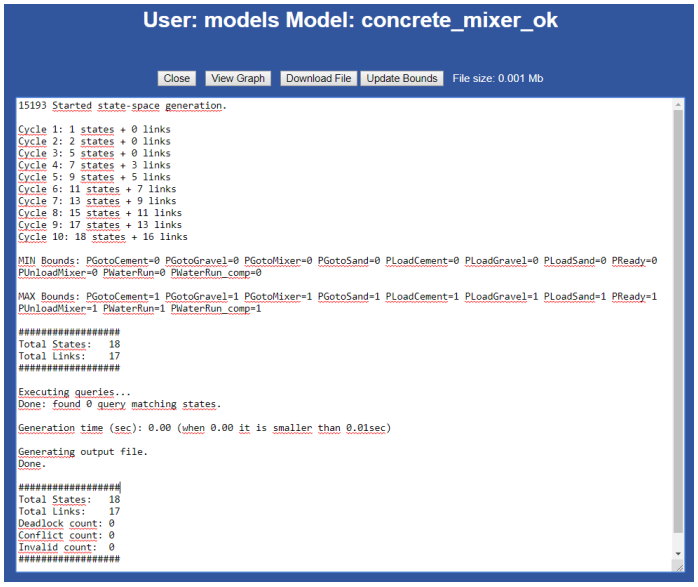

Figura 2.13 - Resposta há saída do gerador do espaço de estados.

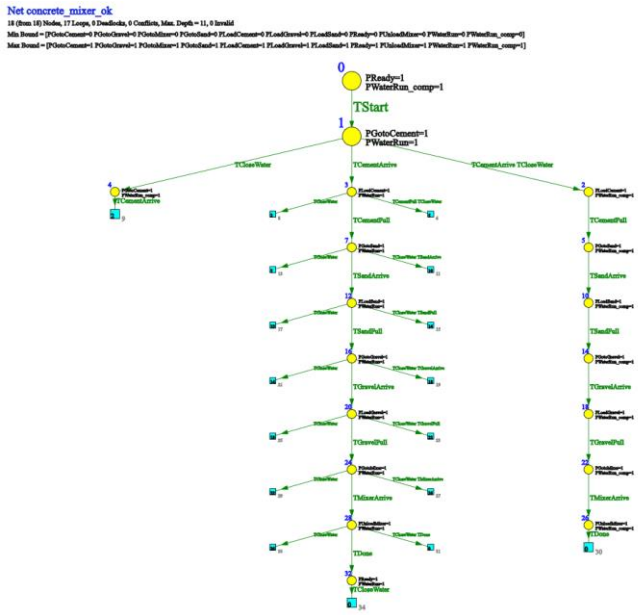

Figura 2.14 - Geração gráfica do espaço de estados do modelo IOPT dado as marcas iniciais da rede.

Esta ferramenta permite ainda a edição de questões e a sua integração no código de validação automaticamente gerado. Permitindo assim a execução de testes mais detalhados ao modelo IOPT e proporcionando uma diversidade de verificações e obtenção de resultados mais específicos para o sistema desenvolvido.

2.2.4.3 Monitorização remota do sistema em tempo de execução

Como mencionado anteriormente, as IOPT-Tools possuem uma ferramenta de monitorização e controlo remoto de controladores em tempo de execução, denominada *remote debugger*.

Esta ferramenta permite a visualização do modelo IOPT (utilizando uma interface semelhante à ferramenta de simulação, Figura 2.11), dos respetivos dos estados, bem como o valor dos sinais de entrada e saída durante a execução na plataforma.

Estas características são possíveis de visualizar e controlar porque à ferramenta de geração de código ao realizar a tradução de RdP IOPT para a linguagem C, anexa ficheiros para a criação de um servidor web e respetivas funções, para a monitorização e o controlo dos controladores por HTTP. Para tal a plataforma de implementação necessita de conter um módulo para a conexão à internet. Utilizando um computador conectado na mesma rede é possível aceder ao controlador por meio remoto.

2.2.4.4 Geração automática de código

Os geradores de código permitem que os modelos desenvolvidos com as RdP IOPT sejam implementados em diversas plataformas possibilitando ao desenvolvedor uma vasta gama de plataformas de teste/execução.

Particularmente para o gerador de código *C* são gerados ficheiros que permitem a implementação tanto em plataformas baseadas em sistemas operativos Linux ou em plataformas da família Arduino. Na Tabela 2.1 são elencadas as descrições de cada documento gerado pela ferramenta, até à data da presente dissertação.

Os documentos gerados possuem características que permitem a comunicação HTTP e a definição das portas de entrada e saída, bem como funções de leitura e escrita nas mesmas, que serão utilizadas pelo modelo. Estas características são essenciais para o funcionamento do sistema de gestão que se pertende desenvolver, permitindo a gerir a execução e os valores dos sinais de entrada do sistema.

Tabela 2.1 - Descrição dos documentos gerados automaticamente pela ferramenta Gerador de código *C*.

Nome do ficheiro	**Descrição**
net_main.c	Este documento contém a definição do ciclo de execução do modelo IOPT.
net_function.c	Neste documento são declaradas as funções para implementação das semânticas e regras da rede de Petri IOPT.
net_types.h	O documento contém a definição dos tipos de dados para a marcação de cada lugar, sinais de entrada ou saída e eventos. Declaração dos métodos associados ao comportamento da rede.
net_server.c	O documento possuí funções para a associação do modelo IOPT com o Ethernet Shield do Arduino para a implementação do protocolo HTTP.
net_server.h	O documento define as funções da biblioteca para o documento net_server.c, onde se encontram declaradas as funções para implementação do servidor HTTP apenas para plataformas Arduino.
net_io.c	O documento contém a definição de quais as entradas e saídas que irão ser utilizadas pelo controlador.
net_dbginfo.c	O documento permite a realização de *debug* da rede, forçar valores nas entradas e obter informação sobre os sinais de entrada e saída, marcas e disparos de transições.
net_exec_step.c	Neste documento encontram-se todas as transições utilizadas no desenvolvimento do modelo IOPT, bem como o local da realização do disparo da transição.
http_server.c	O documento é constituído pela implementação de uma comunicação, monitorização do servidor e HTTP *debug*.
http_server.h	O documento define as funções da biblioteca para o documento http_server.c, são declaradas as funções de *debug* e definida a estrutura do argumento do pedido de *debug*.
linux_sys_gpio.c	O documento permite a utilização das portas físicas associadas ao controlador em sistemas Linux. Sendo implementadas as funções para definição das portas de entrada ou saída e respetivamente a escrita e leitura de sinais.
raspi_mmap_gpio.c	O documento permite a leitura e escrita nas portas de física nas plataformas Raspberry Pi de qualquer geração, sendo mais rápida a utilização deste em relação ao uso do documento linux_sys_gpio.c.
dummy_gpio.c	O documento permite que os sinais de entrada sejam gerados a partir de um documento de texto (inputs.txt), e o registo das saídas sejam guardados para outro documento de texto (outputs.txt).
Makefile	A utilização deste documento é opcional. Construtor de projeto Gnu Make/Unix com instruções para construção do projeto.

2.2.5 DS-Pnets e IOPT-Flow

As *Data-flow, Signals and Petri nets* (DS-Pnets) são um formalismo de modelação desenhado para suportar o desenvolvimento de sistemas ciber-físicos. Combinando RdP e *dataflows*, possibilitam a modelação de sistemas mistos, contendo partes reativas e operações de processamento de dados. Os nós *dataflow* permitem especificar as operações matemáticas e dependências entre sinais, substituindo as funções inseridas nos lugares e transições das IOPT-nets.

As DS-Pnets são suportadas pelas IOPT-Flow [34], que apresentam um conjunto de ferramentas semelhantes às IOPT-Tools, desde o editor de modelos até aos geradores de código. Porém estas ferramentas tendem a resolver duas limitações encontradas nas IOPT-Tools, nomeadamente a impossibilidade de implementar um novo modelo composto por outros componentes/módulos previamente implementados, de forma a reutilizar implementações de outrem e a opção de usar operações complexas para manipulação de dados/sinais.

Deste modo o formalismo das IOPT-Flow é um formalismo de modelação híbrido que combina as IOPT-nets com *dataflows*, possibilitando a especificação de sistemas complexos através do uso de componentes, sendo que a parte de controlo é habitualmente especificada com redes de Petri e a parte de dados com data-flows.

Em relação ao ambiente gráfico, este permite a edição de modelos DS-Pnets, anteriormente referidos. Consistem num grafo composto por cinco tipos diferentes: Lugares de RdP, Transições de RdP, Eventos de entrada e saída, Sinais de entrada e saída e por últimas, por operações de data-flow.

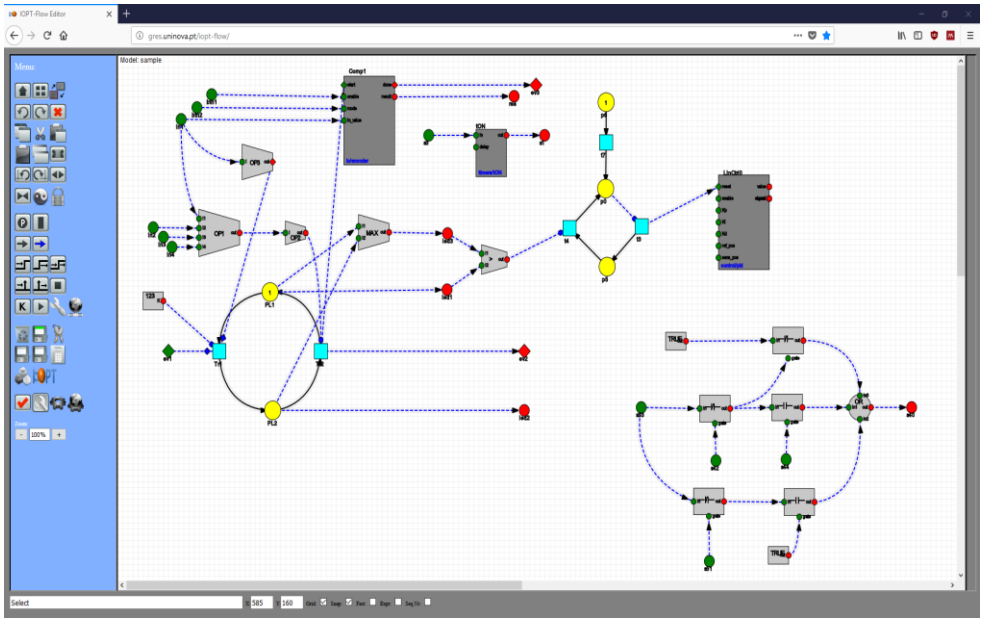

Figura 2.15 - Website da ferramenta IOPT-Flow desenvolvida pelo GRES.

Esta ferramenta, tal como as IOPT-Tools, também se encontra disponível gratuitamente no website do GRES: *http://gres.uninova.pt/iopt-flow/*.

2.2.6 Modelos de interpretação de sinais

As ferramentas IOPT permitem a modelar controladores para sistemas embutidos, analisando os valores das entradas e atuando nas saídas consoante o modelo projetado, porém, o desenvolvimento de sistemas mais complexos pode resultar em modelos muito grandes e, consequentemente, difíceis de interpretar. O uso de outros componentes ou formalismos que permitam a partição dos modelos desenvolvidos, adicionando pré e pós-processamento ao controlador IOPT permitem uma solução de modelação estruturada que permite obter modelos mais reduzidos.

Desta forma surgem os modelos de interpretação de sinais (MIS) [35] um formalismo que permite a modelação da interpretação de sinais de entrada, possibilitando um pré-processamento relativo à informação recebida do meio ambiente.

Este formalismo tem com objetivo modelar a aquisição e interpretação da informação que o sistema recebe. Para tal este modelo recebe sinais provenientes do ambiente externo e gera eventos associados à ocorrência de comportamentos dos sinais analisados.

Para uma melhor perceção do fluxo da informação no MIS e do seu resultado é possível observar na Figura 2.16 os passos de execução do modelo.

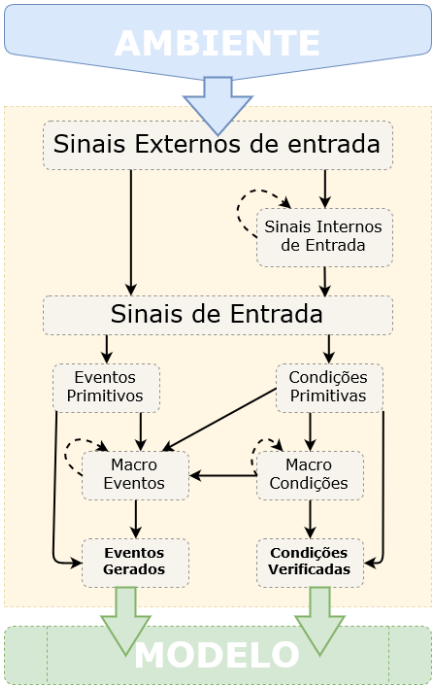

Figura 2.16 - Diagrama de execução do MIS. Adaptado de [35].

Na entrada do MIS, a informação é recebida através de sinais externos, que consistem na informação não tratada proveniente do ambiente, em seguida estes sinais seguem dois percursos com o propósito de se transformarem em sinais de entrada.

O primeiro percurso consiste numa passagem direta dos sinais externos em sinais de entrada, sem a necessidade de modificar a informação. No segundo percurso estes sinais passam por uma análise, gerando novos sinais, denominado sinais internos de entrada.

Estes novos sinais referem-se a sinais calculados através de funções dependendo de outros sinais, por exemplo conversões métricas, valores médios ou o cálculo da *Fast Fourier Transform* (FFT) de um sinal.

Em seguida os sinais internos de entrada são agrupados ao conjunto de sinais de entrada. Toda esta informação é analisada de forma a encontrar comportamentos elementares, tanto através de eventos como de condições previamente definidas.

Por fim os mesmos comportamentos elementares são analisados conjuntamente de forma a determinar comportamentos complexos. Sempre que cada um dos comportamentos que se pretende verificar acontece é gerado um evento para o exterior.

Este fluxo de informação, desde a receção dos sinais externos até aos eventos de saída, resulta num conhecimento mais profundo e diversificado, utilizando o mesmo número de sensores, reduzindo o custo de implementação.

Tal como existem atualmente ferramentas que suportam o formalismo IOPT-nets, mencionadas anteriormente, o formalismo MIS encontra-se inserido numa ferramenta similar as características existentes nas IOPT-Tools e IOPT-Flow.

3 Gestor de execução e reprogramação

Neste terceiro capítulo é proposto um sistema de gestão de execução e reprogramação de controladores. Após a introdução ao propósito deste sistema/gestor, canais de comunicação e requisitos, são expostos os requisitos mínimos que o presente sistema deve conter para o correto funcionamento e implementação.

3.1 Visão geral

Pretende-se desenvolver um sistema capaz de receber e executar diversas instruções autonomamente, sem intervenção humana, a partir da receção de uma mensagem de entrada, bem como decidir em que instante de tempo deve executar a instrução recebida.

Nesta perspetiva, o sistema tem que ser capaz de realizar um conjunto tarefas, de entre as quais se destacam a capacidade de iniciar, pausar, parar, reiniciar a execução e reprogramar o controlador, em execução, a partir de uma nova especificação (define-se a especificação como o conjunto de informação necessário para a implementação do controlador). Adicionalmente, este gestor deverá ter a capacidade de receber várias especificações e gerir a reprogramação e execução dos diferentes controladores associados, em tempo de execução, para o controlo de sistemas de produção automatizada.

Para alcançar as características mencionadas anteriormente o gestor necessita de possuir conhecimento sobre o sistema e requisitar informações durante a execução dos controladores, permitindo determinar o estado atual do sistema, nomeadamente o valor do sinal associado a cada porta de entrada e saída e a correspondência a cada controlador em execução.

Inicialmente definiu-se as interfaces de entrada e saída que o gestor/sistema terá ao seu dispor. Uma vez que se pretende controlar máquinas e sistemas de produção, é imprescindível

que o gestor tenha acesso às portas físicas das mesmas. Esta característica permite a obtenção de informação relevante e a alteração do valor à entrada de cada porta.

Em seguida destaca-se a comunicação com sistemas externos. Neste campo definem-se dois tipos de sistemas externos, i.e., utilizadores humanos e componentes externos. Os utilizadores humanos possuem um acesso ao gestor através de uma interface gráfica, tendo a capacidade de modificar, acionar, realizar *debug*, visualizar a especificação e monitorizar a execução em tempo real. Os componentes externos comunicam diretamente com o gestor transmitindo uma mensagem pré-definida que acionarão uma sequência de funções realizadas autonomamente pelo gestor. Este encarrega-se de responder, ao componente externo, a confirmação de sucesso ou falha da tarefa realizada. A Figura 3.1 apresenta as entradas e saídas mencionados anteriormente.

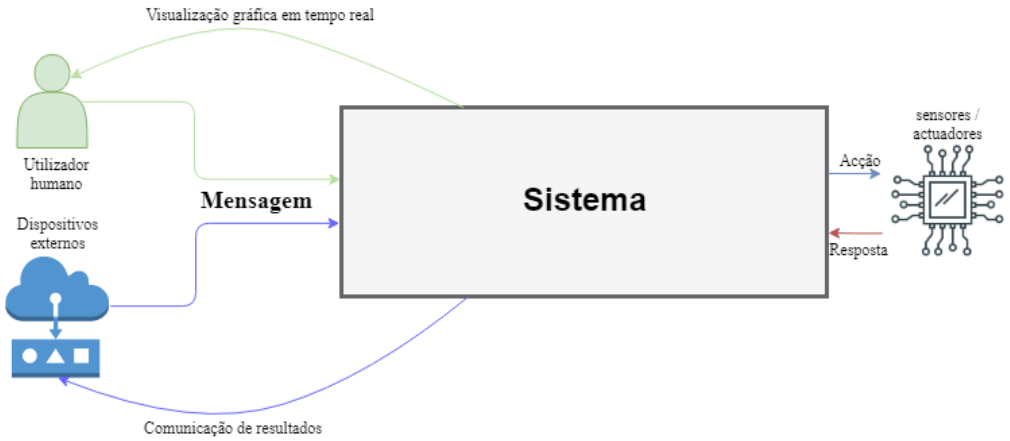

Figura 3.1 - Dinâmica de comunicação entre o sistemas externos, sistemas e sensores/actuadores.

3.2 Especificação do sistema/gestor

Após a definição das entradas e saídas do sistema, focamo-nos no núcleo do sistema. O núcleo é composto por dois módulos principais, o gestor e o(s) controlador(es) (Ver Figura 3.2). A comunicação entre os sistemas externos e o sistema é realizada no gestor de execução e reprogramação. Cabe a este módulo realizar o tratamento das mensagens recebidas e posteriormente comunicar com o(s) controlador(es).

A ligação entre o gestor e o(s) controlador(es) permite ao gestor executar diversas tarefas, nomeadamente iniciar, parar, pausar, reiniciar e programar/reprogramar cada um dos controladores em funcionamento. A comunicação entre o gestor e cada dos controladores poderá ser suportada por *pipes*, ficheiros, comunicação HTTP ou qualquer outro meio de comunicação que permita a transmissão da execução de tarefas para em cada controlador sem afetar os restantes.

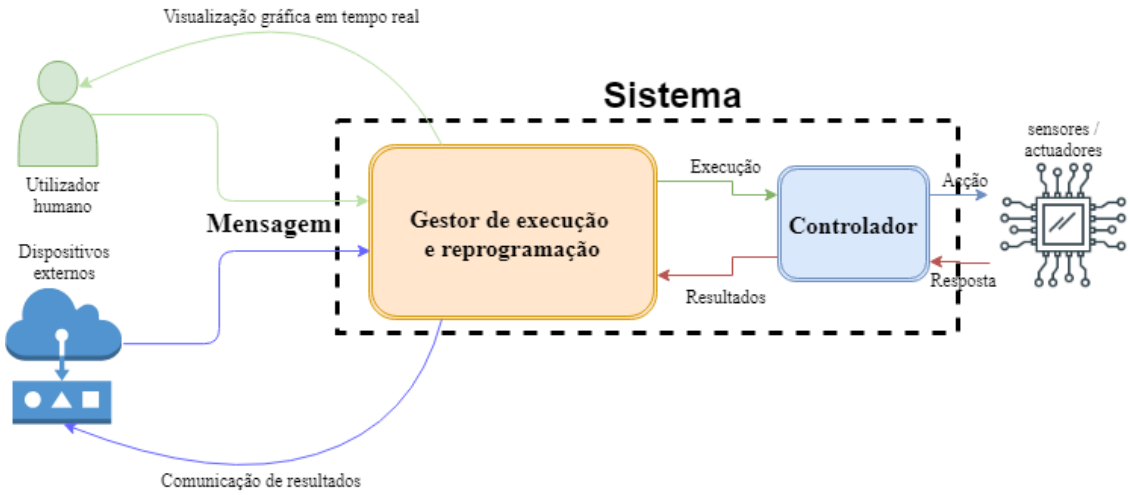

Figura 3.2 – Interação com o gestor de execução e reprogramação.

Requisitos do sistema

Para o sistema proposto nesta dissertação, foram estudados e estipulados determinados requisitos para o correto funcionamento do gestor de execução e reprogramação:

- A especificação proveniente dos componentes externos não pode conter erros de sintaxe ou deadlock;
- A plataforma para a execução do gestor necessita de possuir capacidade para a execução de um servidor web;
- A plataforma para a execução do gestor necessita de possui memória suficiente para comportar todas especificações recebidas e software em execução;

Os requisitos encontrados para a implementação do gestor permitem o desenvolvimento de uma interface para o utilizador, bem como serviços para a comunicação entre dispositivos externos e controladores.

A opção de receber a especificação e gerar o código posteriormente, ao invés de receber previamente o código gerado deve-se às vantagens que esta escolha acarreta. Desta forma é possível alterar o sistema e linguagem para a plataforma onde se pretende implementar sem necessidade de alterar o comportamento e transmissão dos sistemas externos. Outra vantagem é o uso de um ou vários tradutores/geradores para linguagens de programação diferentes, ou seja, a mesma especificação poder ser adaptada para duas ou mais plataformas distintas.

Porém o gerador escolhido para realizar esta função necessita de possuir as seguintes características:

- Conter a definição das entradas e saídas para possibilitar a implementação automática;
- Conter funções de leitura e escrita nas portas físicas da plataforma.

3.3 Interface do sistema/gestor

A interface de comunicação entre sistema/gestor e os outros sistemas é definida nesta secção. São apresentados os requisitos mínimos para as mensagens e respetivas funcionalidades para a comunicação entres sistemas externos e o gestor. Depois é abordado o relacionamento entre gestor e controladores.

Como exemplificada na Figura 3.3, a comunicação entre os dois sistemas inicia-se com um pedido de execução através do sistema externo para o gestor de execução e reprogramação. Este verifica o pedido e em caso de se validar a instrução é comunicado para o controlador qual a sequência de instruções a realizar, em caso do pedido ser invalido é encaminhado um erro (reconhecido) para o sistema externo. Por sua vez, o controlador valida as sequências de instruções requisitadas pelo gestor, em caso de sucesso é realizada a execução na plataforma e encaminhado a resposta ao pedido até ao sistema externo, se porventura alguma das instruções forem invalidas é remetido um erro de instrução até ao sistema externo.

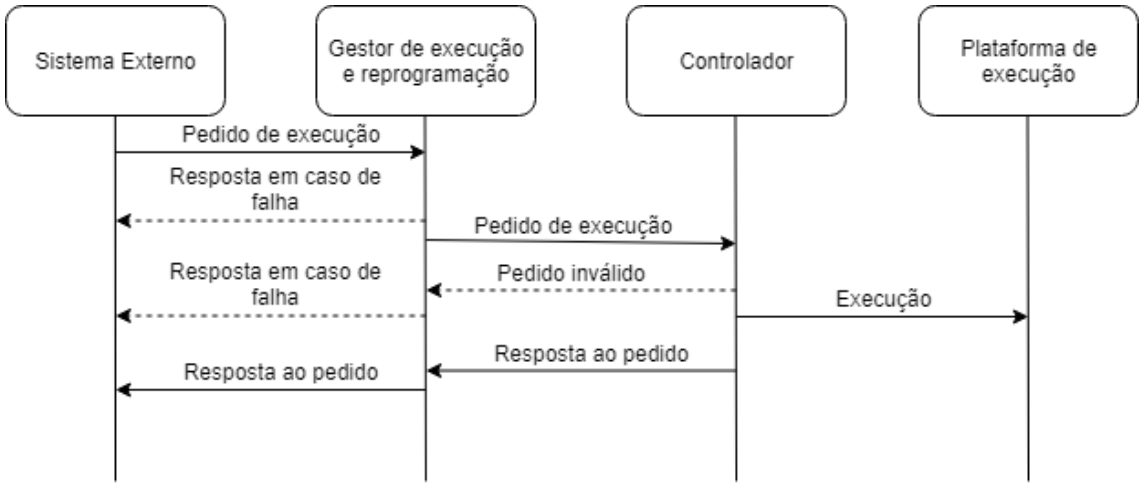

Figura 3.3 - Diagrama de comunicação entre componentes externas e controlador.

3.3.1 Tipos de mensagens

Nesta subsecção são apresentados os cinco tipos de mensagens que devem ser transmitidos pelos sistemas externos e reconhecidos pelo gestor, que fornece um conjunto de serviços web. Estes tipos de mensagens permitem a programação e a reprogramação de controladores, a monitorização dos controladores e o controlo do sistema de forma remota (Ver Tabela 3.1).

Tabela 3.1 - Lista dos cinco tipos de mensagens.

#	Categoria da mensagem	Instrução	Identificador	Complemento 1	Complemento 2
1	Programação	-	Nome do modelo	Especificação	Modo de prioridade
2	Ação	Iniciar	Nome do modelo		-
		Pausar	Nome do modelo		-
		Parar	Nome do modelo		-
		Reiniciar	Nome do modelo		-
3	Reprogramação	Actualizar	Nome do modelo	Especificação	Modo de prioridade
4	Implementação	Entrada	Nome do modelo	Nome do sinal	Valor
5	Obtenção	Entrada	Nome do modelo	Nome do sinal	
		Saída	Nome do modelo	Nome do sinal	

O primeiro tipo de mensagem possibilita o envio de novas especificações e programação de novos controladores, esta mensagem envolve a transmissão da especificação e do modo de prioridade. Este modo permite ao gestor conhecer a urgência da implementação da especificação, para tal foram estudados e estipulados os diferentes modos que o gestor deve de reconhecer, na Tabela 3.2 são elencados os diversos modos.

Tabela 3.2 – Catálogo dos modos de prioridade para a implementação de uma especificação.

Modo	Descrição
Segurança	Entrada imediata em execução e pára qualquer controlador que interfiram com o mesmo (por exemplo o uso das mesmas portas físicas de saída).
Alto	Em caso de uma fila de espera este controlador é prioritário, passando à frente de outros controladores na lista de espera.
Normal	Entra para a fila de espera e quando for possível é executado a sua configuração. (fica à espera que as portas físicas fiquem disponíveis, não sendo é descartado.)
Baixo	É realizado uma tentativa de (re)programação caso não haja fila de espera, em caso de existir alguma impossibilidade de realização da ação é descartado e remetida uma mensagem de falha ao originador

Caso a urgência não seja especificada (valor nulo), o gestor arbitra o modo como “Normal” colocando o modelo em lista de espera e quando for possível é executado a programação do modelo. No caso de o modo ser de “Segurança” é realizada a programação do modelo imediatamente, terminando a execução de outro(s) controlador(es) que interfira(m) com este modelo. O modo “Alto” coloca o modelo no início da fila de espera. Por fim o modo “Baixo” é realizada uma tentativa de programação do modelo e em caso de falha é descartado.

O segundo tipo de mensagem envolve o controlo da especificação com instruções de categoria básica designadamente iniciar, pausar, parar e reiniciar:

- Iniciar – Inicializa a execução da especificação caso não esteja inicializado.
- Pausar – Mantém o valor das portas físicas de entrada e de saída que estejam afetas à especificação selecionada e em seguida pára a execução. Ao iniciar a especificação os valores das respetivas portas são modificados para o estado anterior.
- Parar – Termina a especificação sem guardar o estado no qual se encontrava.
- Reiniciar – Realiza a ação de parar e iniciar consecutivamente voltando ao estado inicial do controlador.

O terceiro tipo destina-se à atualização/reprogramação de uma especificação que esteja previamente em estado de execução. Tal como na primeira mensagem, esta envolve a transmissão da especificação, o modo de implementação e adicionalmente o identificador da especificação, que se encontra em execução. O método para a realização da reprogramação consiste na execução de instruções de categoria básica, Parar, antes da programação, e Iniciar, após a programação da especificação.

A quarta mensagem possibilita a modificação do sinal de uma dada porta física de entrada, de forma a provocar uma alteração no comportamento da especificação durante o seu tempo de execução. Para isso necessita do identificador da especificação e do identificador e valor do sinal em causa.

Por último, a quinta mensagem destina-se à obtenção do valor do sinal durante o tempo de execução das especificações independentemente de se tratarem de sinais entradas ou saídas, sendo que esta última só carece do identificador da especificação e da identificação do sinal.

A mensagem transmitida do sistema externo para o gestor é composta por sete segmentos, Figura 3.4: Emissor, Autenticação, Categoria, Instrução, Identificador, Complemento 1 e Complemento 2. O Emissor é o identificador do originador da mensagem, transmitindo o seu endereço e porto. A Autenticação corresponde à informação referente ao utilizador, que transmite as credencias de acesso às IOPT-Tools, nomeadamente o nome de utilizador e password. A Categoria define o tipo de operação que o gestor irá efetuar na plataforma, para seguidamente validar a Instrução recebida. A Instrução consiste na ação que o gestor deverá realizar na plataforma e define o número de complementos a utilizar. Por fim o Identificador é o segmento que contém o identificador do modelo ao qual será executada a instrução requerida.

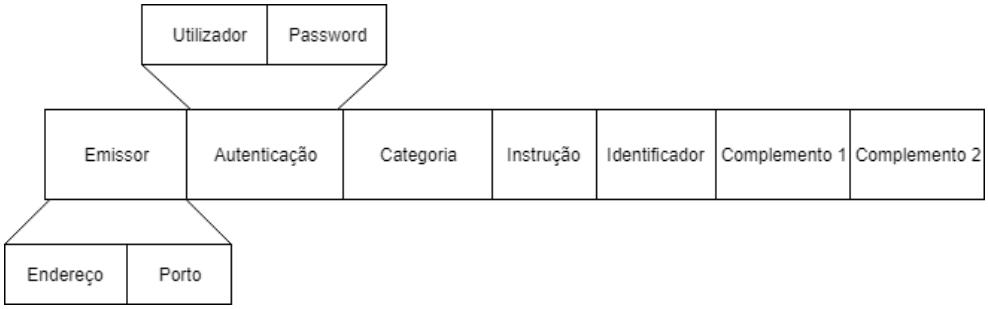

Figura 3.4 - Corpo da mensagem recebida pelo gestor de execução e reprogramação.

3.3.2 Interação com sensores e atuadores

Para o gestor possuir a capacidade de realizar leituras de sinais e alterar os valores dos sinais de entrada, necessita que o(s) controlador(es) em execução possuam determinadas características. Em primeiro lugar, o controlador necessita de ter a capacidade de alterar o comportamento dos sinais de entrada e saída sem a intervenção de outros sistemas/componentes. Em segundo lugar, o controlador necessita de possuir um método de comunicação integrada de forma a permitir que outros componentes enviem pedidos. Por último o controlador necessita de dispor de funções para receber pedidos de informação, obter/implementar a informação requisitada e responder emitente.

Por outro lado, o gestor necessita de ter um canal para a transmitir uma mensagem de resposta ao pedido do(s) sistema(s) externo(s) sobre o valor atual do sinal, para tal foi definida uma estrutura da mensagem a transmitir. A mensagem transmitida é composta pela identificação do controlador, pelo estado da porta física (entrada ou saída), pelo valor lido e o registo da data/hora no qual foi realizada a leitura. Deste modo outros componentes podem utilizar essa informação para uma melhor gestão de recursos. Todos estes dados podem ser futuramente aproveitados para a realização de estatísticas e logística.

3.4 Descrição do funcionamento do gestor

3.4.1 Programação e reprogramação de uma especificação

De forma a validar a coerência da especificação (o controlador), proveniente de um sistema externo (antes da sua implementação), é necessário realizar diversas verificações após a receção da mesma, como ilustrado na Figura 3.5.

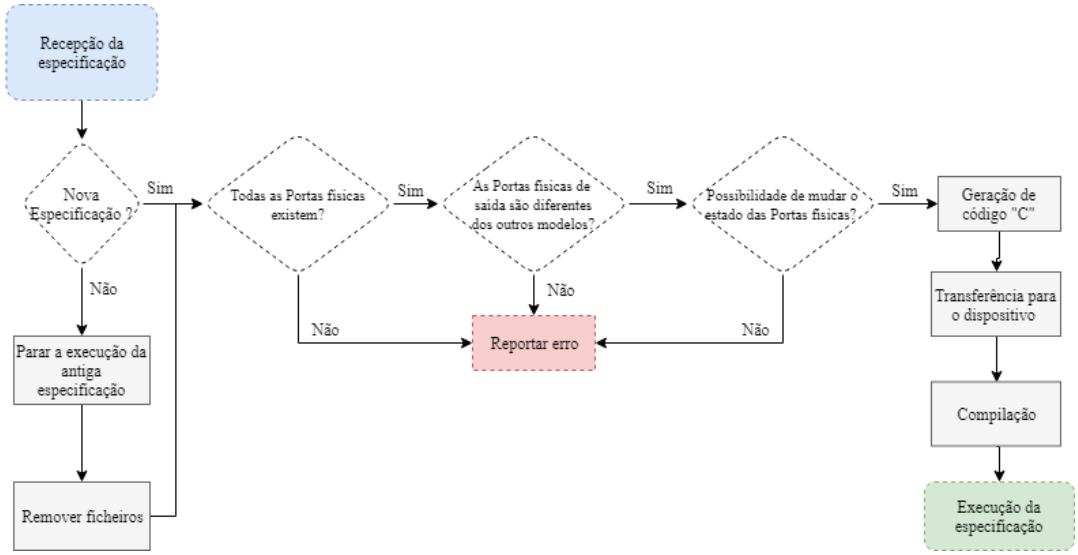

Figura 3.5 - Descrição do funcionamento do gestor desde da receção do modelo até à sua implementação.

A primeira verificação resulta do reconhecimento da especificação. Esta pode corresponder a uma nova especificação ou uma atualização de um controlador em execução. Para esta validação, cabe ao gestor conhecer qual o elemento de identificação para a sua distinção, comparado este com as restantes especificações. Em caso de a especificação ser identificada como nova, o gestor procede para os seguintes passos de verificação. Em caso de existir uma especificação com o mesmo identificador é realizada uma atualização, se o originador da instrução assim o desejar. Define-se por atualização o término do controlador em execução, a sua remoção, a geração do código do novo controlador, a compilação e execução. Antes da execução da atualização a especificação segue o mesmo percurso de verificação que uma nova especificação.

No segundo passo da verificação é necessário o levantamento das portas físicas de entrada e saída usadas. Este processo é efetuado de forma a impedir a especificação de requisitar portas físicas que não existam na plataforma, de afetar portas de alimentação (tensão de entrada e GND) ou ainda de impedir a afetação de portas físicas de saída (atuadores) já em uso por outras especificações em execução.

Para este gestor foi projetado o impedimento de dois ou mais controladores de afetarem as mesmas portas físicas de saída. Isto deve-se há possibilidade de ocorrência de falhas ou informação falsa quando os controladores não se encontram modelados corretamente. Por exemplo, se dois controladores distintos controlarem o movimento horizontal de um braço robótico (uma única porta física de saída). Admitindo que num dado instante de tempo de relógio o primeiro controlador instruir o movimento lateral no sentido esquerda-direita e no mesmo instante de tempo o segundo controlador instruir o movimento lateral no sentido direita-esquerda, poderá originar uma falha ou erro para o um dos controladores deixando o mesmo de conhecer o verdadeiro estado do sistema e potenciar informação falsa para o exterior.

Por fim, é verificada a necessidade de alterar o estado das portas físicas (de entrada para saída ou saída para entrada) utilizadas pela especificação, no caso de a especificação requisitar portas físicas é realizada uma comparação com as restantes especificações em execução para confirmar que a porta física possa ser modificada.

Sucintamente as verificações que são feitas resumem-se a garantir que:

- A especificação destina-se a uma atualização ou não.
- A especificação não pode usar pinos de alimentação ou que não existam.
- A especificação não pode alterar o estado de entrada/saída de portas utilizadas.
- Modelos diferentes não podem afetar as mesmas portas de saída (atuadores).

Após a receção da especificação e validação das condições mencionadas anteriormente é realizada a geração do código para a plataforma desejada, seguidamente é usado um compilador apropriado para a compilação do código sabendo a plataforma de implementação e colocado o controlador em execução, perfazendo a implementação na plataforma desejada.

3.4.2 Controlo de execução

Como abordado anteriormente, o gestor, para além da programação e reprogramação de especificações, permite ainda o controlo da execução das mesmas, em tempo de execução. Esta característica é alcançada devido às funcionalidades extras que os geradores de código automáticos possuem ao realizarem a geração de código de cada a especificação, nomeadamente: a capacidade de executar a leituras e escritas nas portas físicas e a integração de um protocolo de comunicação no controlador para a posterior comunicação com o gestor.

O pedido de execução é transmitido pelo sistema externo para o gestor, este após receber a informação sobre a identificação da especificação e instrução a executar, verifica se a especificação se encontra em execução na plataforma e em seguida executa a instrução, comunicando com o controlador e requisitando que seja aplicada/executada uma função necessária. Por fim o gestor devolve para o sistema externo a confirmação ou refutação do pedido.

4 Implementação do gestor

Para a validação do sistema proposto no capítulo 3, foi desenvolvido o gestor de execução e reprogramação embutido nas IOPT-Tools. Desta forma o gestor beneficia de todo o conjunto de ferramentas existentes nas IOPT-Tools, nomeadamente do gerador de código *C* e do *remote debugger*. Um protótipo do gestor desenvolvido está disponível online em http://gres.uninova.pt/~pmpr/iopt-tools.

4.1 Desenvolvimento da comunicação e o seu funcionamento

Como explicado no capítulo anterior a comunicação entre sistemas externos, gestor e controladores é centralizada no módulo gestor, que permite a comunicação entre os dois extremos.

O gerador de código *C* das IOPT-Tools, ao gerar o código de um controlador e a interface com as portas físicas da plataforma, prepara o mesmo, anexando um servidor HTTP e as respetivas funções para o tratamento de pedidos GET e enviar respostas com a informação requerida, utilizando pacotes JSON. Deste modo, para comunicar com os controladores, o gestor necessita de atuar com um cliente HTTP.

Para suportar a comunicação do gestor com diversos sistemas externos heterógenos, foi também escolhido o protocolo HTTP. O gestor possui um servidor HTTP para processar os pedidos enviados pelos sistemas externos, sendo utilizado o método POST para a transmissão de mensagens entre ambos. Este método permite o envio de ficheiros JSON e não possui limite de caracteres na mensagem (Ver Figura 4.1).

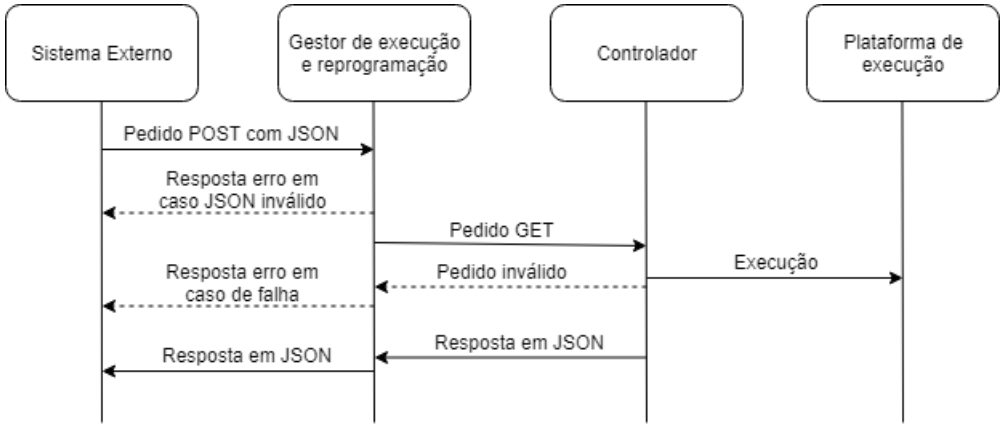

Figura 4.1 - Diagrama de comunicação entre sistemas externos, gestor e controlador usando método POST e GET.

São apresentados na Tabela 4.1 o conjunto de mensagens recebidas pelo gestor e a respetiva informação a transmitir em cada tópico.

Tabela 4.1 - Listagem do conteúdo das mensagens recebidas pelo gestor e respetivo método.

Método	**Categoria da mensagem**	**Instrução**	**Identificador**	**Complemento 1**	**Complemento 2**
POST	Ação	Iniciar	Nome do modelo	-	-
		Pausar	Nome do modelo	-	-
		Parar	Nome do modelo	-	-
		Reiniciar	Nome do modelo	-	-
	Reprogramação	Atualizar	Nome do modelo	Modo de prioridade	Modelo PNML
	Implementação	Entrada	Nome do modelo	Sinal	Valor
	Obtenção	Sinal	Nome do modelo	-	-
		Lista Sinais	Nome do modelo	-	-
		Controlador	Nome do modelo	-	-
		Lista Controladores	-	-	-

Usando JSON formulou-se um exemplo para a transmissão de um pedido para uma instrução (Figura 4.2) da *Categoria* "Ação" com o intuito de "Reiniciar" o controlador, denominado *blink_pi_linux*, baseado no corpo de mensagem da Figura 3.4.

```
{
   "sender": {"address":"127.0.0.1","port":"8454"},
   "authentication": {"user":"pmpr","pass":"pmpr2018"},
   "category": "action",
   "instruction": "restart",
   "modelId": "blink_pi_linux",
   "extra1":  "",
   "extra2":  ""
}
```

Figura 4.2 - Exemplo de JSON para a transmissão de um pedido de uma instrução.

Os campos *category*, *instruction*, *modelId*, *extra1* e *extra2* correspondem aos campos **Categoria da mensagem**, **Instrução**, **Identificador**, **Complemento1** e **Complemento2**, respetivamente, presentes na Tabela 4.1. A adaptação para inglês deve-se a universalidade da linguagem para fins de programação. A Tabela 4.2 faz corresponder cada categoria e instrução em português a *category* e *instruction* aplicada para o desenvolvimento do protótipo.

Tabela 4.2 - Relação entre campos Categoria e Instrução em português com os campos *Category* e *Instruction* usados em inglês.

Categoria da mensagem	**Instrução**	**Category**	**Instruction**
Ação	Iniciar	**action**	start
	Pausar		pause
	Parar		stop
	Reiniciar		restart
Reprogramação	Atualizar	**reprogramming**	update
Implementação	Entrada	**set**	in
Obtenção	Sinal	**get**	signal
	Lista Sinais		allsignal
	Controlador		controller
	Lista Controladores		allcontrolleres

Tal como mencionado, após a receção do pacote JSON proveniente do sistema externo, cabe ao gestor comunicar com o controlador como um cliente e, para tal, foi usada a biblioteca *CURL*. São utilizados os métodos POST e GET para fazer chamadas ao servidor em execução em cada controlador, por forma a realizar os pedidos provenientes de sistemas externos. O gestor procede à invocação de instruções e funções existentes no controlador. Estas instruções destinam-se à modificação e obtenção do estado do modelo e dos valores das portas físicas.

Como mencionado na Secção 3.4 do Capítulo 3, é necessário implementar verificações para legitimar o modelo recebido. Inicialmente é validado o corpo da mensagem verificando se a mensagem se encontra completa e sem falhas de informação.

Em seguida é realizado um levantamento aos modelos em execução na plataforma, utilizando o ficheiro *current_running.xml*. Este ficheiro em XML, possui atributos para o nome do modelo, o número do porto, o número identificador de processo, o estado de execução atual e uma etiqueta para cada sinal usado com os seguintes atributos: identificador das portas de entrada e saída usadas e tipo de sinal, possibilitando uma fácil leitura e extração de informação, tanto para uma máquina como para um humano (Figura 4.3).

```
<?xml version="1.0" encoding="UTF-8"?>
<models>
 <model name="blink_pi_linux" port="8024" pid="1510" state="running">
  <signal gpio_nr="6" type="input"/>
  <signal gpio_nr="7" type="output"/>
 </model>
</models>
```

Figura 4.3 - Exemplo do conteúdo do ficheiro *current_running.xml* quando se encontra um modelo, denominado *blink_pi_linux*.

Usando o nome de cada modelo como a sua identificação, é então efetuada uma comparação com todos os nomes de modelos inseridos no ficheiro e com o nome do modelo recebido para validar a instrução.

Caso não exista nenhum controlador com o mesmo nome em execução na plataforma e a instrução não se destine à execução de uma programação ou reprogramação (por exemplo uma mensagem de controlo), é transmitida uma resposta de falha ao originador da mensagem. Porém na eventualidade da mensagem se destinar a uma reprogramação de um modelo que não conste na plataforma o gestor tratará essa instrução como uma programação de um novo controlador.

Existem ainda mais três verificações necessárias até a implementação do modelo, inerentes à verificação das portas requisitadas pelo controlador. Usando funções para a leitura e interpretação da especificação é possível extrair quais as portas físicas usadas e o respetivo estado. Tendo esse conhecimento em consideração e a pré-existência de um ficheiro informativo, que contém a informação de todas as portas físicas e o estado atual de funcionamento é realizada uma comparação para se obter conclusões sobre a possibilidade de utilização das portas pelo novo controlador.

No caso de o resultado da comparação revelar que o novo controlador tem permissão para o uso de todas as portas este é encaminhado para outra verificação. Porém em caso negativo, o gestor transmite ao sistema externo uma mensagem de erro.

A próxima verificação utiliza o ficheiro *current_running.xml*, para verificar as portas de saída. Estas portas físicas não podem ser utilizadas por dois ou mais controladores em simultâneo. Logo são comparadas as portas físicas requisitadas pelo novo modelo com as portas físicas de saída em utilização, se existir alguma sobreposição é reportado ao sistema externo, se não existir nenhum impedimento passa-se o modelo para a fase de geração e implementação.

Após a validação da instrução da categoria **Ação**, **Implementação** ou **Obtenção** é estabelecida a comunicação com o controlador. Usando o servidor HTTP de cada controlador em execução, o gestor assume o papel de cliente requisitando a execução de funções com a finalidade de manipular o controlador. Os quatro passos de execução entre o gestor e controlador são os seguintes:

1. Inicialização de um canal de comunicação com o controlador;
2. Envio de uma lista de comandos reconhecidos pelo controlador;
3. Resposta do controlador sobre o comando requisitado;
4. Fecho do canal de comunicação.

4.2 Raspberry Pi

Relativamente a plataforma para a validação desta arquitetura foi necessária escolher uma capaz de compilar e executar código “C”, bem como respeitar os requisitos mencionados no capítulo 3. Sendo assim optou-se por escolher sistema-em-um-chip.

Sistema-em-um-chip (SoC – System On Chip) [36] é um circuito integrado capaz de realizar funções analógicas e digitais. É possível encontrar atualmente no mercado pequenos microprocessadores com SoCs para o desenvolvimento de protótipos, nomeadamente o Raspberry Pi.

O Raspberry Pi 3B [37] (Ver Figura 4.4) é um pequeno computador contendo um CPU, com capacidade de executar sistemas operativos baseados em Linux, possui 1GB de RAM e comunicação wireless por WiFi e Bluetooth.

Figura 4.4 - Raspberry Pi modelo 3B.

Este pequeno computador possui portas físicas de entrada e saída (analógica e digital) que permitem a ligação de sensores e atuadores (Ver Figura 4.5). Necessitando apenas da execução um programa que realiza as tarefas de leitura e escrita nas portas físicas.

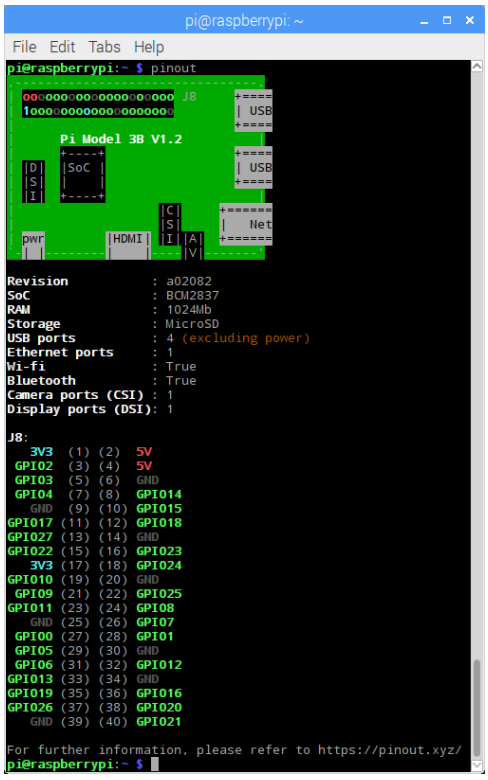

Figura 4.5 - Disposição e identificação das portas físicas presentes num Raspberry Pi 3B.

Tendo em conta que é possível ter um sistema operativo no Raspberry Pi, é possível realizar diversas tarefas em simultâneo, nomeadmanete a execução de diversos controladores e a reprogramação em tempo real de modelos, para tal utilizou-se o sistema operativo Raspian.

4.3 Alterações realizadas nas IOPT-Tools

Para a implementação do sistema/gestor proposto foram utilizadas as IOPT-Tools, devido a possuírem um conjunto de ferramentas com as características necessárias para o tratamento de ficheiros PNML, manipulação de modelos IOPT, comunicação HTTP e geração de código *C*.

Nesta hipótese de implementação, é usado um modelo PNML para especificar o sistema, sendo este o formato de entrada para todas as ferramentas das IOPT-Tools. Porém estas ferramentas necessitaram de modificações para possibilitar a execução automática e o controlo dos respetivos modelos PNML.

Inicialmente foram realizadas modificações aos ficheiros gerados automaticamente para possibilitar a criação de um porto único para o modelo gerado e um nome único para cada controlador em execução, possibilitando assim o controlo individual sem interferência com outros modelos.

Para tal o ficheiro **Makefile**, presente na Tabela 2.1, é modificado (antes de descarregar o código gerado) alterando o porto 8000 (porto predefinido) para o novo porto e o nome do executável para o nome do modelo, como ilustrado na Figura 4.6:

```
$current = file_get_contents("server/http_server.h");
//switch the default PORT to the new PORT
$current = str_replace("8000", $port, $current);
shell_exec('export HTTP_DEBUG=1');
$cMake = 'make net_exec MODEL_NAME='.$out_file;
$check_end = shell_exec($cMake);
```

Figura 4.6 - Código para a alteração do porto e nome do executável do controlador.

Em seguida, foi necessário ceder permissões ao utilizador para guardar e modificar ficheiros diretamente na plataforma. Neste caso é possível realizar estas alterações devido ao servidor se encontrar em execução na mesma plataforma (Raspberry Pi 3B) que irá executar os controladores.

No caso de uma programação ou reprogramação de um modelo IOPT é criada uma pasta com o respetivo nome do modelo e são guardados os ficheiros gerados, de forma a centralizar e simplificar a gestão dos controladores. Porém antes de se proceder à geração de código e implementação do mesmo é necessário aplicar as verificações mencionadas na secção 4.1. Para tal foi desenvolvido uma sequência de verificações usando código PHP, que verifica as diversas hipóteses mencionadas.

No caso de uma instrução para o controlo remoto do controlador, utilizando a interface gráfica das IOPT-Tools, retirou-se partido das funções já desenvolvidas em JavaScript utilizadas pelo *remote debugger*. Porém para a comunicação entre dispositivos externos e o gestor as funções tiveram de ser refeitas para código PHP. Foi utilizanda a biblioteca *CURL* para a comunicação entre gestor e controladores em execução.

Foram ainda desenvolvidas novas funções para comunicação entre sistemas externos e o gestor nomeadamente a obtenção e a implementação de valores de sinais e a obtenção dos controladores do sistema, bem como a sua informação relevante (estado de execução, nome dos sinais de entrada e saída, os respetivos valores e as portas físicas). Estas funcionalidades e a gestão automática de controladores por comunicação por POST encontram-se desenvolvidas no ficheiro *auto_manager.php*.

Por fim, é realizada uma atualização ao um ficheiro *current_running.xml*, sempre que são usadas funções de controlo, programação e reprogramação dos controladores, de forma a manter a informação centralizada para o gestor de execução e reprogramação.

4.4 Interface gráfica de utilizador

As IOPT-Tools apresentam uma interface gráfica web para o desenvolvimento de controladores digitais. Beneficiando desta interface foi implementada uma interface gráfica, similar, para a comunicação com o gestor por utilizadores humanos. Esta área de interação permite supervisionar os controladores de forma remota, permitindo o visionamento em tempo real de todo o sistema, introduzindo uma maior compreensibilidade e simplicidade para o manuseamento dos controladores. As operações mencionadas na secção anterior estão disponíveis através da interface desenvolvida.

A ferramenta *debugger* [38] existente nas IOPT-Tools permite o controlo e monitorização individual das portas físicas e dos estados de execução de um modelo em execução na plataforma pretendida por meio remoto. Servindo-se desta ferramenta, foi implementada uma ação para a implementação de um controlador na plataforma com a possibilidade de redirecionar a página web para a monitorização e teste do modelo correspondente, como demonstrado na Figura 4.7.

Figura 4.7 – Menu para a escolha de implementação com ou sem o *remote debugger*: (1) Implementação do modelo sem abrir a ferramenta *debugger*; (2) Implementação do modelo e redireccionamento para a ferramenta.

Quando esta ferramenta é desativada surge a opção de manter ou remover o controlador em execução, esta característica permite ao utilizador uma maior facilidade para a implementação e remoção de controladores de teste, removendo modelos desnecessários ou mantendo a execução dos mesmos na plataforma.

Para o controlo dos controladores foi desenvolvida uma outra página web que permite o conhecimento de todo o sistema e monitorização individual de cada controlador usufruindo de funções para a remoção do controlador, ferramenta de *debugger* e instruções básicas, tais como, iniciar, pausar, parar e reiniciar. A interface contém o catálogo de todos os controladores e ações possíveis de ativar (Ver Figura 4.8).

Figura 4.8 – Interface gráfica para a visualização e instrução de todos os modelos em execução na plataforma.

5 Validação

Neste capítulo são apresentados testes e resultados efetuados à implementação descrita no capítulo anterior. Esta validação foi realizada em três fases:

1. Programação e Reprogramação de controladores usando a interface humana associada as IOPT-Tools;
2. Validação das instruções básicas implementadas para o controlo do controlador;
3. Uso de um sistema externo para comunicar com o gestor, aplicando as fases anteriores.

5.1 Modelos de controladores para fins de teste

A fim de validar o correto funcionamento do gestor proposto, foi desenvolvido um controlador simplista. Este controlador, denominado *blink_pi_linux*, deverá acender um led quando é acionado um botão de pressão (Ver Figura 5.1).

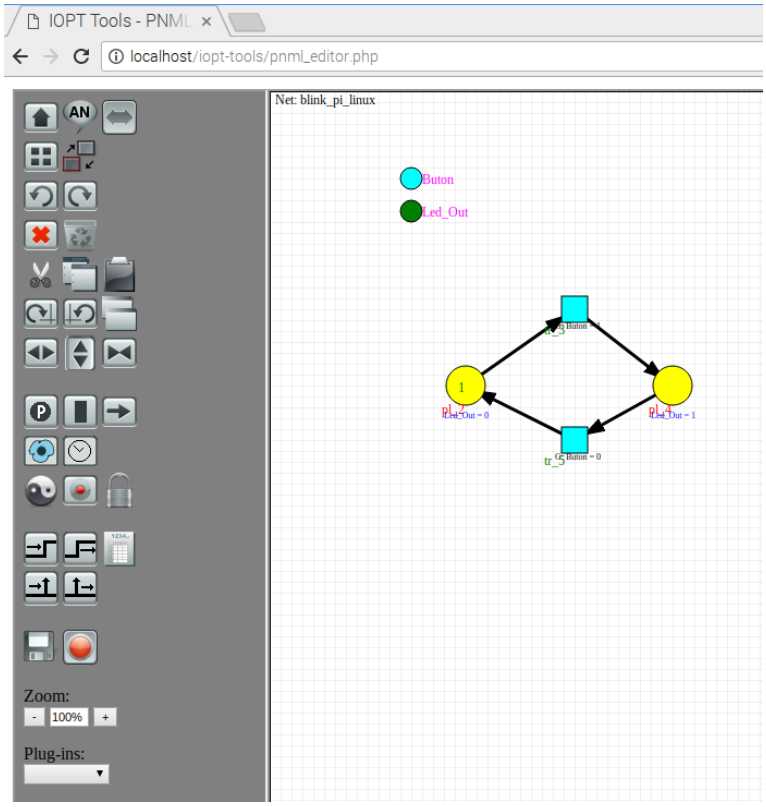

Figura 5.1 - Modelo IOPT desenvolvido do controlador blink_pi_linux.

Definiu-se o sinal de entrada como booleano, para a porta seis do Raspberry Pi, “0” quando o botão de pressão se encontra desligado e “1” quando acionado (Ver Figura 5.2). Para o sinal de saída, também booleano, estabeleceu-se “0” para o estado desligado e “1” para ligado, para a porta sete da plataforma (ver Figura 5.3).

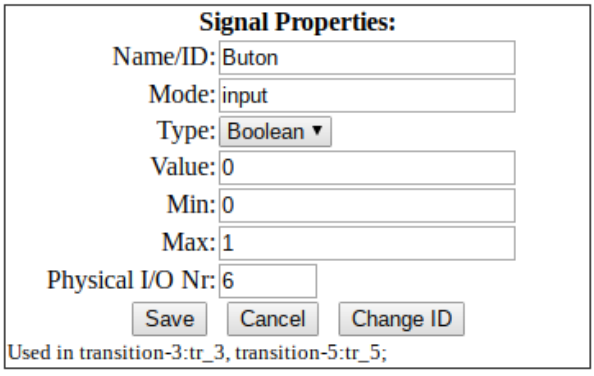

Figura 5.2 - Definição do sinal de entrada para o GPIO 6.

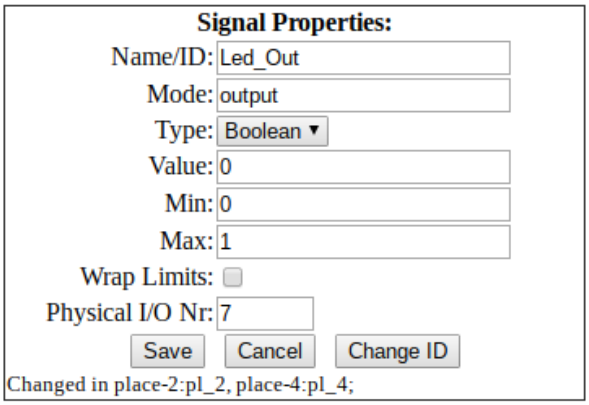

Figura 5.3 - Definição do sinal de saída para o GPIO 7.

Para os testes realizados aqui apresentados, foram selecionados mais dois modelos de controladores, disponíveis no utilizador “models” das IOPT-Tools e descritos em [39] [40].

O primeiro controlador destina-se a monitorizar e controlar o acesso a um parque de estacionamento com uma cancela de entrada e uma de saída. Para a execução deste modelo a plataforma necessita de utilizar seis portas físicas: quatro entradas físicas de sinais para a chegada da viatura, pedido do bilhete, pagamento do estacionamento e saída da viatura e duas saídas de sinal correspondentes a abertura e fecho da cancela, como mostrado na Figura 5.4.

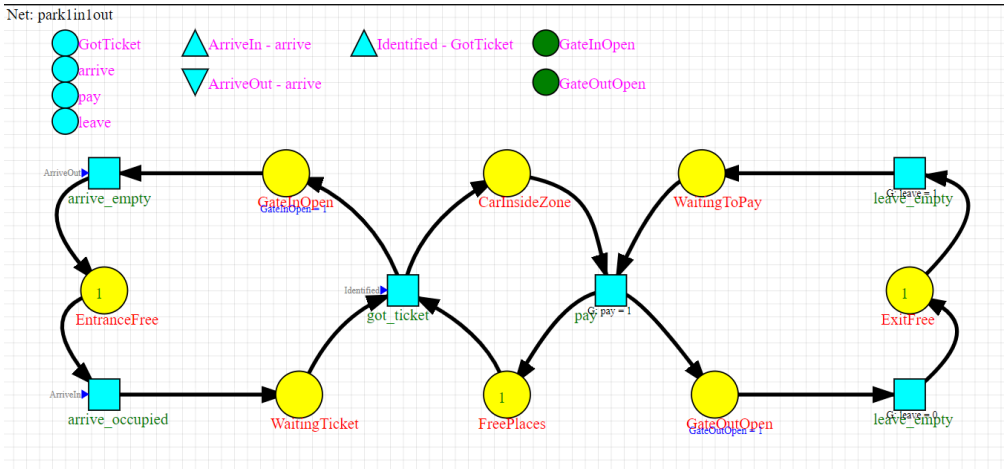

Figura 5.4 – Modelo do controlador de um parque de estacionamento com uma entrada e saída. Adaptado de [39].

O segundo controlador pretende controlar o movimento de um camião-betoneira para o enchimento do deposito com as quantidades ideais de matérias-primas, nomeadamente: água, cimento, areia e gravilha. Por fim é realizado o despejo do betão no local. Neste sentido o modelo IOPT possui dez sinais de entrada: um sinal para o início da execução, um sinal para a chegada em cada estação e um sinal para a confirmação do nível da matéria-prima por estação. Conta ainda

com seis sinais de saída para a início da deslocação e despejo do cimento, da areia, da gravilha, da água e da carga, como observável na Figura 5.5.

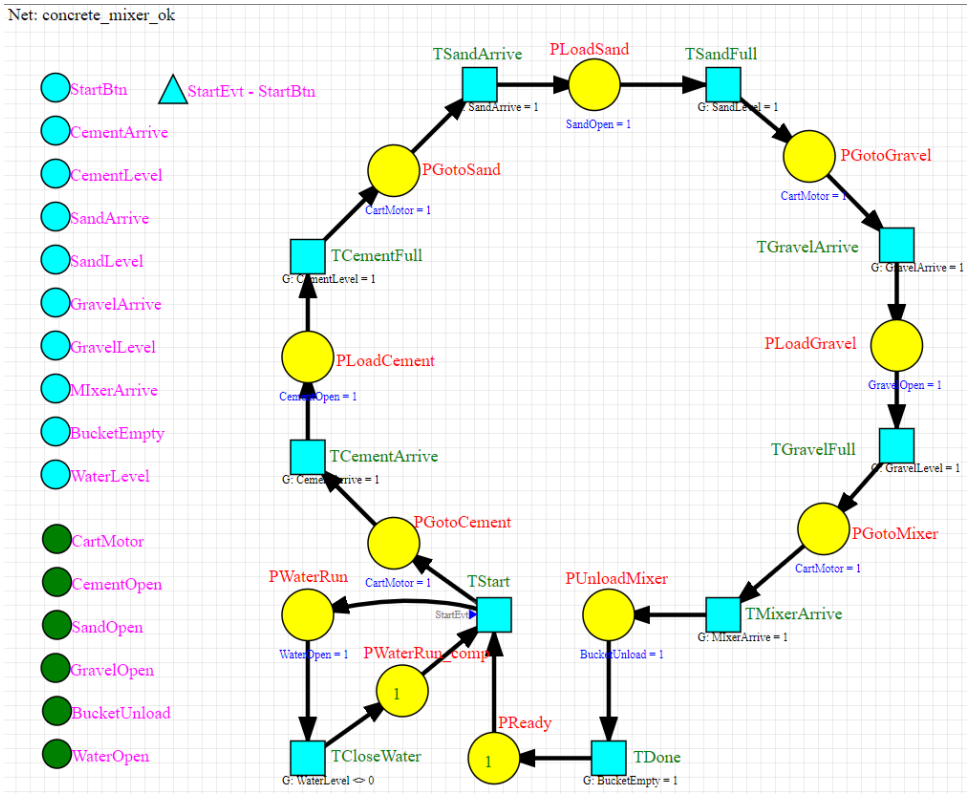

Figura 5.5 – Modelo do controlador de um camião-betoneira para o enchimento de matérias-primas. Adaptado de [40].

5.2 Testes e resultados

Nesta secção são explicados os métodos utilizados de forma a exemplificar a aplicabilidade da solução proposta neste documento, bem como os testes realizados e resultados obtidos. Para a execução dos testes foram utilizados dois métodos: O primeiro método utiliza a interface gráfica web (IOPT-Tools) para o utilizador humano com as novas funcionalidades implementadas; no segundo método pretende-se testar as mesmas funcionalidades e serviços web desenvolvidos para a transmissão de diversas mensagens a partir de um sistema externo para o gestor.

Primeiro método

Com a ajuda de um computador acedeu-se à interface gráfica web (IOPT-Tools), em execução no Raspberry Pi, por via de um internet *browser*. O primeiro teste teve como objetivo a validação da programação automatizada. Ao escolher o modelo *blink_pi_linux* da lista dos modelos disponíveis e acionando o botão *Run in Raspberry* obteve-se a geração de todos os ficheiros

necessários para a fase seguinte. Em seguida, é compilado o código gerado criando um executável do modelo IOPT e por fim é executado o controlador de forma autónoma. Para confirmar que os passos mencionados foram executados com sucesso, o Raspberry Pi foi ligado a uma *breadboard* com um botão de pressão e com um led. Verificou-se que quando o botão gera um sinal de entrada, o controlador em execução interpreta o sinal e gera um sinal de saída, transmitindo tensão aos conectores do led. Como ilustrado na Figura 5.6, o resultado da operação revelou-se positivo, validando a programação do controlador (geração, compilação e execução).

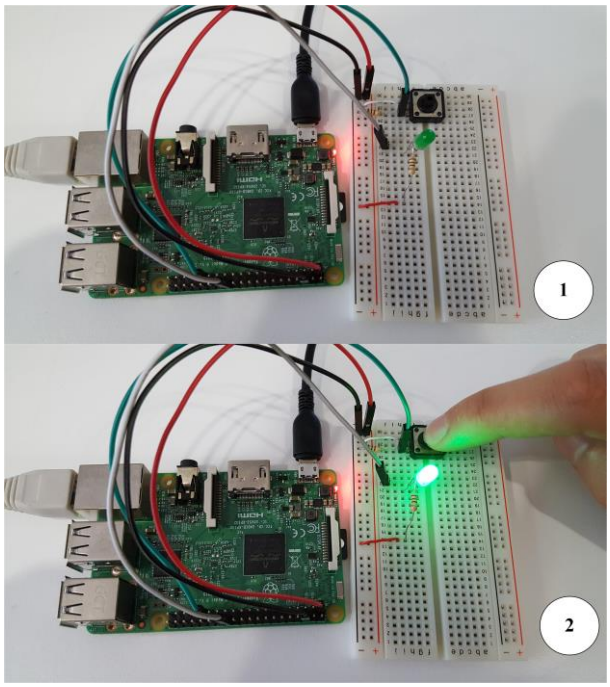

Figura 5.6 - Modelo blink_pi_linux em execução no Raspberry Pi 3; (1) - Botão no estado não acionado com o led em estado desligado; (2) - Botão acionado com o led em estado ligado.

Em seguida, abriu-se a página da interface gráfica correspondente ao catálogo dos controladores em utilização, observando que apenas um controlador se encontrava na plataforma como esperado. O passo seguinte foi acionamento dos botões de Iniciar, Pausar e Parar que constam na interface (ver Figura 4.8), verificando-se que todas as instruções funcionaram como o previsto visto que o estado de execução do modelo foi alterado consoante a instrução envida.

No terceiro teste para o modelo *blink_pi_linux*, foram alteradas as portas de entrada e saída para o botão e led respetivamente. Desta forma foi possível confirmar que no caso de ocorrer uma reprogramação as portas antigas não terão qualquer impacto no funcionamento do controlador e as novas portas irão afetar o controlador. Tal como previsto ao ativar o *botão Run in Raspberry* nenhum erro de implementação ocorreu e as antigas portas deixaram de afetar o controlador.

Após o *debug* com o modelo de teste *blink_pi_linux*, foram colocados os dois modelos mencionados na secção anterior de modo a validar duas características: reinício do um modelo e a gestão/reprogramação de mais do que um modelo em simultâneo.

Sendo estes modelos mais complexos, o reinício de um destes é consideravelmente mais percetível para o ser humano. O modelo do controlador para o camião-betoneira foi colocado em execução, utilizando o mesmo método que anteriormente foi usado para o modelo *blink_pi_linux* e, foi executado até atingir um estado antes de se executar a instrução reiniciar, de modo a verificar o reinício do controlador. Ao verificar que o sinal correspondente à abertura da descarga de areia fora acionado, carregou-se no botão Reiniciar da interface gráfica. Observou-se que o led se apagou e foi necessário o acionamento do botão físico *StartBtn* para o reinício do ciclo de carregamento do camião-betoneira, como esperado.

Por fim foi colocado o modelo para o parque de estacionamento em simultâneo com o modelo já em execução do camião-betoneira. Numa primeira fase, houve a tentativa de implementar um dos modelos com portas físicas de saída iguais a um dos modelos em execução, o que se revelou impossível, tendo sito transmitido um erro para o utilizador, como esperado. Numa segunda fase, colocou-se o modelo com portas físicas de entrada em comum e portas físicas de saída distintas, validando que a implementação se encontrava correta. Em seguida foram executadas as instruções Iniciar, Pausar, Parar, Reiniciar partindo da interface gráfica, estas revelaram o seu correto funcionamento tanto para o novo modelo como para o modelo anteriormente implementado.

Segundo método

Após a validação do correto funcionamento do gestor de execução e reprogramação, utilizando a interface web IOPT-Tools, procedeu-se à aplicação do segundo método de teste, no qual se pertente testar a comunicação entre um sistema externo e o gestor sem a intervenção humana. Para esse fim foi utilizado um programa, num computador Windows, para emular o envio de pacotes de dados com a informação resultante da estrutura no capítulo 4.

Tal como no primeiro método destaca-se o teste do envio de um modelo IOPT para a programação e reprogramação de um controlador e de seguida a transmissão de instruções para monitorização e controlo remoto do controlador.

Do lado do sistema externo (computador Windows), foi formulado a estrutura de um pacote JSON, sendo os campos *sender* e *authentication* obrigatórios e com a informação correspondente à ilustrada no exemplo seguinte:

```
{
    "sender": {"address":"127.0.0.1","port":"8454"},
   "authentication": {"user":"pmpr","pass":"pmpr2018"},
   "category":"action",
   "instruction": "start",
   "modelId": "blink_pi_linux",
   "extra1":  "",
   "extra2":  ""
}
```


Figura 5.7 - Estrutura do pacote JSON enviado do sistema externo para o gestor.

Para o controlo da execução controlador a categoria escolhida é *action*, variando o campo *instruction* com as expressões: *start*, *pause*, *stop* e *restart*. Por fim é indicando o identificador do modelo, que neste protótipo corresponde ao nome do modelo IOPT, deixando os campos *extra1* e *extra2* vazios.

Para a programação e reprogramação de um controlador a categoria escolhida é *reprogramming*, deixando o campo *instruction* vazio e indicando o identificador do modelo (nome do modelo IOPT). O campo *extra1* corresponde ao modo de prioridade da instrução, que nesta implementação não foi considerado e *extra2* corresponde ao modelo IOPT. O modelo IOPT enviado necessita de ser alterado antes de ser envido, substituindo o caracter aspa pelo caracter pelica e o \n pelo código &#10. Na receção o gestor irá interpretar esta informação e descodificar o modelo IOPT.

Por fim, para a monitorização do controlador, o corpo da mensagem enviado necessita de possuir a *category get* e a uma das opções da lista de *instruction*: *signal*, *allsignals*, *controller*, *allcontrollers*. O *signal* permite a obtenção do valor de um determinado sinal que esteja associado ao controlador. O *allsignals* permite a obtenção de todos os valores dos sinais que são utilizados pelo controlador. O *controller* devolve a informação detalhada sobre o modelo (estado de execução, nome dos sinais, a porta física dos sinais, tipo de sinal e valor de cada sinal). O *allcontrolleres* devolve uma lista de todos os controladores e respetivos estados de execução.

Todos os testes realizados obtiveram o mesmo resultado que no primeiro método

Comparação dos métodos

Ambas as implementações, interface gráfica IOPT-Tools e servidor web por POST, reduzem o tempo gasto para a implementação de controladores e facilitam o controlo remoto de diversos controladores em simultâneo, porém é relevante verificar o tempo que as mesmas instruções demoram para os dois métodos implementados. É de salientar que os testes realizados admitem um ambiente onde apenas é abrangido um controlador, porém não é possível garantir que nenhum processo externo se encontra em execução no instante da requisição da instrução.

Em seguida, é apresentada na Tabela 5.1 o tempo que demora a execução da programação e reprogramação de um controlador para os dois métodos implementados. Foram realizados 20 testes para cada um dos métodos para cada um dos métodos.

Tabela 5.1 - Comparação de tempos de programação e reprogramação entre a utilização da interface IOPT-Tools e o método POST enviado por um sistema externo.

Tentativa	**Interface IOPT-Tools tempo de duração (s):**		**Comunicação por POST tempo de duração (s):**	
	Programação	Reprogramação	Programação	Reprogramação
1	3.3450	3.4924	3.4493	3.4217
2	3.4012	3.8022	3.3997	3.4527
3	3.3876	3.6522	3.3787	3.5462
4	4.0108	3.5964	3.3554	3.5324
5	3.4113	3.6325	3.4098	3.5414
6	3.3717	3.6657	3.4024	3.4247
7	3.3915	3.8062	3.3999	3.5695
8	3.4167	3.6921	3.4881	3.6240
9	3.4300	3.7280	3.4074	3.4388
10	3.5070	3.7571	3.4077	3.5900
11	3.4556	3.4868	3.3697	3.4886
12	3.3819	3.5378	3.3683	3.4784
13	3.3207	3.5421	3.4162	3.5262
14	3.5211	3.8812	3.4912	3.4912
15	3.4338	3.6159	3.3862	3.4476
16	3.4395	3.5282	3.3075	3.4881
17	3.5074	3.5549	3.4141	3.5516
18	3.4523	3.5884	3.3082	3.4767
19	3.4279	3.6300	3.3740	3.5103
20	3.4434	3.7487	3.3001	3.5167
Média	3.4528	3.6469	3.3917	3.5058
Mediana	3.4289	3.6313	3.3998	3.5007
Máximo	4.0108	3.8812	3.4912	3.6240
Mínimo	3.3207	3.4868	3.3001	3.4217

A partir da tabela anterior, da Figura 5.8 e da Figura 5.9 é possível validar que ambas as implementações demoram tempos semelhantes tanto para a programação como para a reprogramação do controlador, porém é de salientar que o método de comunicação por POST é mais rápido e obtiveram-se tempos mais semelhantes que o método da interface IOPT-Tools.

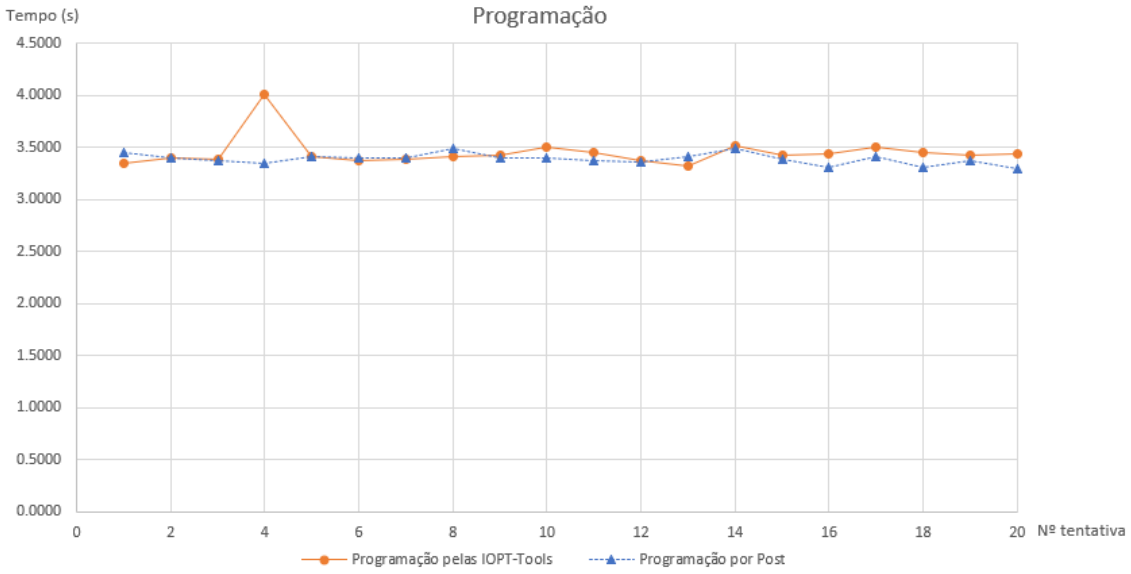

Figura 5.8 - Gráfico relativo ao tempo necessário para a realização da programação utilizando os dois métodos.

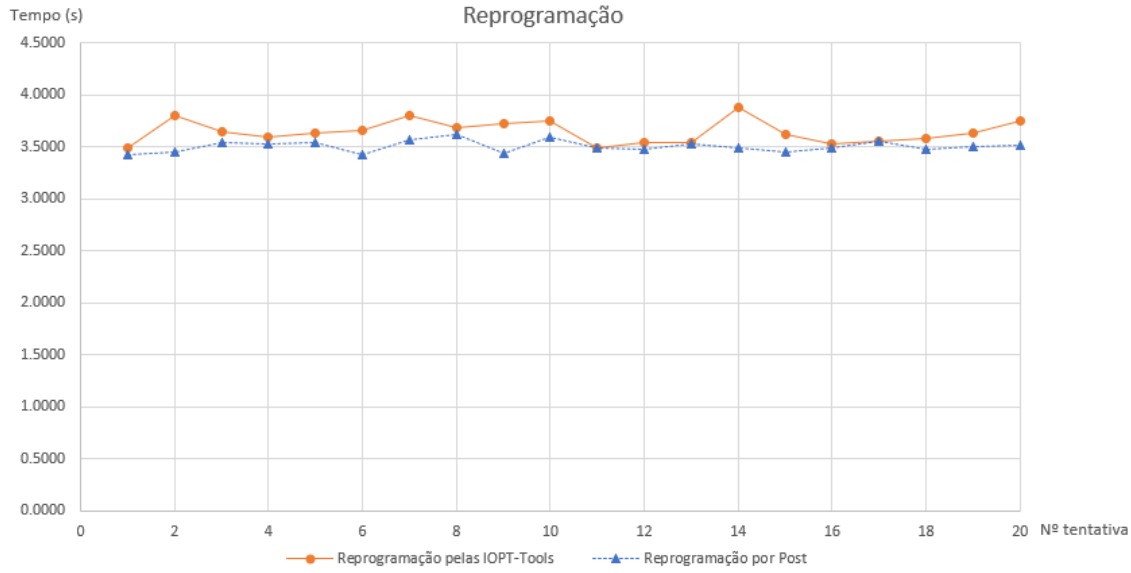

Figura 5.9 - Gráfico relativo ao tempo necessário para a realização da reprogramação utilizando os dois métodos.

Foram também retirados os tempos de execução das instruções para o controlo remoto de controladores em execução. A Tabela 5.2 apresenta os diversos tempos obtidos desde do pedido até à resposta para cada instrução de controlo.

Tabela 5.2 - Comparação de tempos para as diversas instruções de controlo remoto dos controladores em execução entre a utilização da interface IOPT-Tools e o método POST enviado por um sistema externo.

Tentativa	Interface IOPT-Tools, tempo de duração (s):					Comunicação por POST, tempo de duração (s):				
	Iniciar para Pausa	Pausa para Iniciar	Iniciar para Parar	Parar para Iniciar	Reiniciar	Iniciar para Pausa	Pausa para Iniciar	Iniciar para Parar	Parar para Iniciar	Reiniciar
1	0.0840	0.0560	0.1230	0.1730	0.2290	0.0105	0.0042	0.0649	0.2639	0.2865
2	0.0520	0.0800	0.1230	0.2150	0.1970	0.0088	0.0109	0.0589	0.2550	0.2650
3	0.0800	0.0790	0.1160	0.2250	0.2240	0.0042	0.0112	0.0573	0.2637	0.2682
4	0.0530	0.0850	0.1640	0.2030	0.1930	0.0058	0.0029	0.0527	0.3075	0.2650
5	0.0800	0.0760	0.1190	0.1920	0.1920	0.0025	0.0088	0.0531	0.2638	0.2707
6	0.0780	0.0590	0.1220	0.1870	0.2060	0.0069	0.0203	0.0520	0.2597	0.2761
7	0.0520	0.0810	0.1210	0.2100	0.2040	0.0121	0.0058	0.0520	0.2953	0.2708
8	0.0740	0.0730	0.1690	0.2350	0.2280	0.0021	0.0021	0.0579	0.2919	0.2883
9	0.0470	0.0820	0.1130	0.2230	0.2070	0.0020	0.0020	0.0510	0.2981	0.2962
10	0.0830	0.0670	0.1120	0.2220	0.2020	0.0071	0.0022	0.0539	0.2497	0.2726
11	0.0840	0.0740	0.1210	0.2260	0.2050	0.0138	0.0171	0.0540	0.2924	0.2659
12	0.0520	0.0810	0.1140	0.2360	0.1830	0.0085	0.0083	0.0536	0.2960	0.2726
13	0.0550	0.0790	0.1200	0.2060	0.2230	0.0084	0.0100	0.0561	0.2729	0.2862
14	0.0570	0.0770	0.1130	0.2130	0.2040	0.0259	0.0166	0.0500	0.2560	0.2704
15	0.0750	0.0790	0.1670	0.2070	0.1970	0.0078	0.0040	0.0581	0.2454	0.2700
16	0.0570	0.0750	0.1440	0.1740	0.2150	0.0020	0.0069	0.0575	0.2692	0.2752
17	0.0530	0.0780	0.1320	0.1730	0.1970	0.0020	0.0020	0.0606	0.2703	0.2957
18	0.0570	0.0770	0.1140	0.1960	0.2100	0.0020	0.0070	0.0541	0.2616	0.2752
19	0.0560	0.0770	0.1270	0.2000	0.2160	0.0020	0.0021	0.0563	0.2960	0.2683
20	0.0820	0.0640	0.1160	0.1830	0.1960	0.0033	0.0022	0.0541	0.2738	0.2745
Média	0.0656	0.0750	0.1275	0.2050	0.2064	0.0069	0.0073	0.0554	0.2741	0.2757
Mediana	0.0570	0.0770	0.1210	0.2065	0.2045	0.0064	0.0064	0.0541	0.2697	0.2726
Máximo	0.0840	0.0850	0.1690	0.2360	0.2290	0.0259	0.0203	0.0649	0.3075	0.2962
Mínimo	0.0470	0.0560	0.1120	0.1730	0.1830	0.0020	0.0020	0.0500	0.2454	0.2650

Figura 5.10 - Gráfico relativo ao tempo necessário para a realização da instrução “Pausar” quando o controlador se encontra em execução, utilizando os dois métodos.

Figura 5.11 - Gráfico relativo ao tempo necessário para a realização da instrução “Iniciar” quando o controlador se encontra em estado de pausa, utilizando os dois métodos.

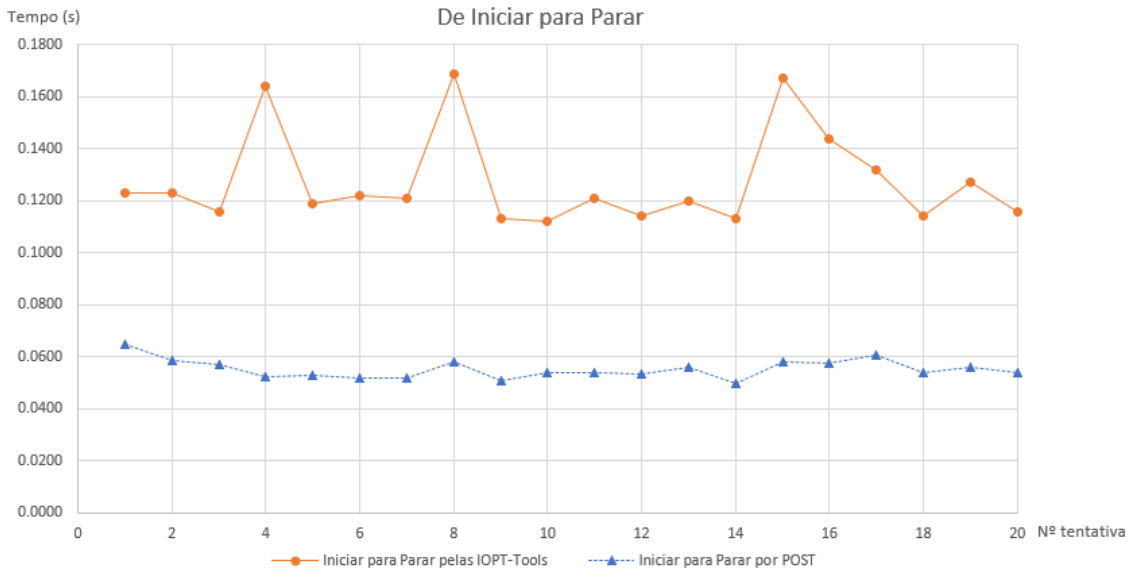

Figura 5.12 - Gráfico relativo ao tempo necessário para a realização da instrução "Parar" quando o controlador se encontra em execução, utilizando os dois métodos.

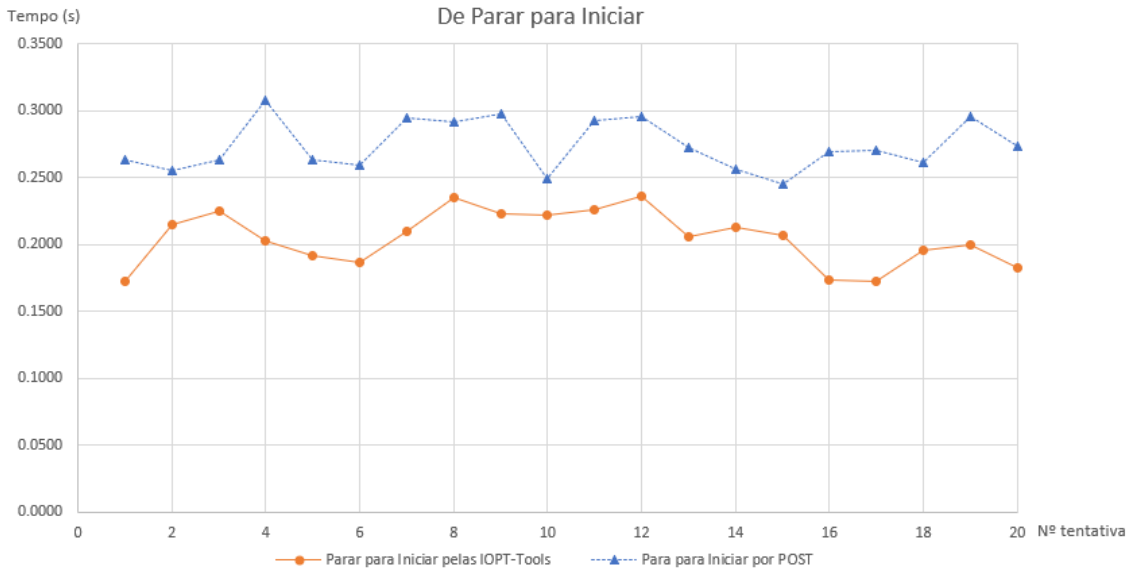

Figura 5.13 - Gráfico relativo ao tempo necessário para a realização da instrução "Iniciar" quando o controlador não se encontra execução, utilizando os dois métodos.

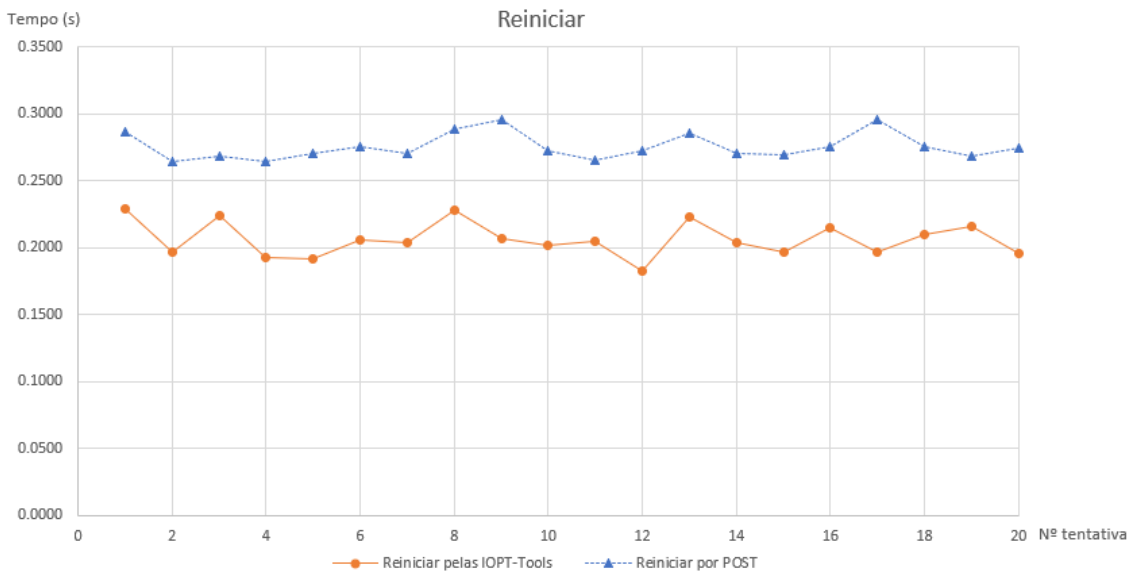

Figura 5.14 - Gráfico relativo ao tempo necessário para a realização da instrução "Reiniciar" de um controlador, utilizando os dois métodos.

Ao traçar o gráfico de cada instrução (Figura 5.11, Figura 5.12, Figura 5.13 e Figura 5.14) é possível observar que o uso do método POST é relativamente mais rápido que o uso da interface gráfica das IOPT-Tools, isto deve-se ao facto do uso da interface estar associada a um servidor que necessita de executar funções para a atualização da página e executar diversos métodos implementados em linguagens como JavaScript, HTML, CSS e PHP. Porém, tanto iniciar um controlador quando este se encontra parado como reiniciar um controlador, demoram mais tempo usando o método POST. Isto acontece porque a implementação deste método realiza mais procedimentos do que realiza o método que é executado quando é usada a interface IOPT-Tools.

No decorrer dos testes para a validação do sistema observou-se que as instruções mais demoradas são a programação e reprogramação, obtendo um tempo médio aproximado de 3.5 segundos desde o pedido até ao controlador se encontrar no estado de execução para ambos os métodos. Este acontecimento deve-se maioritariamente à compilação do código gerado na plataforma de desenvolvimento. Durante este período o servidor web fica num estado de bloqueio, impedindo qualquer interação do utilizador com a mesma. Em relação a todas as outras instruções o tempo de espera é quase inexistente para o ser humano por que se encontram muito abaixo de 0.5 segundos.

6 Conclusões e Trabalhos futuros

Neste capítulo são apresentadas as conclusões alcançadas ao longo do desenvolvimento do trabalho apresentado neste documento, bem como algumas hipóteses de trabalhos a desenvolver no futuro.

De acordo com os objetivos inicialmente traçados, este trabalho propõe um sistema para a gestão e reprogramação de controladores, em tempo de execução. Este gestor permite que sistemas externos possam comunicar com os controladores, sendo disponibilizados métodos para a automatização de processos de controlo e monitorização, sem a necessidade de intervenção humana.

Este sistema possibilita a gestão da execução de diversos controladores em simultâneo e a sua reprogramação. Disponibiliza serviços web para permitir que sistemas externos possam reprogramar e manipular os controladores. Este gestor foi testado em conjunto com as ferramentas IOPT que lhe permitem usufruir de geradores automáticos de código e uma interface para implementar automaticamente o modelo em diferentes plataformas. Antes de gerar e executar o código de implementação dos controladores, o gestor valida os modelos, de maneira a precaver erros ou falhas, possibilitando uma reprogramação em tempo de execução sem afetar o funcionamento dos restantes controladores.

Comparando este sistema com as WSN apresentadas no capítulo 2, este sistema permite a reprogramação e monitorização do ambiente envolvente bem como a comunicação entre diversos dispositivos. Adicionalmente, este sistema possibilita o controlo remoto de atuadores, afetando outros sistemas.

Para a gestão de diversos controladores em simultâneo, o sistema dispõe de um canal de comunicação privado para cada controlador em funcionamento e o conhecimento das portas utilizadas por cada um.

A solução proposta no capítulo 3 deste documento define o comportamento, as entradas e saídas do gestor, deixando a responsabilidade a quem implementa o gestor sobre qual o protocolo de comunicação a aplicar entre o sistema externo e os controladores.

No capítulo 4 é apresentada uma implementação do gestor com recurso às ferramentas existentes nas IOPT-Tools, validando as mesmas para a implementação desejada. Foram realizadas diversas alterações às ferramentas para correcta implementação do gestor. Este protótipo foi implementado e usado para validar o gestor proposto, estando disponível online em *http://gres.uninova.pt/~pmpr/iopt-tools*.

O sistema desenvolvido é capaz de efetuar a reprogramação e manipulação de modelos, tanto quando são transmitidos pela interface gráfica das IOPT-Tools como quando são enviados por um outro dispositivo. A invocação e adaptação de algumas ferramentas das IOPT-Tools, nomeadamente, o gerador de código, o *remote debugger* e a interface gráfica, revelaram que o uso das ferramentas para os fins requeridos é essencial na implementação e na gestão dos modelos.

Assim sendo, as IOPT-Tools mostraram-se como uma boa escolha para o desenvolvimento deste sistema, por possuírem características que permitem a geração de código automático contendo a definição das portas de entrada e saída, bem como uma biblioteca de funções integrada para a manipulação/monitorização de cada controlador.

Desta forma pode concluir-se que o sistema proposto é uma solução válida para a gestão de execução e reprogramação de controladores sem a intervenção humana. O facto deste gestor receber modelos dos controladores em vez de código de implementação dos mesmos, mostrou-se uma vantagem, permitindo aos sistemas externos não necessitarem de conhecer a plataforma em que será colocado o modelo em funcionamento. Cabe ao gestor a tradução do formato do modelo numa linguagem compreendida pela plataforma. Esta abordagem permite a substituição da plataforma sem necessidade de afectar ou informar os sistemas que comunicam com o gestor.

Comparando a implementação deste sistema com os recursos humanos e ferramentas necessárias para o mesmo efeito, considera-se que o tempo de reprogramação e modificação da execução do modelo, bem como a logística e acesso ao equipamento desejado compensam a introdução deste sistema na rede. Os reduzidos tempos observados aquando da implementação de controladores, a facilidade e comodidade tornam este sistema desejável, além que a integração do mesmo torna possível a comunicação entre diferentes sistemas.

Todavia ficou por desenvolver a utilização do "modo de prioridade" mencionado no capítulo 3. Este modo destina-se a analizar a prioridade que uma instrução/reprogramação tem perante outra proveniente do mesmo ou de outro sistema externo, estando prevista a sua implementação num futuro próximo.

Com trabalho futuro pretende-se adicionar diversos geradores de código, para suportar a implementação dos controladores em plataformas heterógeneas, bem como um algoritmo de decisão para a escolha autónoma por parte do gestor sobre quais destes geradores deve utilizar.

Seria também de interesse realizar a mesma implementação usada para as IOPT-Tools com outros ambientes que suportem geração de código a partir de modelos como as IOPT-Flow, o que permitirá ter sistemas que recebam, além de modelos IOPT, outros modelos como DS-Pnets e/ou modelos MIS, expandido a aplicabilidade do gestor e contribuindo para o melhoramento destas ferramentas.

Importa referir que durante a realização da presente dissertação foi publicado um artigo na conferência, REC 2018 - XIV Jornadas sobre Sistemas Reconfiguráveis, intitulado "Componente modular reconfigurável para a indústria 4.0" [41] da autoria de Thomas Pedro Noronha, Pedro Rodrigues, Filipe Moutinho, Rogério Campos-Rebelo, Pedro Maló e Luís Gomes, onde foram propostas, ainda que de forma preliminar, as funcionalidades do sistema/gestor aqui descrito.

Por fim, o protótipo do gestor que se encontra na área *http://gres.uninova.pt/~pmpr/iopt-tools* será colocado na versão atual das IOPT-Tools, em *http://gres.uninova.pt/IOPT-Tools/*, para o usufruto de qualquer utilizador da ferramenta.

Referências

[1] Deloitte, “Industry 4.0. Challenges and solutions for the digital transformation and use of exponential technologies,” *Deloitte*, pp. 1–30, 2015.

[2] R. Petrasch and R. Hentschke, “Cloud storage hub: Data management for IoT and industry 4.0 applications: Towards a consistent enterprise information management system,” in *2016 Management and Innovation Technology International Conference, MITiCON 2016*, 2017.

[3] “What is Big Data? - Definition from Techopedia.” [Online]. Available: https://www.techopedia.com/definition/27745/big-data. [Accessed: 24-Jun-2018].

[4] A. Bröring *et al.*, “New generation Sensor Web Enablement,” *Sensors*. 2011.

[5] D. Ahuja and N. Chaudhary, “Programmable Logic Controller,” *Int. J. Inf. Comput. Sci.*, no. August, pp. 115–120, 2012.

[6] “OTA (Over-The-Air) Definition.” [Online]. Available: https://techterms.com/definition/ota. [Accessed: 03-Mar-2018].

[7] H. Chandra, E. Anggadjaja, P. S. Wijaya, and E. Gunawan, “Internet of Things: Over-the-Air (OTA) firmware update in Lightweight mesh network protocol for smart urban development,” in *Proceedings - Asia-Pacific Conference on Communications, APCC 2016*, 2016.

[8] V. S. Varadharajan, D. S. Onge, C. Gub, and G. Beltrame, “Over-the-air updates for robotic swarms,” *IEEE Softw.*, 2018.

[9] IEC, “Internet of Things : Wireless Sensor Networks Executive summary,” p. 78, 2014.

[10] E. Eronu, S. Misra, and M. Aibinu, “Reconfiguration approaches in Wireless Sensor Network: Issues and challenges,” in *2nd International Conference on Emerging and Sustainable Technologies for Power and ICT in a Developing Society, IEEE NIGERCON 2013 - Proceedings*, 2013, pp. 143–152.

[11] P. Levis *et al.*, “TinyOS: An operating system for sensor networks,” in *Ambient Intelligence*, 2005.

[12] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, “A dynamic operating system for sensor nodes,” in *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys ’05*, 2005.

[13] A. Dunkels, B. Grönvall, and T. Voigt, “Contiki - A lightweight and flexible operating system for tiny networked sensors,” in *Proceedings - Conference on Local Computer Networks, LCN*, 2004.

[14] S. Bhatti *et al.*, “MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms,” in *Mobile Networks and Applications*, 2005.

[15] A. S. A. Quadri and B. O. Sidek, “An Introduction to Over-the-Air Programming in Wireless Sensor Networks,” *Int. J. Comput. Sci. Netw. Solut.*, vol. 2, pp. 33–49, 2014.

[16] “Libelium - Connecting Sensors to the Cloud.” [Online]. Available: http://www.libelium.com/. [Accessed: 25-Feb-2018].

[17] S. Bougouffa, K. Meszmer, S. Cha, E. Trunzer, and B. Vogel-Heuser, “Industry 4.0 interface for dynamic reconfiguration of an open lab size automated production system to allow remote community experiments,” in *IEEE International Conference on Industrial Engineering and Engineering Management*, 2018, vol. 2017–Decem, pp. 2058–2062.

[18] Omg, “Meta Object Facility (MOF) Core Specification,” *Management*, vol. 080907, no. January, pp. 1–76, 2006.

[19] DIN-EN-61512-1, “Batch control - Part 1: Models and terminology,” in *Batch control - Part 1: Models and terminology*, no. 1, 2000, p. 52.

[20] S. Berlik, “Eclipse Modeling Framework,” *Framework*, pp. 1–14, 2007.

[21] M. Marcos, E. Estevez, F. Perez, and E. Van Der Wal, “XML exchange of control programs,” *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 32–35, 2009.

[22] OPC Foundation, “OPC Unified Architecture Part 1: Overview and Concepts Version 1.02,” *OPC Foundation*. pp. 1–30, 2012.

[23] Z. Nedic, “Demonstration of collaborative features of remote laboratory NetLab,” *Int. J. Online Eng.*, 2012.

[24] R. Bitter, T. Mohiuddin, and M. Nawrocki, “Introduction to LabVIEW,” *Fac. Technol.*, 2014.

[25] M. H. Siddiqui, V. Purohit, and S. Mane, “Embedded web server based NetLab for remote access,” in *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, vol. 3, pp. 1–5.

[26] F. Pereira, F. Moutinho, L. Gomes, J. Ribeiro, and R. Campos-Rebelo, “An IOPT-net state-space generator tool,” in *IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 383–389.

[27] L. Gomes, F. Moutinho, F. Pereira, J. Ribeiro, A. Costa, and J. P. Barros, “Extending input-output place-transition Petri nets for distributed controller systems development,” in *Proceedings - 2014 International Conference on Mechatronics and Control, ICMC 2014*, 2015, pp. 1099–1104.

[28] M. Weber and E. Kindler, “The Petri Net Markup Language,” *Petri Net Technol. Commun. Syst.*, pp. 124–144, 2003.

[29] F. Pereira, F. Moutinho, and L. Gomes, “IOPT-tools-Towards cloud design automation of digital controllers with Petri nets,” in *Proceedings - 2014 International Conference on Mechatronics and Control, ICMC 2014*, 2015, pp. 2414–2419.

[30] R. Campos-Rebelo, F. Pereira, F. Moutinho, and L. Gomes, “From IOPT Petri nets to C: An automatic code generator tool,” in *IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 390–395.

[31] F. Pereira and L. Gomes, “Automatic synthesis of VHDL hardware components from IOPT Petri net models,” in *IECON Proceedings (Industrial Electronics Conference)*, 2013, pp. 2214–2219.

[32] R. Jorge, P. Moreira, and H. Académicas, “IOPT2IEC61131 – Execução de redes de Petri em Autómatos industriais (PLCs),” 2016.

[33] F. Moutinho, F. Pereira, and L. Gomes, “IOPT Tools User Manual,” vol. 2014, no. C, pp. 1–50, 2014.

[34] F. Pereira *et al.*, “The IOPT-Flow Modeling Framework Applied to Power Electronics Controllers,” *IEEE Trans. Ind. Informatics*, vol. 3203, no. 3, pp. 1–1, 2017.

[35] R. A. B. C. Rebelo, “Modelização de Eventos : de interação do sistema com o ambiente a modelos,” pp. 1–148, 2016.

[36] “System-on-chip | Article about System-on-chip by The Free Dictionary.” [Online]. Available: http://encyclopedia2.thefreedictionary.com/System-On-Chip. [Accessed: 02-Mar-2018].

[37] “Raspberry Pi 3 Model B - Raspberry Pi.” [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 02-Mar-2018].

[38] F. Pereira, A. Melo, and L. Gomes, “Remote operation of embedded controllers designed using IOPT Petri-nets,” in *Proceeding - 2015 IEEE International Conference on Industrial Informatics, INDIN 2015*, 2015.

[39] L. Gomes, “Model-based development of distributed embedded controllers - Rapid prototyping using IOPT-tools and FPGAs - Rapid p,” in *2016 International Conference on FPGA Reconfiguration for General-Purpose Computing, FPGA4GPC 2016*, 2016.

[40] F. Pereira and L. Gomes, “The IOPT-Flow framework pairing Petri nets and data-flows for embedded controller development,” in *IECON Proceedings (Industrial Electronics Conference)*, 2016, pp. 4832–4837.

[41] T. P. Noronha, P. Rodrigues, R. Campos-Rebelo, F. Moutinho, P. Maló, and L. Gomes, “Componente modular reconfigurável para a indústria 4.0,” *REC*, p. 4, 2018.