Duarte Maria Macedo Saraiva

Licenciado em Engenharia Eletrotécnica e de Computadores

Simulador Híbrido de Processo Industrial

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores

Orientador: Doutor Fernando José Vieira do Coito, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente: Professor Doutor Rui Miguel Henriques Dias Morgado Dinis

Arguente: Professor Doutor Luís Filipe Figueira Brito Palma

Vogais: Professor Doutor Fernando José Almeida Vieira do

Setembro, 2021

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Simulador Híbrido de Processo Industrial

Dedicatória aos meus avós Alice e António Aparício

Agradecimentos

Ao meu orientador Professor Fernando Coito pela ajuda e orientação prestada. À minha família, Mãe e Pai pela paciência. A todos aqueles que incansavelmente me motivaram e raciocinaram comigo.

Resumo

Hoje em dia, existe falta de equipamentos de baixo custo que permitam tanto na indústria como na área educacional (conceito aplicando à pequena escala) conceder acesso a simuladores e controladores que possibilitem estudos de comportamento de sistemas de controlos remotamente. Estes dispositivos são essenciais no ensino pois possibilitam a execução de experiências aplicada à área de controlo. Esta experiência é importante porque ao ser bem-sucedida admite dotar a universidade ou uma indústria da capacidade de replicação de módulos de controlo de baixo custo, resolvendo uma necessidade de simular remotamente processos que estão a correr, consentido se assim for desejado pôr a correr no processo real novos controladores. A solução que se propõe neste projecto é criar um sistema modular, com *hardware* de baixo custo. Pretende-se que este projecto consiga dar aos estudantes e professores, no âmbito educativo, uma capacidade de simulação, ensino e avaliação através dos vários perfis de utilizador, fazer a configuração de uma experiência remotamente, facilitando o acesso aos alunos nos seus projectos de controlo e quando estes correctos possibilita a sua testagem. Esta nova realidade permitirá aos alunos testar sistemas remotos de controlo. Deverá permitir ser actualizável, conter um módulo base de comunicação remota, conjunto entradas/saídas digitais e analógicas, e um sistema de gestão, simulação e actuação de um processo industrial de controlo.

Palavras-chave: Simulador, Controlo, Acesso Remoto, Processo Real, *Python*, *Raspberry Pi*, *SQL*

Abstract

Nowadays, there is a lack of low-cost equipment that allows both industry and education (concept applied to the small scale) to provide access to simulators and controllers that allow remote studies of control system behaviour. These devices are essential in education because they allow the execution of experiments applied to the control area. This experiment is important because if successful it will allow the university or an industry to provide the capability to replicate low-cost control modules, solving a need to remotely simulate processes that are running, allowing new controllers to be run in the real process if so desired. The solution proposed in this project is to create a modular system, with low cost hardware. It is intended that this project will be able to give students and teachers in the educational case a capacity for simulation, teaching and evaluation through the various user profiles, make the configuration of an experiment remotely, facilitating access to students in their control projects and when these are correct allow them to test them. This new reality will allow students to test remote systems. It should be upgradeable, contain a basic remote communication module, digital and analog input/output set, and a system for managing, simulating and acting on an industrial process control.

Keywords: Simulator, Control Systems, Remote Access, Real-time Systems, *Python, Raspberry Pi, SQL*

Conteúdo

Índice de Figuras

Índice de Tabelas

Índice de Acrónimos

AD: Analógico-Digital

BD: Base de Dados

DA: Digital-Analógico

GPIO: *General-Purpose Input/Output*

HTML: Hypertext Markup Language

I2C: *Inter-Integrated Circuit*

IEEE: *Institute of Electrical and Electronics Engineers*

IPV4: *Internet Protocol version 4*

JDBC: *Java Data Base Connectivity*

JSON: Java Script Object Notation

MISO: *Master In Slave Out*

MOSI: *Master Out Slave In*

NCS: *Network Control System*

NI: *National Instruments*

ODBC: *Open Database Connectivity*

PHP: Hypertext Preprocessor

PWM: Pulse Width Modulation

SCL: Serial Clock

SDA:Serial Data

SPI: *Serial Peripheral Interface*

TCP/IP: Transmission Control Protocol/Internet Protocol

UART: Universal Asynchronous Receiver-transmitter

USB: *Universal Serial Bus*

Wi-Fi: Wireless Fidelity

1

Introdução

No mundo actual os seres humanos encontram-se ligados à rede de *Internet*, através de vários dispositivos, e de certo modo participam no funcionamento da mesma [1].

A massificação da utilização da rede de internet impulsionou o desenvolvimento de novos dispositivos como microcomputadores e microcontroladores de baixo custo. Estes viabilizam a interacção do utilizador comum com o mundo digital, e para utilizadores mais avançados traz a possibilidade de desenvolver projectos de electrónica e computação, entre outros.

Os microcomputadores, e os microcontroladores de baixo custo, num contexto industrial permitem:

- Desenvolver formas de controlar processos de fabrico;
- Simulação de processos físicos de diferente natureza, como térmicos e mecânicos;
- Ainda podem ser utilizados, por exemplo, para realizar simulações sem interferir no processo real a decorrer, como por exemplo uma linha de montagem [2];
- Permitem também o armazenamento dos dados, gerando assim um histórico que permite analisar o comportamento dos processos e dos controladores;
- Após simulações e testes efectuados com sucesso é possível a implementação imediata em linhas de produção.

Os microcomputadores de baixo custo, num contexto educacional, também apresentam enumeras vantagens como:

- Implementar laboratórios;
- Existe a possibilidade de implementar acessos distintos para professor e alunos. Os professores por acesso remoto podem verificar resultados de simulações de alunos e configurar o dispositivo para o respectivo processo a simular e testar. Os alunos poderão fazer as simulações e se a permissão for concedida podem experimentar no processo real;
- Este simulador pode ser utilizado no geral para qualquer processo experimental laboratorial que exista no departamento. Sendo para isso necessário interligar a plataforma de simulação processo laboratorial;
- Este simulador está associado a custos baixos e acesso remoto. Sendo o seu custo baixo facilmente será possível replicar o sistema com poucas alterações, e até criar um sistema que interligue várias unidades deste tipo;
- Com este sistema é possível melhorar as capacidades do ensino uma vez que é adaptável a diferentes tipos de processos e são objectos de pequenas dimensões;
- Se correctamente configurado é possível aumentar a capacidade de experimentação e ensino em qualquer sala de aula e ainda é possível realizar testes para aprendizagem em casa, sem influenciar o sistema real, que poderá estar ou não em funcionamento.

Em ambas as vertentes é necessário considerar as suas valiosas características. Tais como:

- Simulador apresenta baixo custo, ou seja, é facilmente adquirido em todos os âmbitos;
- Apresenta tamanho pequeno, logo é facilmente implementável em todos os ambientes;
- Independência de licenças de *software*. Este também permite replicar o sistema facilmente e ter vários modelos de processos físicos a correr em paralelo gerando assim um elevado grau de versatilidade.

1.1-Motivação

Tendo em conta o contexto da pandemia é necessário pensar em formas de poder leccionar de forma remota. Actualmente existe uma massificação de acesso à rede, o que facilita a criação de sistemas controlados via rede.

Neste trabalho práctico pretende-se desenvolver um laboratório de modo que seja possível simular processos reais, controladores e realizar testes nos mesmos. Ao colocar este sistema na rede *Internet* potenciam-se as capacidades dos processos simulados, que passam a poder ser controlados remotamente também. Isto promove a possibilidade de desenvolver um laboratório, de dimensões reduzidas e baixo custo, como referido anteriormente [3].

1.2-Objectivos

O objectivo desta tese é desenvolver um sistema didáctico de emulação de processos reais. O sistema deve permitir controlo remoto tanto de processos reais como de processos simulados.

Este sistema didáctico quando em simulação, o modelo dinâmico do sistema simulado deverá poder ser seleccionado por um administrador. A implementação de novos modelos para simulação deve ser possível sem significativo esforço de aprendizagem por parte do administrador.

Para atingir os objectivos supracitados, será desenvolvida uma estrutura modular, em que cada módulo visa ser independente dos restantes, permitindo assim a fácil alteração do sistema, adição de novos módulos que consintam mais funcionalidades ao sistema, ou utilização do sistema noutro processo de controlo, entre outros. Para isso será desenvolvida uma arquitectura de raiz, utilizando ferramentas de software de licença aberta, seleccionar hardware de baixo custo com capacidade de executar as operações necessárias e de menor custo possível.

É de notar que um objectivo fundamental para o sucesso do sistema, é ser possível fazer testes no modelo de simulação, armazenar os dados a longo prazo numa base de dados que tem de ser implementada sem qualquer tipo de influência no modelo físico definido pelo administrador do sistema. Também se pretende a implementação do sistema de autenticação para utilização do simulador.

Com esta dissertação pretende-se obter uma unidade de simulação com determinadas características e capacidades bem de definidas, tais como:

1. Protótipo funcional. Este protótipo inclui a implementação de um sistema que permite controlo remoto, e simulação de modelos de instalações.
2. Acesso a redes locais com adaptador *Wi-Fi* ou porta *Ethernet*, com a capacidade de correr equações de controlo em tempo real para modelos físicos
3. Deve ter portos digitais e analógicos para controlar directamente processos reais, e conter um simulador que possa carregar diferentes modelos de simulação.
4. Deverá também ter a capacidade de ser dotado de um sistema de bases de dados e um modo de visualização dos dados.
5. Um sistema de comunicação por mensagens, com formato próprio para comunicação entre os sistemas.

1.3-Contribuições

A elaboração deste sistema de controlo apresenta diversas contribuições salientam-se o facto de:

- Ser desenvolvida uma arquitectura de raiz. Permitindo assim moldá-la aos objectivos citados (Secção 1.2**-Objectivos);**
- Existir a construção e implementação de um protótipo de *Software*. Ser possível analisar os resultados em ambiente real;
- Ser desenvolvido um sistema que permite controlo a remoto;
- Implementar um simulador de modelos físicos;
- Ser criado um sistema de mensagens com formato próprio;
- Utilização de várias linguagens versáteis e gratuitas, desta destaca-se o *Python* (linguagem escolhida para desenvolver o sistema que apresenta diversas mais valias para se alcançar os objectivos).
- Com menor impacto também é utilizado um sistema "cliente" para interagir com o sistema (realizar testes e para validação da implementação do sistema)

Actualmente reconhece-se mundialmente que a linguagem *Python* é uma ferramenta muito versátil e com amplas capacidades de desenvolver sistemas. Enumera-se agora algumas das suas vantagens nesta aplicação prática:

- Permite uma implementação simples;
- Caracterizada por possuir uma série de bibliotecas dedicadas aos vários tipos de operações;
- Deter acesso à rede;
- Possibilidade de deter uma base dados e realizar a sua gestão;
- Acesso a entradas e saídas físicas;
- Manipulação de equações de controlo (revela-se uma mais-valia pois possibilita simplificar e organizar todo o processo, o que se traduz em termos de resultados, num sistema simples e eficaz de se utilizar).

1.4-Organização

Este documento encontra-se organizado da seguinte forma:

- Inicialmente analisou-se o que existe disponível e o que poderíamos utilizar para fazer o proposto no Capítulo 2- **Estado de Arte**;
- A arquitectura do sistema bem como os seus módulos são descritos no Capítulo 3 – **Arquitectura do sistema**;
- A implementação do sistema e a sua descrição (de quais os módulos, como foram construídos e as dificuldades encontradas durante a construção do mesmo) detalhada é apresentada no Capítulo 4- **Implementação**;
- Os resultados são enunciados no Capítulo 5- **Resultados**;
- As conclusões desta dissertação são expostas no Capítulo 6 – **Conclusões**.

2

Estado de Arte

Neste capítulo são descritas as alternativas em termos de tecnologias disponíveis para atingir o objectivo pretendido com esta dissertação. São discutidas as várias ferramentas e são analisadas as potencialidades de cada uma.

2.1 Simuladores e bibliotecas de simulação

Actualmente existem várias ferramentas que permitem desenvolver sistemas de simulação remotos. Das várias ferramentas disponíveis vão ser analisadas as seguintes: *Matlab*, o *Labview* e o *Python*. Sendo estes os mais conhecidos e divulgados, mas existem ainda muitas outras ferramentas que poderiam ser analisadas.

MATLAB/SIMULINK

O *Matlab* é uma plataforma de programação desenhada especificamente para ciência e engenharia, que permite analisar e desenvolver sistemas. Permite examinar dados, desenvolver algoritmos e criar modelos utilizando uma linguagem própria baseada em matrizes e não só. Associado ao *Matlab* existe uma ferramenta de simulação extremamente eficiente denominada *Simulink* [4].

O *Simulink* possui um conjunto de ferramentas agrupadas por áreas de Ciência, gera um interface gráfico de fácil interpretação por diagramas de blocos personalizáveis, simplificando assim a sua aprendizagem e programação.

Na área de desenvolvimento de controladores já existem alguns simuladores utilizando as referidas ferramentas. No contexto educativo tem-se por exemplo um laboratório com acesso por vídeo, onde se consegue ver o processo

instalado a correr e a simulação a ser demonstrada num sinóptico ou um controlador para aplicação em motores de corrente contínua [5].

O *Matlab* e o *Simulink* têm a capacidade de funcionar num sistema embebido/plataforma, como o *Raspberry Pi*. O *Simulink*, em particular, consente a implementação de sistemas de controlo instalando uma instalação no *Raspberry Pi* como uma "caixa negra". Isto é, um sistema que em que não se sabe o que está lá dentro ou como funciona, apenas se sabe o que está implementado, e onde se encontram as saídas perante as entradas dadas ao sistema, ou seja, o desenvolvimento do projecto tem de ser realizado num computador em *Matlab*/*Simulink* e depois implementado nestas plataformas.

Este tipo de implementação, como referido no parágrafo anterior, também pode ser desenvolvido noutras plataformas como no *Beaglebone Black* e *Blue* ou *Arduino*, e é similar em todas nas referidas plataformas [6][7].

As plataformas enunciadas anteriormente são compatíveis com o *Matlab* e *Simulink* através do pacote de suporte para cada uma delas, que é disponibilizado pela *Mathworks*. O pacote admite, por exemplo, que dois sistemas que comuniquem entre si como semáforos e sensores utilizando um Arduíno e um Raspberry pi[8].

Os diversos pacotes do *Simulink*, admitem de modo autónomo, correr modelos nas respectivas plataformas. Estes pacotes contém as bibliotecas do *Simulink*, facultam acesso físico aos portos das placas e também alteram valores em tempo real no caso de algum utilizador estar ligado ao *hardware*[9].

O pacote de suporte do *Simulink* possibilita desenvolver sistemas controlados pela rede, tendo sempre em consideração a qualidade de serviço da rede. Também é possível realizar autorizações para sincronizar as simulações entre um cliente e um simulador automaticamente[10].

Como principais funcionalidades de interacção da biblioteca do pacote de suporte com o *Raspberry P*i tem-se:

1. Acesso directo aos portos físicos de entrada e saída;
2. Acesso directo a protocolos de comunicação *I2C*, *UART-RS-232* e *SPI*;
3. Saídas com modulação de largura de pulsos (*PWM*).

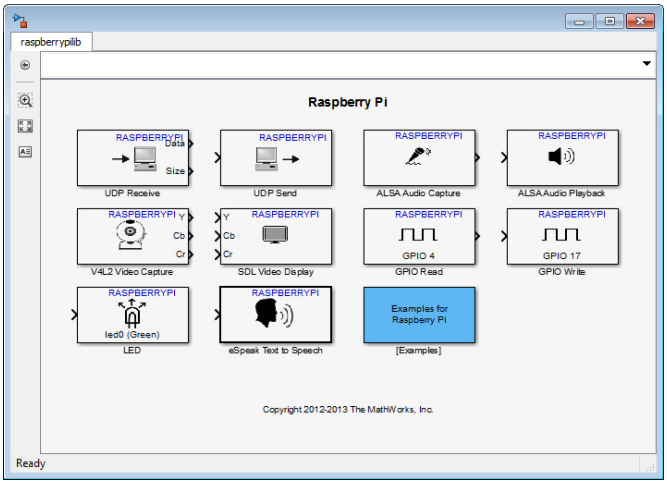


Figura 1 - Biblioteca do pacote de suporte do Raspberry Pi para o Simulink. Reproduzido de [11].

Na Figura 1 visualizam-se algumas das utilidades disponíveis no módulo do *Simulink*, que combinando com linguagem de programação *Matlab* torna-se uma ferramenta poderosa. A combinação destes também apresenta desvantagens como:

- Combinação extremamente dispendiosa (licenças não gratuitas e com preços elevados). O *Matlab* e o *Simulink* são ferramentas proprietárias, tendo por isso elevados custos associados. Para obter uma licença para utilização do *Matlab* e *Simulink* ter-se ia um custo entre 250€ e 800€ anuais, ou 500€ e 2000€ para perpétuas no caso de licenças académicas ou pessoais[12].A faculdade actualmente não tem licenças desta ferramenta.
- Com modelos demasiado complexos no *Raspberry Pi*, os resultados começam a atrasar devido as suas capacidades de *hardware*.

Actualmente, o *Simulink* consegue implementar estratégias para corrigir os atrasos temporais provocados pelas várias variáveis como o tempo de atraso de rede e o tempo de processamento do simulador[13]. Estas correcções permitem o sincronismo dos tempos de amostragem para melhorar o desempenho do simulador. O *Matlab* quando contido no *Raspberry Pi* permite fazer experiências de controlo e criar laboratórios portáteis[14].

LABVIEW

O *Labview* (Laboratório Virtual de Instrumentação em Engenharia) é uma ferramenta de programação por interface gráfico, baseada no conceito de programação por fluxo de dados, da *National Instruments* (NI). Este paradigma foi largamente adoptado pela indústria, comunidade académica e laboratórios de pesquisa, como o standard para aquisição de dados e software de instrumentação e controlo[15].

Como se pode observar nas Figuras 2 e 3 é orientado para programação por blocos e permite desenhar interfaces gráficos intuitivos, reduzindo o tempo de implementação de um projecto de controladores e o desenvolvimento dos seus interfaces gráficos.

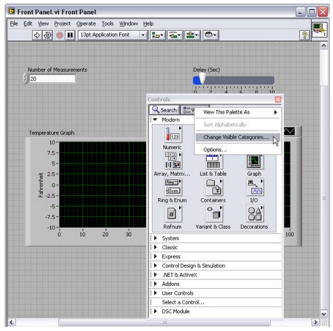


Figura 2 - Painel principal de Labview. Reproduzido de [16]

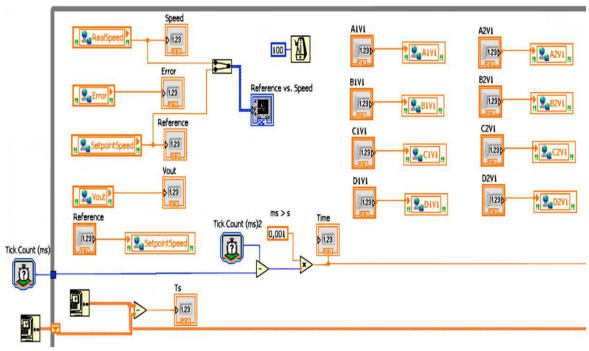


Figura 3 - Exemplo de projecto de Sistema controlado pela rede com o Labview. Reproduzido de [17]

O *Labview* tem a capacidade de ser executado nos sistemas operativos, *Windows*, *macOS*, *Linux* [18].

Esta ferramenta requer elevados recursos de computação, mas beneficia de elevadas capacidades para desenvolver sistemas controlados pela rede coexistindo na sua própria rede através de simuladores ou então pela própria rede[19]. Pode ser visto como um programa com uma interface de utilizador intuitivo e destacam-se as seguintes características:

- Facilita a aquisição de dados a partir de entradas físicas;

- Permite o processamento em paralelo;

- Facilita construção de interfaces gráficos;

- Facilita comunicação por portas físicas Ethernet, USB e ligações específicas para instalações de controlo;

- Possuí interfaces gráficos para visualização dos estados dos processos e resultados em tempo real.

- Suporta através da instalação de pacotes a utilização em sistemas embebidos como *BeagleBone, Arduino* e *Raspberry Pi*[20][21].

O *Labview* consegue comunicar com bases de dados, mas dificilmente consegue executar operações com os mesmos. Consequentemente a implementação de um sistema servidor-cliente com fácil acesso e interacção com a base de dados torna-se difícil. Mesmo implementando um sistema deste tipo apenas é possível realizar supervisão não permitindo controlar sistemas pela rede. Mais ainda se acrescenta que não detém portabilidade para plataformas móveis de *iOS* e *Android*[22] .

Com o *Labview* e o *Raspberry Pi* é possível desenvolver sistemas embebidos, utilizando a ferramenta *Makerhub Linx* da NI que facilita a comunicação do Labview com o Raspberry pi e outras plataformas de sistemas embebidos[23]. A licença de *Labview* com o sistema de processamento de sinais tem um custo anual de 2987 €[24], o que pode ser uma desvantagem na indústria.

Python

O *Python* é uma linguagem relativamente simples de aprender. Permite a interligação de vários tipos de dados e fazer operações com os mesmos, como modelos de processos físicos e controladores, fáceis de trabalhar[25]. Por curiosidade foi nomeada de *Python* pela sátira dos Monthy Python[26]. *Python* foi

desenvolvido em 1989 por Guido Van Rossum e é considerada a linguagem de maior crescimento dos últimos tempos [27].

É uma linguagem interpretada, de alto nível. Foi desenhada para resolver problemas com matrizes, a partir de uma extensão de linguagem *C*. O passo seguinte foi desenvolver a biblioteca *NumPy* que potencia trabalhar com estruturas de vectores multidimensionais em linguagem de alto nível, e definida com uma estrutura para computação numérica eficiente. De seguida foram desenvolvidas, no meio dos anos noventa, a biblioteca *Sympy* para trabalhar com modelos matemáticos que se desenvolveu para a biblioteca *SciPy*, e a biblioteca *MatplotLib* que lhe concedem quase as mesmas característica que o *Matlab* [28].

É uma linguagem baseada em classes, funções e tuplos e foi desenvolvida inicialmente para executar tarefas rotineiras em servidores [29].

As funcionalidades que tornam esta linguagem atractiva, intuitiva e fácil de rever devem-se:

a) Suportam múltiplos paradigmas de programação, incluindo programação orientada a objectos;
b) Conter um sistema dinâmico de alocação de memória;
c) Possuir diversas bibliotecas e normas;
d) Existirem vários interpretadores *Python* disponíveis para os diversos sistemas operativos (como o *Thonny* para *Raspbian*, o *Anaconda* ou *Pycharm* para *Windows* ou *MacOs, Pydoid3* para *Android*);
e) Elevada capacidade de integração de muitas outras linguagens [27];
f) Destaca-se ainda que esta linguagem sabe enviar e receber pedidos da *web* e interagir com bases de dados.

O *Python* dispõe ainda de todas as ferramentas necessárias para lidar com sistemas controlados pela rede, simuladores de controlo e bases de dados, criar interfaces gráficos.

Esta poderosa ferramenta, para além das vantagens salientadas anteriormente dispõem de licenças gratuitas para toda a população. O facto de ser gratuito torna-a atractiva para a população em geral. Aliado à atracção a comunidade de programadores tem se expandido e desenvolvido muito, criando novas bibliotecas diariamente. Para alem disso devido à sua expressão na comunidade existe muito suporte técnico como por exemplo: partilha dos seus problemas em fóruns na internet, vídeos e páginas *Web* dedicadas a esta linguagem.

Aliado aos factores referidos anteriormente é ainda importante salientar que a comunidade académica e de conhecimento beneficia da utilização desta linguagem, pois actualmente é uma das mais utilizada (tornou-se na linguagem mais utilizada, e foi denominada da linguagem do ano em 2018 pelo IEEE). Por estes motivos é útil para os alunos interagir de início com ela para que se consiga manter a aprendizagem actualizada com o standard global de hoje [30].

O *Python* pode ser utilizado nas mais diversas áreas como:

-Realizar um sistema de segurança doméstica embebido no Raspberry pi para controlo de abertura de portas baseado em reconhecimento facial [31];

- Criar simuladores de sistemas celulares biológicos [32];

- Desenvolver plataformas de simulações com robots [33].

Para alcançar os objectivos, no caso do *Matlab* e *Python*, recorre-se a algumas ferramentas que não são nativas das linguagens. É, portanto, necessário estudar as bibliotecas, as funções e os métodos, que permitem adicionar as funcionalidades requeridas.

Qtdesigner

O *Qtdesigner* é um ambiente gráfico para criar aplicações gráficas sem ser necessário programar "linhas de código", simplesmente utilizando programação orientada a objectos gráficos, ou seja, através de uma interface gráfica e diagrama de blocos, como se pode observar na Figura 4 [34].

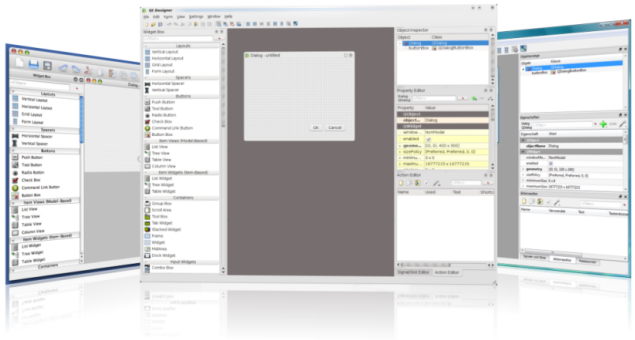


Figura 4 - Software de criação de interface gráfico QtDesigner5. Reproduzido de [35]

O *Qtdesigner* pode ser utilizado nos sistemas operativos *Windows* e *Linux*, esta versão não requer nenhuma licença paga. Este ambiente gráfico foi desenvolvido para ser utilizado com linguagem C embora o Python tenha bibliotecas disponíveis que utilizam *Qtdesigner* de forma eficiente.

Existem experiências bem-sucedidas de simulação de modelos que demonstram que é possível realizar simuladores em *Python* com o *Qtdesigner*. Como exemplos desta aplicação, temos:

- A Implementação de uma central térmica, em que primeiramente foi desenvolvido o modelo matemático e só depois o simulador [36].
- Execução de um modelo de condições ambientais que responde perante condições meteorológicas, podendo prever colheitas agrícola e simular os resultados para diferentes condições [37].

Após os diversos sucessos deste tipo de implementação desenvolvida através de simuladores, conclui-se que é de encorajar o uso de programas orientados a objectos

Flask

Uma alternativa viável para criar interfaces gráficos que não sejam locais em relação ao servidor é desenvolver interfaces gráficos em páginas *Web* utilizando micro-serviço de *Python* como o *Flask*. O *Flask* permite o lançamento de páginas de rede é ainda possível aceder às bases de dados permitindo criar outro modo de interface gráfico.

2.1.1- Bibliotecas

No caso do *Matlab\Simulink*, para adicionar funcionalidades para implementação de um sistema de simulação, basta descarregar o pacote de suporte do *Simulink* e o pacote conector *Java Data Base Connectivity* (JDBC). O último permite executar ligações a bases de dados.

- ***Labview:*** Como referido anteriormente é uma ferramenta muito completa que contém as bibliotecas necessárias para o funcionamento de simuladores de controlo.
- ***Python:*** Pelas características enunciadas anteriormente, esta linguagem será uma das linguagens a utilizar. Para o desenvolvimento das diferentes funcionalidades são necessárias bibliotecas, das quais se destacam:
  - *Threading*- permite tarefas em paralelo, potência a capacidade de correr processos em concorrência;
  - *Socket* - esta biblioteca gera o acesso, a programas de *Python*, à rede;
  - *Psycopg2* - concede a um programa de *Python* a capacidade de comunicar com as bases de dados;
  - *Scipy. solve*- método numérico de resolução de equações diferenciais
  - *Configparser* - para criar e carregar ficheiros de configuração genéricos.
  - *GpioZERO*- biblioteca que permite ligar o *Python* as entradas e saídas físicas do *Raspberry Pi*

Neste ponto serão desenvolvidos alguns detalhes sobre as mesmas:

i) ***Threading:***
- Uma biblioteca nativa de *Python*, permite correr várias tarefas em concorrência, possibilitando assim, por exemplo, colocar um servidor a enviar e receber mensagens e ao mesmo tempo estar a correr uma instalação real e um modelo de simulação;
- Esta biblioteca corre as tarefas todas na mesma aplicação não sendo possível seleccionar o processador onde é executada. Ou seja, aplica-se a programas que correm em sistemas com um processador apenas (não permite a escolha do processador para correr a tarefa);
- Existe uma alternativa a esta que é a biblioteca *Multiprocessing* que também dota o sistema com tarefas que correm em paralelo, mas que podem ser distribuídas pelos diferentes processadores, da unidade de hardware, escolhidas na implementação;

– Os estudos revelam que para utilizar em microcomputadores como *Raspberry Pi*, ou *BeagleBon*e se obtém melhores performances com a biblioteca *threading* [38].

ii) <u>Socket:</u>

– Produz uma interface de rede de baixo nível;
– Criado como um objecto, cujos métodos são os tradicionais de comunicação por redes de computador ao nível de interfaces de linguagem C;
– Apresenta algumas funcionalidades, incluídas, de comunicação. Para se poder enviar uma mensagem, basta definir a família de *socket*, o tipo de rede e depois escolher o método enviar, ou receber, para podermos comunicar. Esta biblioteca contém várias famílias de *socket* disponíveis, sendo as mais comuns:
  - *AF_UNIX* que fica associado a um endereço de *Linux*,
  - *AF_INET* que gera um endereço no domínio da *Internet*,
  - *AF_BLUETOOTH* concede comunicação em *Bluetooth* com um programa em *Python*.
– Para além de ter as famílias já vem com as funcionalidades que permitem aceder as camadas de baixo nível, e de modo a estabilizar as comunicações já tem integrados um conjunto de excepções configuráveis. Um exemplo destas é uma excepção que faz com que se a ligação estiver aberta um tempo definido sem actividade esta feche, libertando assim recursos para a rede e processo a controlar.[38] Como exemplo temos também a possibilidade de ser utilizada para colocar sensores e actuadores em campos agrícolas a comunicar com telefones móveis e com a *Internet* [39].

iii) <u>Psycopg2:</u>

– É o adaptador de bases de dados mais popular para *Python*;
– Tem como missão criar uma ligação a uma base de dados *PostgreSQL*. Como principais características ter implementadas as especificações da interface programável de aplicação para bases de dados (*Python* DB API2.0). Com esta consegue-se manter uma ligação de bases de dados e partilhá-la com diferentes tarefas de um programa em execução;
– Vê-se como mais-valia o facto de aceitar todos os tipos nativos de *Python*, e aceita manipulação e dados e da base de dados para criar

tipos de dados próprios se necessário[40]. Este é por exemplo utilizado num sistema que executa radioterapia, quando o controlo do operador da sessão de terapia tem que estar afastado por segurança do paciente e dos operadores, e utilizada uma base de dados *PostgreSQL* e um conector *Psycopg2*, para interligar o processo terapêutico com o seu controlo através da base de dados[41];

- É também utilizado para testar sistemas de armazenamento de imagens de satélite, com processamento de pontos geospaciais em formato de vectores, para poder interligar as imagens, e devolver um mapa com o processamento das mesmas com o *psycopg2* e o *PostgreSQL*[42].

iv) SciPy.Solve:

- Para implementar modelos matemáticos de processos que consigam demonstrar e testar os sistemas de controlo são necessárias ferramentas para resolução de problemas de equações diferenciais;
- Assim considera-se a ferramenta *solve_ivp* do pacote de integração da biblioteca *SciPy*. O *SciPy* é uma biblioteca de rotinas numéricas, que providencia uma série de blocos fundamentais para criar modelos matemáticos e resolver problemas científicos. O *SciPy* inclui algoritmos de optimização, integração, interpolação, problemas de valores próprios, equações algébricas e outras classes de problemas. Também consente a utilização de estruturas de dados especializadas como matrizes esparsas e árvores de k-dimensões. O *SciPy* é baseado na biblioteca *NumPy*, que fornece estruturas de vectores, e as rotinas que permitem fazer operações entre estas [43];
- Uma vez que se pretende realizar um modelo de instalação para testar o simulador e os seus componentes, um método adequado que se pode utilizar é *RungeKutta*. Este tem uma família de algoritmos que se pode utilizar, o selecionado foi algoritmo *RungeKutta* iterativo de quarta ordem. Este permite o cálculo integral com um elevado grau de precisão, e requer pouco poder de processamento [44];
- Actualmente já existem algumas experiências com instalações termodinâmicas [45] e bioquímicas onde se conclui que o *Matlab* é mais preciso, mas o *Python* é mais rápido a realizar as iterações especialmente em sistemas com pouca capacidade de computação. Assim este método em *Python* considera-se o ideal para realizar um modelo de testes.

v) ***Configparser:***

- Concede à linguagem *Python* a possibilidade de criar e ler ficheiros de configuração, com uma estrutura similar à dos ficheiros em *Windows* com o formato INI. Permitindo assim facilmente alterar as configurações de qualquer programa *Python*;
- É compatível com dicionários de *JSON(Javascript Object Notation)*[46]. Tendo assim a possibilidade adicionar secções, o que se torna útil por exemplo quando módulos diferentes do programa carregam simulações;
- Esta biblioteca tem parâmetros de configuração próprios que permitem criar ficheiros de configuração estruturada pelo programador da aplicação, que consegue comunicar com outras linguagens de programação [47];
- Também se torna muito útil à sua utilização, uma vez que já consegue importar ficheiros de configuração com comentários, o que os torna mais legíveis para quem configura o sistema.

vi) GPIO ZERO:

- Consegue o acesso aos portos físicos de um microcomputador, garantindo assim que podemos receber enviar e receber dados de modo digital;
- Também consegue comunicar por SPI (Serial Peripheral Interface) e I2C (*Inter Integrated Circuit*), e com estes adicionar conversores analógico-digitais para receber sinais analógicos com baixa latência[48];
- Disponibiliza uma série de funções básicas para poder realizar ligação a alguns sensores e circuitos integrados. Existem já vários estudos que utilizam esta biblioteca para fazer por exemplo detecção de movimento com um sensor[49], ou para realizar activação ou desactivação de portos com *relays*, ou aquisição de dados [50].

2.2-Modelos Físicos e Matemáticos de Sistemas NCS

Os modelos de estado transpõem as diversas áreas da ciência nomeadamente a biologia, a economia, as finanças e especialmente as áreas de engenharia.

O controlo assenta em dois pilares, a capacidade de analisar a natureza dinâmica dos sistemas e capacitar-nos de métodos para modificar o desempenho de sistemas. Um sistema de controlo é uma ferramenta ou um conjunto de ferramentas para gerir, comandar, dirigir ou regular o comportamento de outros dispositivos ou sistemas [51].

Na generalidade define-se um sistema dinâmico como uma estrutura organizada de transferência de um sinal de entrada u para um sinal de saída y. À função que representa o sistema chamamos de modelo matemático.

Os sistemas são dinâmicos se as suas saídas actuais dependerem das entradas ou saídas passadas. Nos processos de controlo há uma enorme vantagem em conhecer a dinâmica do sistema, uma vez que nos autoriza replicar os sistemas físicos em modelos matemáticos ou de *software* num computador, ou microcomputador.

Durante muitos anos os investigadores encontraram estratégias de controlo óptimo que emergem da teoria de controlo clássico. No entanto o surgimento das redes de telecomunicações gerou o conceito de controlar remotamente um sistema, que deu a origem aos sistemas controlados via rede (NCS) [51].

2.2.1 – Modelos Físicos e modelos matemáticos de instalações

Existem os modelos físicos reais que representam os processos reais ou a instalação, e também existem modelos matemáticos dos mesmos que permitem simulações sem influenciar a instalação (que pode, ou não, estar a operar).

Um modelo matemático pode descrever as características dinâmicas dos sistemas, possibilitando assim prever o comportamento dos sistemas antes de estes serem executados, o modelo também pode ter como propósito elaborar o projecto do controlador [52].É de notar que o modelo matemático é uma representação do sistema físico, assim este modelo não tem todas as características do modelo físico tratando-se de uma aproximação. Mas tal é suficiente para se poder testar e simular o sistema físico dentro da validade do modelo matemático. Ao desenvolver um sistema físico-matemático deve ter-se em conta a sua simplicidade versus a sua exactidão [52].

Os modelos físico-matemáticos admitem conceber uma base de análise de sistemas (não interessa a origem tecnológica). Um bom exemplo desta funcionalidade é quando o mesmo modelo matemático apresenta diferentes constantes associadas e, por isso, pode definir tanto um sistema massa, mola e amortecedor, assim como um sistema com resistor, indutor, condensador em série. Estes

modelos normalmente são baseados em equações às diferenças que definem as diferentes forças, caudais, velocidades e aceleração em cada instante do tempo, para o caso de sistemas em tempo discreto.

2.2.2- Controlo por retroacção

O controlo prende-se com a capacidade de alterar o desempenho dinâmico de sistemas. Tipicamente uma instalação (sistema físico) é fixa, por exemplo, um processo industrial, uma fábrica ou um motor eléctrico. O controlador é um sistema externo à instalação que nos permite implementar o comportamento dinâmico pretendido. As componentes da instalação quando acopladas as do controlador obtém-se um sistema de controlo. Na Figura 5 exemplifica-se um dos tipos de controlo: controlo por retroacção [52].

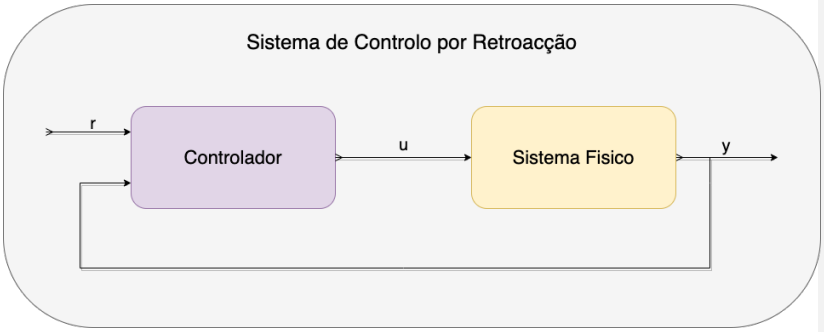


Figura 5 - Sistema de controlo por retroacção.

O sistema de controlo por retroacção, descreve as funcionalidades de um sistema de controlo por retroacção, este tipo de sistemas funciona de modo a alterar a dinâmica do sistema para que este tenha o comportamento desejado tendo em consideração a entrada *u* e a saída *y*. Salienta-se que o controlo se exerce sobre a instalação a controlar.

2.2.3-Sistemas controlados via rede

Uma definição clássica de *Networked Controled Systems* (NCS) diz que num sistema de controlo em anel fechado o fecho do anel é sempre feito por um canal de comunicação (canal que pode ser partilhado por outros nós que não se encontram no sistema de controlo) [53]. Um NCS também pode ser definido como um sistema de controlo por realimentação onde o anel de controlo é fechado por uma rede em tempo real [51].

Concluindo, aquilo que define NCS é que a informação (entrada de referência, saída da instalação, entrada de controlo, etc.) é trocada utilizando uma rede entre os componentes do sistema de controlo (Figura 6).

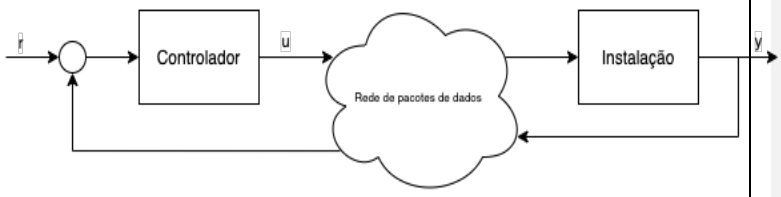


Figura 6 - Exemplo de sistema controlado via rede. Adaptado de [10].

A massificação da utilização das redes de comunicação conduziu a um aumento do número de redes domésticas, académicas, empresariais e militares, que juntas transportam informação por todo o mundo. Nos últimos anos tem havido um enorme crescimento na implementação de sistemas com redes sem fios, que naturalmente despoletou o desenvolvimento e pesquisa de NCS distribuídos.

Os NCS distribuídos, em geral, estão definidos por sistemas que comunicam com sensores, actuadores ou simuladores, cuja operação é controlada remotamente e coordenada de algum modo via rede. A rede nestes casos é um factor decisivo na operação do sistema. De outro modo, a rede de comunicação pode ser utilizada no sistema de supervisão para poder aceder ao controlador remotamente [17].

Os NCS distribuídos reduzem os custos de instalação e manutenção, uma vez, que removem elevadas quantidades de cablagem e geram a possibilidade de existirem controladores descentralizados [54][55].

Na Figura 7 pode-se observar os sistemas:

-Sistema supervisionado pela rede, onde se pode observar e aceder ao controlador, mas o controlador e a instalação estão directamente ligados;

-Sistema controlado pela rede, onde a instalação e o controlador não se encontram ligados fisicamente.

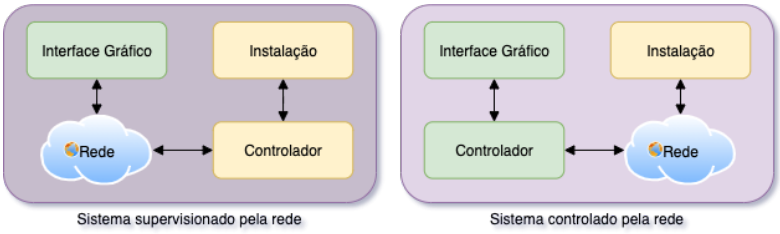


Figura 7 - Exemplos de sistemas e controladores a interagir via rede.

Actualmente existem algumas questões que devem ser abordadas no domínio de sistemas controlados via rede. Estas questões prendem-se com:

- A introdução de uma rede de comunicações que tem a sua capacidade máxima de carga, velocidade, atrasos por pacotes perdidos. Assim quando se executa controlo via rede é aconselhável definir estratégias com observadores do lado do controlador (de modo a conseguir obter controladores estáveis e úteis).
- Se não existirem algoritmos de correcção destes atrasos, o sistema em anel fechado pode tornar-se instável [56], [57]. Quanto maior for a distância entre o controlador e a instalação, os atrasos acumulam-se (maior o atraso que encontramos no processamento de uma acção e consequentemente uma resposta de instalação com atraso) e também poderá haver perda de pacotes.

2.3- Unidades de computação

Actualmente há no mercado dispositivos disponíveis de pequenas dimensões, com características similares. Nos próximos subcapítulos é apresentado um breve estudo de alguns desses dispositivos.

2.3.1- Raspberry Pi

O *Raspberry Pi* (Figura 8) é um computador do tamanho de um cartão de crédito que se pode conectar à *Internet*, com saída de vídeo que se pode ligar um monitor, contendo várias entradas e saídas digitais.

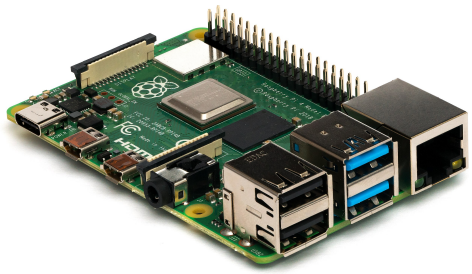


Figura 8 - Microcomputador Raspberry Pi 4. Reproduzido de [58].

O *Raspberry Pi*, poderá ser também utilizado como computador de secretária com acesso à *Internet*, ou utilizado num laboratório que interage com o mundo exterior (por exemplo a leitura de sensores e escrita em actuadores) pelos seus portos *GPIO*. O *Raspberry Pi* tem também a capacidade de computação através da linguagem *Python* ou *C++*, a possibilidade de acesso a dispositivos externos por *Wi-Fi* ou *Bluetooth*, executa servidores *web* com páginas e através destes possibilita elaborar sistemas de controlo com o *Raspberry Pi* (Tabela 1) [3][59].

	Raspberry Pi 4	Raspberry Pi 3 B+
Sistema num processador(SoC)	Broadcom BCM2711, Quad coreCortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	Broadcom BCM2837B0,quad-core A53 (ARMv8) 64-bi @ 1.4GHz
GPU	Broadcom VideoCore VI	Broadcom,Videocore-IV
RAM	2 GB, 4 GB, or 8 GB LPDDR4 SDRAM	1 GB LPDDR2 SDRAM
Bluetooth	Bluetooth 5.0, BLE	Bluetooth 4.2, BLE
Portas audio e video	2 × micro-HDMI 2.0, ficha 3.5 mm analógica audio-vídeo	Full size – HDMI, 3.5 mm ficha 3.5 mm analógica audio-vídeo
USB	2x USB 3.0 + 2x USB 2.0	4x USB 2.0
Ethernet	Gigabit Ethernet Nativo	300 Mbps Giga Ethernet
Descodificador Video	H.265 4Kp60, H.264 1080p60	H.264 & MPEG-4 1080p30
Fonte de Alimentação	5V via USB Tipo-C até 3A, via GPIO até 3A	5V via micro USB até 2.5A, via GPIO até 3A
Nr. de pinos GPIO	40	40
Funcionalidades GPIO	17 x GPIO, UART, I²C, SPI, I²S, 1-Wire, 3.3V/5V/GND, EEPROM, PWM	17 x GPIO, UART, I²C, SPI, I²S, 1-Wire, 3.3V/5V/GND, EEPROM, PWM
Wifi	2.4 GHz and 5 GHz 802.11b/g/n/acwireless LAN	2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN
Armazenamento	cartão microSD	cartão microSD
Preço	45€ – 2 GB RAM	37€ – 1 GB RAM
	62€ – 4 GB RAM	
	87€ – 8 GB RAM	

Tabela 1 – Tabela comparativa dos dois modelos mais recentes de *Raspberry Pi*

O microcomputador Raspberry Pi pode utilizar um sistema operativo *Linux* com ou sem a modalidade de *Real Time*. Este segundo modo melhora o desempenho no caso de escrita e de leitura nos actuadores e sensores (este sistema operativo, não implica a utilização de interfaces gráficas o que faz com que o *Raspberry Pi* possa libertar mais recursos podendo ter mais poder de processamento disponível) [3],[59].

Já existem versões mais actualizadas que o *Raspberry Pi 3*, com melhores capacidades, mas com um preço superior [60]. O *Raspberry Pi 3* tem um custo de 36.9€ [61]. É de acrescentar que já existem testes realizados que demonstram que o *Raspberry Pi 3* se torna numa ferramenta rápida e prática de utilizar, mesmo quando se adicionam diversos tipos de sensores e linguagens de programação, cada vez mais indispensável no processo educativo da área de ciências e engenharias [50].

2.3.2 – BeagleBone

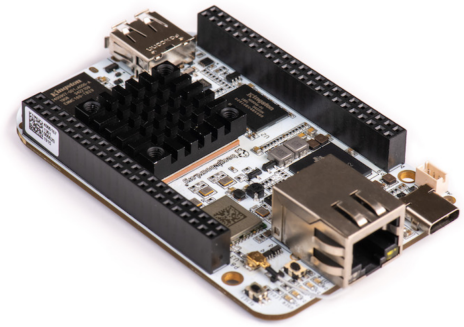


Figura 9 - *BeagleBone* AI. Reproduzido de [62].

O *BeagleBone AI* (Figura 9), é um dos microcomputadores considerado de entre os pequenos computadores de placa única e computadores com elevadas capacidades de processamento. Esta unidade já inclui *software* para processamento de inteligência artificial, contém núcleos de processamento para sinais digitais e é um dispositivo focado para automação em instalações domésticas, industriais e comerciais.

	BeagleBone® AI	BeagleBone® Blue	BeagleBone® Black
Processador	TI AM5729 ARM® Cortex-A15 Dual Core SoC	TI AM335x, ARM® Cortex-A8 32Bit SoC	TI AM335x, ARM® Cortex-A8 32bit
Velocidade de Processador	Dual Core @ 1.5GHz	Single Core @ 1 GHz	Single Core @ 1 GHz
GPU	Dual-Core PowerVR® SGX544 3D GPU Vivante® GC320 2D GPU	PowerVR® SGX530™ 3D GPU	PowerVR® SGX530™ 3D GPU
RAM	1GB DDR3 SDRAM	512MB DDR3 SDRAM	512MB DDR3 SDRAM
Armazenamento	microSD, 16GB eMMC	microSD, 4GB eMMC	microSD, 4GB eMMC
USB	1x USB 2.0 Tipo-A 1x USB 3.0 Tipo-C	1x USB 2.0 Tipo-A 1x microUSB 2.0 Tipo-B	1x USB 2.0 Tipo-A 1x miniUSB 2.0 Tipo-B
Fonte de Alimentação	5V	9V-18V, dependendo da utilização.	5V
GPIO	2x 46 Pinos	Não disponível, acesso por fichas especificas	2x 46 Pinos
Funcionalidades GPO	UART, I2C, SPI, PWM, Temporizador, A/D(3.3V), CAN, Captura de Pulsos	GPS, DSM2 radio, UARTs, SPI, I2C, A/D(1.8V), 3.3V GPIOs	UART, I2C, SPI, PWM, Temporizador, A/D(1.8V), CAN, Captura de Pulsos
Ethernet	1GB Ethernet	Não disponível	10/100MB Ethernet
Wireless LAN	2.4GHz and 5GHz IEEE 802.11 b/g/n/ac WiFi	2.4GHz IEEE 802.11.b/g/n wireless LAN	Não disponível
Bluetooth	Bluetooth 4.2	Bluetooth 4.1 com BLE	Não disponível
Sistema Operativo	Linux	Linux	Linux
Saida de Video	1x microHDMI	Não disponível	1x microHDMI
Preço	€92,75	€91,30	€45,00

Tabela 2 - Características Técnicas de Microcomputadores *BeagleBone*.

Como se pode verificar na Tabela 2 os microcomputadores *BeagleBone* têm preços elevados, por isso, tendo em conta o propósito desta experiência não são adequados. O único que seria adequado seria o *BeagleBone Black*, no entanto as suas características promovem um baixo desempenho no sistema implementado, logo, não se adequa a este trabalho.

2.3.3 – Computador

No desenvolvimento deste estudo havia a possibilidade de utilizar um computador para realizar as tarefas pretendidas. A utilização de um computador aumentaria a capacidade de processamento, muito superior às alternativas apresentadas anteriormente, permitiria até correr vários simuladores, também seria possível que vários clientes se pudessem ligar em simultâneo fazendo assim diversas simulações em sistemas diferentes.

Apesar de todas as vantagens citadas em cima, este dispositivo encontra-se completamente fora do objectivo pretendido neste trabalho, uma vez que iria categoricamente aumentar o custo do sistema, e ainda requeria que fossem adquiridas placas de aquisição de dados uma vez que não tem pinos de acesso directo

para entradas e saídas. Para poder fazê-lo seria necessário adquirir uma placa de aquisição de dados do lado do processo real, para ser possível interligar fisicamente o simulador com o processo real.

2.4-Unidades de conversão de sinal Analógico Digital/Digital Analógico

Normalmente no estudo e controlo de processos reais, que envolvem medidas físicas, estes requerem o envio e recepção de sinais analógicos e digitais. É pertinente estudar algumas opções de conversão analógico-digital para podermos interagir entre o mundo real e o mundo virtual onde o processo de controlo será programado.

Habitualmente os conversores analógico-digital são caracterizados pelo número de amostras por segundo, pela sua resolução, pelo número de canais de entrada e saída, e pelo tipo de comunicação entre eles próprios e outros dispositivos.

Alguns exemplos da sua utilização podem ser encontrados em controlo linear e não linear com um motor de corrente contínua ou um sistema de controlo de tanques[70][71]. Para trabalhar com este tipo de sinais é requerido sempre a conversão do sinal Analógico em Digital (AD) e de sinal Digital em Analógico (DA).

2.4.1- Protocolos de Comunicação de Hardware

A maioria dos microcomputadores de baixo custo não incluem conversores DA, por isso, para suprir esta necessidade deverão ser utilizadas portas físicas no microcomputador. As portas físicas possuem a capacidade de utilização dos mesmos protocolos de comunicação que os conversores. Os dois protocolos de comunicação I2C e SPI são os mais comuns para comunicação entre microcomputadores e sensores, actuadores, conversores de sinal ou via rede.

A Figura 10 ilustra o funcionamento *Master-Slave*, que comunica por I2C em série de dados de um microcomputador. O protocolo I2C, funciona com um dispositivo principal (*Master*) ligado a um conjunto de outros "dispositivos escravos" (*Slave*), estes regem-se pelo relógio do dispositivo mestre e cada um tem um endereço físico de 7 *bits*. A transmissão com cada dispositivo é feita pela linha de dados; seleccionando o dispositivo pelo seu endereço físico, respondendo este com a informação pedida [72]. O I2C é um protocolo de sequência que depende

do ciclo de relógio do dispositivo mestre. É de notar que este protocolo envia e recebe a informação pela mesma linha de dados, logo este é bidireccional. Este módulo deverá permitir o acesso das portas de *GPIO* a outros dispositivos de aquisição de dados.

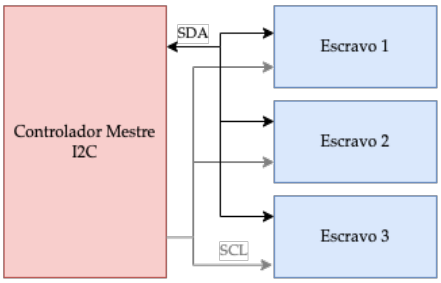


Figura 10 - Arquitectura Master-Slave I2C adaptado de [64]

Existe outro protocolo de comunicação inter-dispositivos, a considerar: o SPI (Interface de periféricos por série) é um protocolo que executa comunicação em tempo real em ambas as direcções. A forma de selecção deste dispositivo requer a activação do mesmo, o que faz com que precise de mais portos físicos.

O SPI difere do protocolo de I2C por solicitar o envio do endereço do dispositivo para comunicar, uma vez que o canal de dados está em espera e não pode estar ocupado com mais informação, o que faz com que este só consiga comunicar numa direcção de cada vez.

Na Figura 11 é descrita a estrutura de ligações SPI. O dispositivo mestre contém as seguintes saídas:

- SCLK - relógio de sincronismo dos dispositivos;

- MOSI (*Master Out Slave In*) – ligação que realiza pedidos ao canal dispositivo "escravo";

-CS – Por cada dispositivo "escravo" é necessário ter esta ligação para o activar.

Verifica-se também que existe uma ligação adicional no dispositivo *mestre*, este tem a ligação *MISO* (*Master In Slave Out*) que recebe os dados pedidos aos dispositivos "escravo".

Analisando a Figura 11, esta permite aferir que a maior desvantagem deste protocolo é requerer uma ligação no dispositivo mestre, por cada dispositivo escravo instalado.

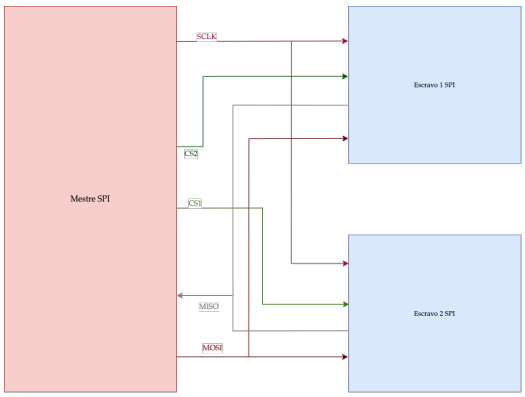


Figura 11 - Arquitectura SPI.

Com estes dois protocolos integrados no sistema podem-se realizar ligações à generalidade dos sensores, actuadores e adquirir dados pelos diversos tipos de dispositivos *escravos* no mercado; ou utilizando conversores analógicos digitais que comuniquem com a unidade central através deste método. Isto faz com que seja possível a aquisição de tensões (sinais) para que depois estas sejam analisadas e processadas. Dotando o sistema desta capacidade de comunicação por estes métodos garante-se que é possível adquirir e trabalhar com a maior parte dos sistemas que se pretendem simular.

2.4.2- ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier

Este dispositivo (Figura 12) consegue converter o sinal analógico com uma amostragem desde 8 a 860 amostras por segundo e possui quatro entradas analógicas no modo independente ou dois canais no modo diferencial. Também beneficia de um comparador que lhe permite detectar sobretensões e subtensões e ainda um amplificador programável com saída entre 2V e 5.5V. O método de comunicação deste dispositivo é por *I2C* [64].

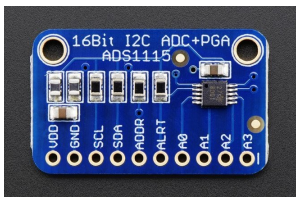


Figura 12 - Conversor analógico digital. Reproduzido de [63].

2.4.3- IC Texas Instruments 74LVC245 converter 3.3 to 5V

A Figura 13 apresenta o conversor linear de 3.3V para 5.5V bidireccional que possibilita se necessário adaptar as tensões entre o sistema e os sensores e actuadores.

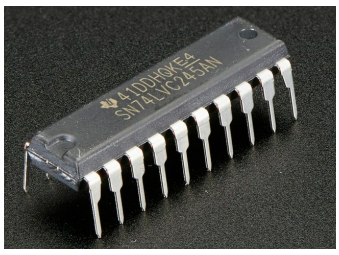


Figura 13 - Circuito Integrado TI 74LVC245. Reproduzido de [15].

Este dispositivo pode ser necessário uma vez que o *Raspberry Pi* trabalha com saídas de 3.3V e há sensores e actuadores nas experiências de controlo que funcionam com 5V .

2.4.4-MCP 3008

O MCP3008, na Figura 14, é um dispositivo integrado da *Microchip*, com objectivo de adicionar funcionalidades de conversão de sinais analógicos em digitais.

O MCP 3008 consegue realizar leituras de 8 entradas analógicas com tensão máxima por entrada regulável através da tensão de operação entre 2.7V e 5V. Este chip tem uma taxa de amostragem máxima entre 75000 e 200000 amostras por segundo, e tem de ser alimentado com uma tensão entre 2.7V e 5V para operar. As taxas de amostragem possíveis de utilizar dependem da tensão de operação definida.

Este equipamento (Figura 14) é amplamente utilizado em aquisição de dados para sistemas de controlo. O conversor é vantajoso pois consegue adquirir dados entradas de 3.3V, ou 5V que são as tensões mais comuns nos sensores e actuadores. Este é um dos mais actualizados actualmente para fazer conversão de sinal com *Raspberry Pi*[23].

Figura 14 - Circuito integrado MCP3008. Reproduzido de [65].

O MCP 4912 é um parente deste circuito integrado, com as mesmas características eléctricas, mas é conversor de sinal digital em analógico para podermos enviar ordens de actuação contínuas. O modelo equiparado, difere apenas no facto de ter duas saídas independentes, em vez de oito entradas [66].

2.5-Armazenamento de dados

De modo a garantir um sistema útil e funcional a longo prazo é necessário pensar num sistema de armazenamento de dados. Pretende-se que este armazenamento possa ser persistente. Assim o modo seleccionado é um sistema de base de dados que garante que conseguimos acautelar a integridade dos dados a longo prazo. Assim iremos ver algumas bases de dados disponíveis como o *MongoDB*, ou o *PostgreSQL*.

MongoDB:

- Projectado para ir ao encontro das aplicações modernas;
- Funciona com modelos de documentos de dados;
- Pode ser utilizado como um sistema distribuído, e é livre para correr em qualquer linguagem. Este armazena os dados com estrutura similar à de *JSON*, o que lhe confere uma estrutura mutável a qualquer altura e com capacidade de agregação de dados em tempo real. É uma base de dados não relacional [67]. Isto faz com tenha uma maior ocupação de espaço de armazenamento físico onde o sistema se encontrar instalado [68].

PostgreSQL:

- Considera-se uma base de dados mais estável, com sistema de limpeza de dados obsoletos;
- Esta equipada de segurança e autenticação;
- Admite todos os tipos dados que são nativos ao *Python*, o que facilita o envio de dados entre um programa a correr e a bases de dados;
- Consegue armazenar dados de forma consistente, e tem uma taxa de acesso em multi-leitura mais rápida que o *MongoDB*, o que é pretendido quando se trabalha em tempo real.
- Tem a possibilidade de ser utilizada com modelos que podem ou não ser relacionais. O facto de se utilizar em base de dados providenciam um armazenamento a longo prazo e consistente.
- Está demonstrado que para o objectivo pretendido que será a base de dados ter capacidade para conseguir fazer múltiplas leituras com um utilizador, o *PostgreSQL* apresenta um melhor desempenho [69].

3

Arquitectura do Sistema

Considerando a experimentação um requisito fundamental de aprendizagem na área de engenharia, e a tendência de ensino que promove a utilização de redes de *internet*, entre outras, este trabalho pretende facultar aos alunos a hipótese de aprender a utilizar controladores remotos.

3.1- Descrição geral do sistema

Nesta tese, pretende-se desenvolver um sistema de simulação híbrida industrial. O termo híbrido pode ser definido como sistema que pode funcionar integrado num NSC, ou como sistema de controlo directo, por *hardware*, através de conversão AD e DA.

Este sistema poderá ter utilizadores externos (ligados ao mesmo de modo que consigam controlar um processo industrial através de um controlador remoto) ou funcionar de modo autónomo.

Para se poder desenvolver a arquitectura do simulador é necessário ter em conta os objectivos estabelecidos. Neste caso pretende-se garantir que é possível um aluno poder estabelecer uma ligação a um simulador que está em funcionamento em tempo real a operar na sua própria escala de tempo, do mesmo modo em que faria com uma instalação real. Esse contexto requer compreender os papéis dos diferentes tipos de utilizadores (Tabela 3).

Administrador-Professor pode:	Cliente-Aluno pode após autenticação
Alterar os modelos físico-matemáticos, que mimicam a instalação industrial a utilizar	Interagir com o sistema desenvolvido por meio de uma rede de pacotes de dados
Consultar os dados de simulações anteriores	Enviar pedidos ao sistema e receber dados dos estados do sistema
Alterar configurações do simulador	Enviar acções de controlo para o sistema
O administrador do sistema terá que ter acesso ao módulo onde se pode encontrar o modelo físico-matemático que representa a instalação; este é de fácil acesso e modificação	Executar simulações de controlo em anel fechado.

Tabela 3 - Acções administrador versus utilizador.

Esta plataforma facilita o ensino porque:

- Não obriga a que um utilizador esteja no mesmo local que o simulador. Assim qualquer aluno com acesso via rede pode executar experiências formuladas por um administrador do sistema.
- O administrador deve poder elaborar os modelos físico-matemáticos da instalação que se pretende testar ou simular com o intuito de providenciar diversas instalações, e estas conterem simuladores de diferentes contextos científicos de aprendizagem.
- Considerando que o sistema deverá armazenar dados a longo e a curto prazo, e ser desenvolvido um processo para tal, com recurso as ferramentas estudadas anteriormente.
- A plataforma deverá conter, dependendo do caso, a ligação bidireccional a um sistema físico em tempo real. Este contexto permite aos alunos estudar o desempenho de sistemas de controlo por eles desenvolvidos em experiências de controlo directo por *hardware*.
- Adicionalmente, este projecto concede aos alunos a capacidade de desenvolver controladores que controlem um sistema por meio de

uma rede, e depararem-se com problemas de atraso de rede, que terão de ser considerados nos seus projectos de controladores.

- Este simulador requer ser de fácil acesso à população, isto é, ao realizar este modelo de instalações controladas via rede, habilitam-se os alunos a estudar problemas de controlo remoto via rede, como atrasos devido a latências da rede. Será possível testar controladores definidos por um aluno, e melhorar os seus algoritmos considerando a rede, já que os sistemas controlados via rede são a direcção actual da indústria tornam se uma mais-valia.
- Pretende-se também que seja possível armazenar os dados dos diferentes estados e acesso no simulador, assim será considerado um sistema de bases de dados, que com as portas de acesso que contém, gerem acesso aos dados por outras linguagens, dando assim o potencial para criar interfaces gráficos para diferentes simuladores, ou por exemplos criar servidores de páginas de internet. Garante-se o armazenamento persistente e estável ao utilizar uma base de dados, evitando assim uma perda de dados, se por exemplo o sistema geral tiver alguma falha, ou para a verificação de resultados posteriormente para professores ou alunos.

Hardware

Assim será necessário um microcomputador com:

- Capacidade de processamento elevada;

- Baixo custo;

-Acesso por *TCP/IP*, seja por *Ethernet* ou *Wi-Fi;*

-Portos de acesso físico entrada/saída.

As características enunciadas, são as fundamentais para cumprir o propósito desta experiência. Este microcomputador terá de correr um sistema operativo leve libertando assim recursos para o sistema a implementar. Deverá também executar uma linguagem de programação rápida e fácil de aprender, que se encontre nas tendências actuais da indústria. Esta linguagem tem de interagir com o *hardware* de rede, para poder comunicar, também têm de ter a capacidade de interagir com os portos de aquisição de dados do microcomputador.

Outra característica indispensável é o microcomputador ter a capacidade para processar equações diferenciais e controladores, conseguir comunicar com um sistema de base de dados.

Software

De um modo geral, o sistema será composto por módulos principais imutáveis, e um modulo configurável onde se poderá colocar o modelo de instalação para ser carregado pelo simulador:

-Módulo de comunicação;

-Módulo de gestão de dados;

-Módulo de gestão de processo real;

-Módulo de sistema de simulação;

-Módulo modelo de instalação física;

-Módulo de configuração.

Frisa-se que quando o sistema está a funcionar com um utilizador a executar controlo remoto estamos a utilizar um modo de funcionamento servidor-cliente. Neste modo de funcionamento o nosso sistema é o servidor, e o utilizador será um cliente.

Deve ser considerado um módulo funcional de armazenamento de dados a correr em paralelo. Este sistema tem acesso dinâmico e assim permite que mais plataformas e mais linguagens possam aceder à base de dados. A base de dados a utilizar deverá ter a capacidade de armazenar uma quantidade de dados a longo prazo. Estas características combinadas possibilitam fazer estudos dos dados a longo prazo: como por exemplo fazer estudos comportamentais de vários modelos de instalações.

Com as características expostas anteriormente é possível com as devidas credenciais de acesso; facilmente um telemóvel ou outro dispositivo que consiga executar linguagens de programação com acesso à *internet* poderá aceder e utilizar o sistema de simulação como utilizador.

A aplicação de cliente pode ser desenvolvida em qualquer linguagem/ambiente de programação, apenas é necessário considerar que para a comunicação com a plataforma é necessário que o protocolo de comunicação seja cumprido.

As características descritas fazem com que em qualquer outro ponto do planeta se consiga ter acesso à utilização deste sistema. Tais atributos possibilitam ao utilizador com uma unidade de cliente e uma outra de sistema de simulação

(servidor) consiga operar, independentemente do seu sistema operativo. Isto cria um modelo de fácil instalação de acesso e utilização.

A Figura 15, esquematiza a arquitectura geral, na qual se identifica as diferentes componentes do simulador.

A plataforma de simulação, implementado no microcomputador, a seleccionar, corre no seu próprio ritmo de amostragem. A execução de testes sobre o sistema de simulação, é realizada através do fecho do anel entre a instalação-simulador, processo real e o cliente.

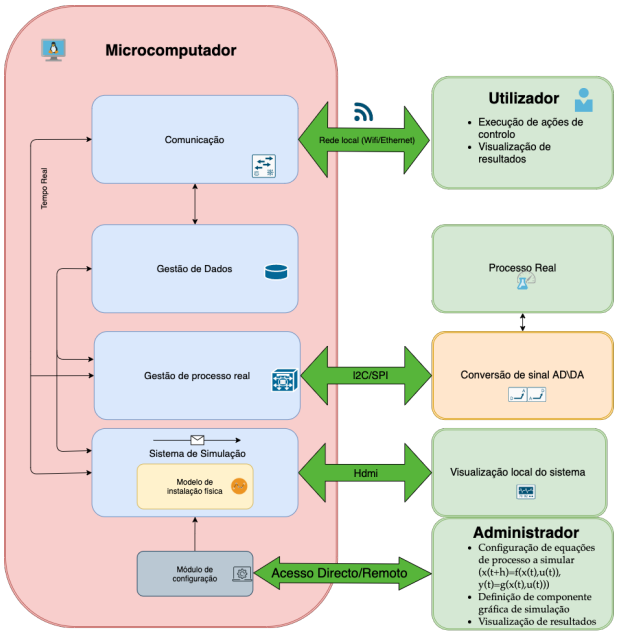


Figura 15 - Arquitectura geral do Sistema

Consegue-se definir os dois componentes principais integrados no microcomputador a seleccionar, o sistema de simulação e controlo por *hardware* e o sistema de base dados. Em termos de papeis: verifica-se que o utilizador terá que aceder ao sistema para executar as simulações. O administrador deve ter a

possibilidade de executar a configuração do simulador, para conseguir simular instalações representadas por modelos de estado.

$$\frac{d}{dt}x(t) = f\big(x(t), u(t)\big) - equação\ de\ estado$$

$$y(t) = g\big(x(t), u(t)\big) - equação\ de\ saida$$

3.2-Plataforma de simulação

Para alcançar este objectivo foi considerado um módulo principal de *Python* (plataforma de simulação) onde correm os processos de comunicação, gestão de dados, gestão de processo real e sistema de simulação.

O módulo principal, para além de ter vários processos a decorrer em paralelo necessita capacidade de troca de informação entre os mesmos, para isso é necessário elaborar um sistema de mensagens neste módulo.

Este sistema deverá ser fiável e permitir a verificação de mensagens em tempo real por cada um dos módulos, para garantir que o simulador se mantenha com o menor atraso de troca de mensagens. Este atraso propagar-se-á para o simulador, provocando a acumulação de mensagens, podendo gerar falhas em certas amostras de simulação, por isso, deve-se ter em conta a importância deste requisito. O formato de mensagens deverá ser transversal a todos os módulos, gerando assim uma capacidade de leitura das mensagens pelos mesmos. As mensagens podem ser lidas desde o ponto de entrada pelo servidor, até ao ponto de processamento por parte do simulador, devolvendo este as mensagens no formato implementado. Estas características facilitam a comunicação entre os clientes e o sistema

Nos próximos subcapítulos serão abordados os seguintes subprocessos:

3.2.1- Módulo de comunicação

O módulo de comunicação vai ser o agente responsável por dar a capacidade de acesso ao mundo exterior e ao cliente, através do sistema de rede do hardware seleccionado. Este módulo, vai ser capaz de interagir com uma rede *TCP/IP*, enviar e receber mensagens de clientes e transmiti-las para os respectivos módulos do sistema.

Estas mensagens, consoante a operação pedida ao servidor, são distribuídas pelo simulador ou pelo processo real. Uma vez que o sistema terá uma porta

para o exterior através deste módulo, este requer um sistema de autenticação, de modo, a que se consiga garantir a segurança do mesmo.

É importante garantir que as credenciais de acesso se encontrem armazenadas de modo persistente de forma que se o sistema necessitar de manutenção, ou reiniciar se mantenham as permissões e acessos por parte dos utilizadores ao sistema. Isto também garante um registo de todas interacções com os clientes.

O sistema de comunicações deve admitir o acesso por uma rede *Internet* utilizando o protocolo mais comum gerando assim o fácil acesso aos clientes. Este protocolo será IPV4(Protocolo de *Internet* Versão 4). Este protocolo consegue aumentar a versatilidade do sistema uma vez que as generalidades das linguagens de programação conseguem comunicar com este protocolo. Assim diversas plataformas que não consigam correr alguma linguagem de programação definida para este projecto, conseguirão comunicar á mesma com o sistema, utilizando outras linguagens desde que seja mantido o formato da mensagem e os parâmetros de ligação.

3.2.2- Sistema de simulação

O simulador é a componente responsável por garantir que existe uma instalação que executa o modelo, com um ritmo de amostragem pré-definido. Tem de conseguir armazenar dados de funcionamento numa "janela temporal" curta e ainda a monitorização do simulador. Também tem de ter a capacidade de ser autónomo do modelo que está a simular. Ou seja, deve executar diferentes modelos matemáticos de instalações físicas diferentes. Para garantir esta versatilidade o simulador é desenvolvido de modo a ter a instalação a carregar encapsulada num ficheiro editável pelo administrador do sistema.

O simulador tem de estar habilitado a simular modelos de simulação que sejam definidos por equações diferenciais, uma vez que este será o método mais comum para executar nos modelos de instalações. Para isso, em caso ideal todo o processamento de sinais ocorre apenas na execução do modelo de simulação; e os tempos de execução são geridos pelo simulador e não pelo modelo. Assim o simulador é configurável para poder ser definido o tempo de execução e amostragem à "priori", e a partir daí apenas interagir para despachar mensagens entre os diferentes processos do sistema e com o mesmo formato de mensagens que um cliente utiliza para comunicar com o sistema. Conjuntamente tem de se desenvolver um método para carregar as configurações, este método deverá ser comum a todo o sistema de modo que seja fácil para um administrador definir os parâmetros de execução do sistema.

O simulador é ainda responsável por garantir o funcionamento de todo o sistema, em completa autonomia do cliente (ligado ou desligado neste). Para isso, existe a necessidade de elaborar uma ferramenta que possibilite a configuração do simulador de modo a definir intervalos de amostragem, definições de armazenamento e algumas definições genéricas que se queiram adicionar.

3.2.2.1- Modelo de instalação física

Para que se consiga desenvolver um modelo de simulação a que corresponde um modelo físico de uma instalação: será necessário o uso de equações diferenciais. Na Figura 16, é representado um exemplo de um modelo clássico, o sistema de massa, mola e amortecedor.

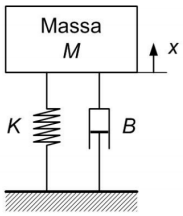


Figura 16 Modelo massa, mola, amortecedor[52].

As forças definidas pela mola e amortecedor são dadas por:

$$\begin{cases} F_m = -K(x(t) - x_0) \\ F_a = -B\frac{dx}{dt} \\ F_g = Mg \end{cases} \quad (1)$$

Onde K é a constante de elasticidade da mola em $[Nm^{-1}]$ e B é a constante de amortecimento em $[Nms^{-1}]$ e x a posição da mola relativamente ao eixo. Utilizando a lei de Newton $F = Ma$ e que:

$$\begin{cases} F_t(t) = F_m(t) + F_a(t) - F_g(t) \\ F_t(t) = M\frac{d^2x(t)}{dt^2} \end{cases} \quad (2)$$

Considerando que a posição inicial é de repouso $Kx_0 = Mg$ e as condições de (1) e (2) obtém-se:

$$M\frac{d^2x(t)}{dt^2} = -Kx(t) + Mg - B\frac{dx}{dt} - Mg \quad (3)$$

Por meio de manipulação da equação (3) chegamos à conclusão de que o sistema pode ser representado numericamente por:

$$\frac{d^2x(t)}{dt^2} + \frac{B}{M}\frac{dx}{dt} + \frac{K}{M}x(t) = 0 \quad \text{(4)}$$

Por fim a equação (4) pode ser convertido num modelo de 1ª ordem:

$$\begin{matrix} \frac{dx_1(t)}{dt} = x_2(t) \\ \frac{dx_2(t)}{dt} = -\frac{B}{M}x_2(t) - \frac{K}{M}x_1(t) \end{matrix} \quad \text{(5)}$$

Na equação (5) vê-se o modelo matemático que representa o sistema: massa, mola e amortecedor. Traduz-se num conjunto de equações diferenciais de primeira ordem. Aqui fica representado um tipo de modelo matemático que é necessário ser possível desenvolver e executar pelo simulador do sistema.

3.2.3- Módulo de gestão de processo real

Quanto à gestão do processo real, é desejável que seja executado através de controlo remoto e localmente. O *Raspberry Pi* permite o acesso a portas físicas através do seu acesso *GPIO* (*General Purpose Input/Output*). Os *GPIO's* recebem e enviam sinais digitais, e têm duas portas especificas, *SDA* (*Serial Data*) e *SCL* (*Serial Clock*) que lhe garante o acesso por comunicação I2C, SPI e UART.

O Raspberry Pi não permite escrita nem leitura de sinais analógicos. Para desenvolver este protótipo é prossuposto a utilização de conversores analógico-digitais; assim permite ler os sensores e actuadores de um sistema real e traduzir para sinal digital, ainda possibilita analisar os dados e interagir com o utilizador.

Para que seja possível realizar esta componente, como mencionado atrás, uma vez que existe a obrigatoriedade de adicionar conversores AD e DA e a maioria dos microcomputadores de baixo custo não incluem conversores DA é necessário utilizar portas físicas no microcomputador. Estas possuem a capacidade de utilização dos mesmos protocolos de comunicação que os conversores

O protocolo de comunicação SPI apesar de necessitar de mais portos físicos, como têm quatro linhas de comunicação das quais duas são de linhas de dados de entrada MOSI ("Saída Mestre", "Entrada Escravo") e MISO ("Entrada Mestre", "Saída Escravo"), permite comunicação em tempo real em ambas as direcções.

Nenhuma destas soluções consente a comunicação em tempo real com todos os dispositivos ao mesmo tempo, no entanto o SPI é o mais rápido, pois consegue a comunicação bidireccional gerando respostas mais rápidas[73].

A utilização dos protocolos descritos anteriormente (Capítulo 2.4.1) também darão escalabilidade ao sistema, uma vez que com pequenas alterações facilmente são adicionados outros dispositivos de conversão de sinal, ou sensores e actuadores. Assim rapidamente se adapta a diferentes modelos físicos de processos reais.

Considerando os meios de comunicação anteriormente referidos habilita-se o sistema de comunicação com circuitos inter-integrados. Estes circuitos inter-integrados são dispositivos de diversas naturezas como *microchips* que realizam a aquisição de sinais de sensores analógicos, sensores que respondem directamente quando invocados pelos seus endereços físicos.

A utilização destes *microchips* mais aconselhada são os conversores MCP 3008, referidos anteriormente, que possibilitam o acesso a entradas físicas por parte do sistema. Este componente possibilita a ligação à maioria dos sensores e actuadores laboratoriais disponíveis na universidade, uma vez que ambos operam em tensões de 3.3V e 5V.

Para coordenar entre o microcomputador e dispositivos referidos anteriormente deve ser criado um processo que esteja a monitorizar o processo real. Nesse processo também deve ser permitido realizar testes. Assim um utilizador com o devido acesso poderá executar acções de controlo remoto no sistema real em que se encontra.

A Figura 17 apresenta um modelo de ligações para ambos os modos de aquisição de dados.

Analisando esta Figura verifica-se que alguns sensores passam a informação, convertida, pelos protocolos I2C ou SPI sendo este também o mesmo meio de comunicação que o MCP 3008 utiliza. Este circuito integrado habilita a plataforma para aquisição de tensões de até oito canais de captação de tensão, havendo sempre a necessidade de configurá-lo para dispositivos de 3.3V ou 5V.

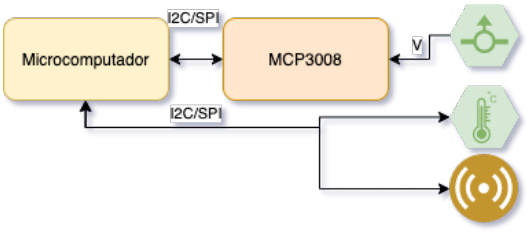


Figura 17 - Modelo de ligação de microcomputador com portos externos.

3.2.4- Módulo de gestão de dados

Na componente de armazenamento de dados existem duas vertentes a considerar:

-Curto prazo;

-Longo prazo.

O armazenamento a curto prazo permitirá aos intervenientes no sistema conseguirem saber os estados em tempo real. Este não necessita de uma "janela" temporal grande. Quando são conhecidos os estados em tempo real do sistema, consegue-se passar a informação para outras máquinas e também se gera a possibilidade de criar interfaces gráficos com informação em tempo real.

O armazenamento de dados a longo prazo concebe a possibilidade de armazenar as simulações. Estes dados já têm necessidade de uma maior "janela" temporal, e diferentes simulações de várias instalações industriais. Gera também a robustez dos dados, uma vez que o sistema, mesmo que o microcomputador seja reiniciado mantém, sempre a integridade dos dados.

Para este armazenamento deve ser considerado uma base de dados relacional, uma vez que é muito importante os dados estarem ordenados por categorias. Também se valoriza o facto deste tipo de base de dados deixar adicionar tabelas (onde se pode considerar os dados de outros simuladores) e relacioná-los com o simulador em execução. O seu acesso por máquinas externas é possível e fácil, só necessita que se coloquem outros microcomputadores a comunicar com a base de dados.

3.2.5- Módulo de configuração

O módulo de configuração é o local onde será possível configurar os parâmetros de funcionamento geral do sistema. Estes parâmetros habilitarão o administrador a configurar os restantes módulos do sistema. Deve ser de fácil acesso e compreensão. Para isso deverá ser considerado um ficheiro, que permita a utilização de comentários para qualquer administrador conseguir compreender as variáveis de configuração de modo intuitivo.

Deve conter parâmetros de configuração como o endereço e a porta TCP/IP, por exemplo, do módulo de comunicação, tempo de amostragem para o sistema de simulação, ou os valores iniciais para o modelo da instalação física.

4

Implementação do Sistema

Neste capítulo é abordada a implementação do sistema, assim como o seu funcionamento.

4.1- Hardware

4.1.1- Unidade de simulação

O *Raspberry Pi 3B* foi considerado o microcomputador ideal para este trabalho, pois possui as seguintes características:

- Possui placa única;
- Dotado de capacidades de computação necessárias;
- Acesso à rede;
- Acesso a portas físicas;
- Apresenta o melhor desempenho face ao seu custo e um baixo consumo energético;
- Considera-se também que possui uma elevada capacidade de replicação, uma vez que basta copiar o cartão de memória e introduzir esta cópia num segundo *Raspberry Pi* para ter uma réplica do sistema;
- O *Raspberry Pi* permite a actualização do microcomputador, já que ao replicar o cartão de memória e introduzi-lo noutro modelo de *Raspberry Pi* mais recente, e com melhores capacidades de hardware o sistema irá funcionar apresentando um melhor desempenho.

Tendo em conta a existência de *Raspberry Pi's* mais recentes e com melhores capacidades, ao seleccionar o *Raspberry Pi 3B* consegue-se garantir que se o

sistema estiver funcional, e com um desempenho aceitável, o desempenho irá ser superior nos modelos mais recentes. É de salientar que para utilizarmos o *Raspberry Pi* é necessário um cartão de memória que tenha pelo menos 64GB para operar o sistema correctamente.

4.1.2- Unidade de Utilizador

O cliente apenas necessita de ter um equipamento com acesso à mesma rede local que a unidade de simulação, e capacidade de comunicar utilizando o protocolo definido. A linguagem e ambiente de desenvolvimento não são limitação, independentemente do sistema operativo que este estiver a utilizar.

Existe uma grande diversidade de dispositivos que têm capacidade para correr linguagem *Python:* como *smartphones*, computadores, *Raspberry Pi*'s, entre outros. Optou-se neste trabalho, para se poder testar a unidade de simulação, pela linguagem *Python* para desenvolver a unidade cliente (utilizador) de testes.

Neste caso serão utilizados um computador com sistema operativo *Windows* e um telemóvel com sistema operativo *Android* na realização de testes para demonstrar a versatilidade do sistema.

4.2- Software

Na implementação de software foram escolhidas a linguagem *Python*, e a base de dados PostgreSQL, como principais infra-estruturas. Na Figura 18 consta um esquema geral onde se elucida o que foi implementado. Em *Python* foi implementada a classe principal (*Main*) que interage e controla as classes simulador e servidor.

A classe simulador controla os tempos de execução se a instalação estiver a correr em modo automático, ou acoplada a um cliente. Este também comunica com o modelo de simulação em execução. O servidor executa a gestão e interacção de clientes externos, certificando-se da autenticação com auxílio do gestor de dados.

O modelo de simulação representa a instalação industrial que se pretende simular (Figura 18).

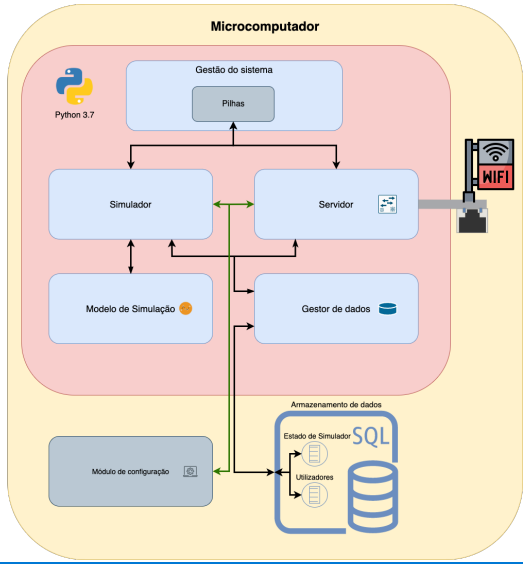


Figura 18 - Esquema de implementação dos módulos de software no microcomputador *Raspberry Pi*.

A selecção da linguagem *Python* relacionou-se com o facto de esta apresentar um bom desempenho: ser versátil, deter uma licença de código aberta, gratuita e simples de implementar. De notar que a escolha desta linguagem também se relaciona com o facto da mesma estar referida em vários artigos e sites *online*, o que facilitou o desenvolvimento do protótipo elaborado nesta tese [74].

Por fim, este protótipo será integralmente independente de licenças pagas, o que valorizará este trabalho.

De modo a criar as funcionalidades propostas, foram desenvolvidos os seguintes módulos:

4.2.1-Sistema de armazenamento:

O sistema de base de dados selecionado foi o sistema *PostgreSQL*.

Este sistema vem com a ferramenta *pgadmin*, que consiste numa ferramenta de administração do serviço de base de dados. Esta ferramenta pode ser acedida por uma página no endereço local do servidor de base de dados.

Esta ferramenta permite:

- Fazer as operações de administração de base de dados;
- Controlar os acessos à mesma;
- Estudar o espaço que está a ser ocupado por esta;
- Aceder remotamente ao sistema de dados com autenticação.

Assim quando se pretende, por exemplo adicionar ou remover utilizadores, é conseguido de forma segura. A ferramenta *pgadmin* é acessível por páginas de *HTML(Hypertext Markup Language)*. Também contem conector JDBC que, possibilita a criação de interfaces gráficos, assim como o próprio simulador pode armazenar os dados de operação, mantendo o histórico, e enviá-los para um cliente.

O *PostgreSQL* tem um elevado grau de versatilidade; não apenas pelas suas capacidades como base de dados, mas também por dar possibilidades de acesso ao servidor a qualquer aplicação que tenha ligação ODBC ou JDBC.

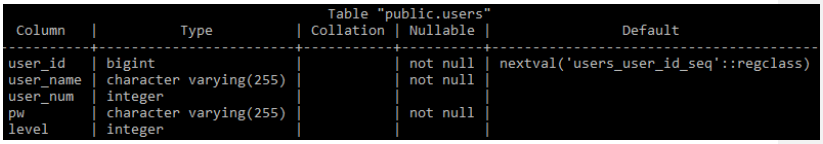
Concretamente, algumas linguagens possíveis de garantir o acesso são: o *Python*, *PHP*, e *Java*, permitindo assim criar diferentes plataformas, por exemplo de PHP para criar páginas de *HTML* e aceder a base de dados pela *internet* e através de *Java*. A programação por *software* permite retirar e inserir dados na base de dados se necessário.

Em *Python* podem-se criar: páginas de acesso na rede, *software* local, assim como realizar as operações necessárias que envolvam o simulador.

Inicialmente por acesso directo apenas se pretende que o simulador tenha permissão para aceder em modo de leitura á base de dados. O acesso directo poderá ser testado e manipulado no *Raspberry Pi* por acesso por terminal, ou através de linguagem *Python*, ou poderá ainda ser acedido por meios de *internet* actuais com as devidas credenciais de acesso.

Para este projecto foram desenvolvidas duas tabelas: a tabela Utilizadores que consta na Figura 19, contem os dados dos utilizadores e os seus níveis de privilégio no sistema; a tabela Simulações armazena os dados de simulação e regista as alterações de entradas assim como os estados do sistema.

Formatted: Not Highlight



```
                                Table "public.users"
 Column   |          Type          | Collation | Nullable |               Default
----------+------------------------+-----------+----------+-------------------------------------
user_id   | bigint                 |           | not null | nextval('users_user_id_seq'::regclass)
user_name | character varying(255) |           | not null |
user_num  | integer                |           |          |
pw        | character varying(255) |           | not null |
level     | integer                |           |          |
```

Figura 19 - Tabela que indica os atributos e os tipos de dados em cada atributo.

Na tabela Utilizadores (Figura 19) encontramos os seguintes campos:

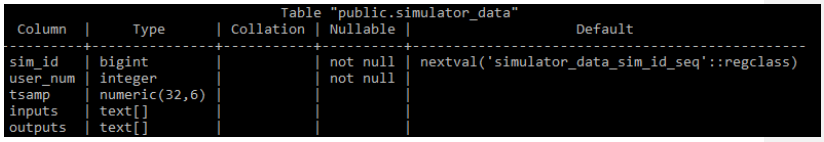
-user_name: Nome (pretende-se armazenar o nome do utilizador);

-user_num: Número (de modo a identificar um utilizador);

-pw: Palavra-passe (garante possibilidade de autenticação);

-level: Nível de privilégios (Nível: 1-Administrador, 0-Utilizador).

De modo a se conseguir armazenar os dados foi criada a tabela Simulator_data com os tipos de dados que constam na Figura 20.



```
                                Table "public.simulator_data"
 Column  |     Type      | Collation | Nullable |                   Default
---------+---------------+-----------+----------+----------------------------------------------
sim_id   | bigint        |           | not null | nextval('simulator_data_sim_id_seq'::regclass)
user_num | integer       |           | not null |
tsamp    | numeric(32,6) |           |          |
inputs   | text[]        |           |          |
outputs  | text[]        |           |          |
```

Figura 20 - Tabela que contem os atributos e os tipos de dados dos atributos dos dados de simulaçao armazenados.

Na tabela de Simulações(*simulator_data*) (Figura 20) encontramos os campos necessários para armazenar a informação das simulações a longo prazo:

-Sim_id: índice da tabela *simulator_data*, do tipo inteiro. Este é automaticamente gerado, com uma ferramenta de sequências da base de dados;

- user_num: Número de utilizador (permite identificar quem solicitou a alteração de valores de entrada). É do tipo inteiro considerando os exemplos de números de alunos ou utilizadores como dados ordinais contínuos;

-tsamp: tempo de amostra (medido em segundos a partir de 1 Janeiro de 1970), estes serão variáveis do tipo ordinais continuas do tempo, discretizadas pelo simulador com seis casas decimais no máximo;

- Inputs: armazena os valores de entradas u(tsamp) da instalação, do tipo de texto, o que em *PostgreSQL* permite armazenar vectores;

-Outputs: armazena os valores das saídas y(tsamp), da instalação, também do tipo texto, armazenando assim vectores de saída.

O sistema de base de dados tanto pode estar a verificar utilizadores em coordenação com o servidor, assim como o simulador poderá estar a ler e escrever na sua tabela de dados. Se quisermos ainda adicionar funções é permitida conectividade por outros meios que não *Python*, admite assim escalabilidade para no futuro serem adicionadas novas funcionalidades ao sistema, como o acesso por outras instalações replicadas, possibilitando a recolha a longo prazo dos vários simuladores por uma base de dados.

4.2.2-Módulo de configuração do sistema

Para realizar as configurações do sistema foi criado um ficheiro com formato ".ini" que possibilita a sua utilização transversalmente no sistema. Esta estratégia foi adoptada para poder ter um ficheiro de configuração de fácil acesso e alteração. Também tem uma elevada legibilidade, uma vez, que permite a introdução de comentários. Na Figura 21 pode-se ver as linhas que são iniciadas com cardinal são linhas de comentários. As linhas que têm itens com parêntesis rectos são secções, onde se encontram linhas sem nenhum caracter especial, são as variáveis de configuração de configuração.

```
# SETUP FILE
[server]
# SERVER PORT SETUP
serverport=5046

[sim]
# SIMULATOR SAMPLING TIME
ts = 0.25
# INITIAL SYSTEM INPUT VALUES
uinit = 0.0, 0.0, 0.0, 0.0
# INITIAL SYSTEM STATE VALUES
yinitial = 0.1, 0.0
# STORAGE INTERVAL BETWEEN SIMULATION ITERATIONS
storageinterval = 10
[storage]
# INSTANT STORAGE SIZE
inst_stor_size = 3
```

Figura 21 - Exemplo de ficheiro de configuração.

No exemplo da Figura 21 podemos identificar no ficheiro que existem duas secções, do servidor e do simulador. Nas configurações do servidor([server]) apenas temos como variável a porta(serverport) que se quer definir para o servidor. Considerando a secção do simulador([sim]) encontramos as variáveis associadas ao funcionamento do simulador.

Assim tem-se:

-Intervalo de amostragem definido (Ts) com 0.25*s*;

- Valores iniciais de entradas(*uinit*) do modelo a simular, neste caso quatro entradas com valores iniciais nulos;

-Valores iniciais das variáveis de estado do modelo(*yinit*);

- Intervalo de armazenamento de valores de simulação (*storage interval*), este define o intervalo de quantas em quantas interacções de simulação ocorridas se armazenam os dados;

- Número de amostras a armazenar na memória instantânea (inst_stor_size).

É de notar que quando se definem as variáveis iniciais *unit* e *yinit*, estas são vectores. Quando definimos estas variáveis iniciais estamos a definir a configuração de mensagem, dos dados, e também a dimensão do sistema, ou seja, o número de entradas, número de variáveis de estado e número de saídas. Estas configurações têm de ser coerentes com o modelo a simular.

4.2.3-Unidade de gestão do sistema

A unidade de gestão do sistema é onde se encontra a gestão de todos os módulos da plataforma. Aqui geram-se os principais subprocessos do sistema e o módulo de pilhas.

O sistema de pilhas realiza a troca de informação entre os vários subprocessos do sistema. Este sistema também aloja o sistema de inicialização do sistema de base de dados *PostgreSQL*, garantindo assim uma verificação do estado geral da base de dados aquando da inicialização do simulador. Os subprocessos entram em funcionamento ao invocar uma classe do tipo servidor e outra do tipo simulador, e integra as classes em tarefas(*threads*) a correr em concorrência. Estas tarefas estão a funcionar enquanto o sistema principal se encontra activo.

4.2.4-Pilhas

Com implementação do serviço multitarefa, e tratando se de um sistema a correr em tempo real, é necessário colocar as tarefas a comunicar entre si.

De modo a poder passar a informação entre os vários subprocessos foi utilizado um sistema de filas de esperas. Estas são pilhas onde podemos armazenar objectos para transmitir informação entre tarefas a correr. Neste caso as filas irão

comunicar entre os serviços do sistema, enviar os dados de simulação, tempo de amostragem e os valores dos parâmetros de entrada e de saída do simulador.

As pilhas, sendo elas, neste caso, do tipo primeiro a entrar primeiro a sair (*FiFo-First In First Out*), necessitam especial cuidado com a sua utilização, uma vez que podem gerar erros de memória. Para isso as pilhas foram utilizadas apenas para quando o simulador responder para o exterior da plataforma. Como formato de mensagem a colocar dentro das pilhas foram concebidos dois dicionários, simplificando assim todo o processo de fecho do sistema entre cliente que envia mensagem para o servidor através da rede, e por fim na plataforma de simulação servidor que partilha a mensagem para o simulador através da pilha.

Em análise, o caso de utilização do dicionário garante uma uniformização de dados ponto a ponto, entre o controlador e o modelo de simulação.

A Figura 22 é uma representação das pilhas implementadas. A implementação das pilhas inicia-se no programa principal recorrendo a uma biblioteca de *Queues* de *Python* que gera duas pilhas diferentes para comunicar nos dois sentidos:

- Servidor para simulador;

- Simulador para servidor.

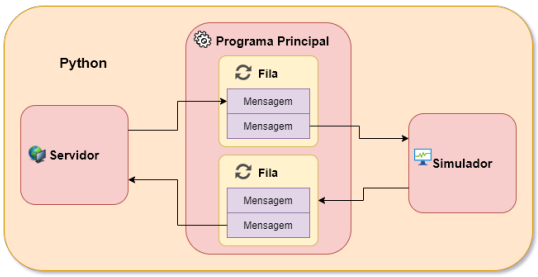


Figura 22 - Representação das Pilhas.

Quando o servidor quer comunicar com o simulador, se autorizado, coloca a mensagem na pilha através do método "put("mensagem")", o simulador retira a mensagem através do método "get("mensagem")", disponibilizando assim os dados. O mesmo acontece no sentido oposto, simulador para o servidor.

Por vezes existem falhas, que geram alguns pacotes que se perdem pela rede. Tal fenómeno provoca acumulação de dados na pilha. Neste contexto

também foi criada uma solução que limpa a pilha periodicamente garantindo assim que não existe um atraso demasiado elevado nas informações de retorno para o cliente. Pois a reacção em cadeia de um elevado número de amostras acumuladas, irá provocar um atraso nos resultados das simulações.

Ao utilizarmos as filas de espera é importante estabelecer um intervalo de espera entre as leituras das mesmas. Este intervalo é o que permite que diferentes processos consigam entrar nas pilhas para poder colocar e retirar dados destas. Caso contrário criam-se conflitos através diferentes processos a correr (não podem estar a aceder a pilha dois processos em simultâneo).

Como existe certo intervalo de tempo entre as leituras, este processo provoca atrasos na simulação. Se os tempos de espera forem estáticos criam entraves a resposta do sistema, limitam a velocidade com que se pode colocar o simulador a correr, consequentemente restringe todas as potencialidades do sistema. Assim de modo a colmatar esta limitação, o tempo de espera de acesso à pilha foi definido para estar dependente do tempo de amostragem do simulador. Colocando a pilha a trabalhar coordenada com o simulador e evitando atrasos no sistema, esta adaptação concedeu um melhor desempenho do sistema.

4.2.5-Servidor

Foi criado uma classe servidor que permite ligações assíncronas podendo assim enviar e receber mensagens entre o servidor e o cliente.

Para fazer isso teremos que ter um endereço de protocolo de internet e uma porta de acesso ao sistema (por exemplo o endereço 192.168.1.1, porta:5046)

De modo que as operações sejam seguras foi implementado um sistema de autenticação com a ajuda da base de dados. Para um cliente ter permissão quando envia uma mensagem de operação, tem também que enviar um utilizador e uma palavra-passe. O servidor compara dados recebidos com os existentes na base de dados e se forem concordantes serão permitidas as operações. Na Figura 23 está representada a sequência de operações de interacção com o sistema de servidor.

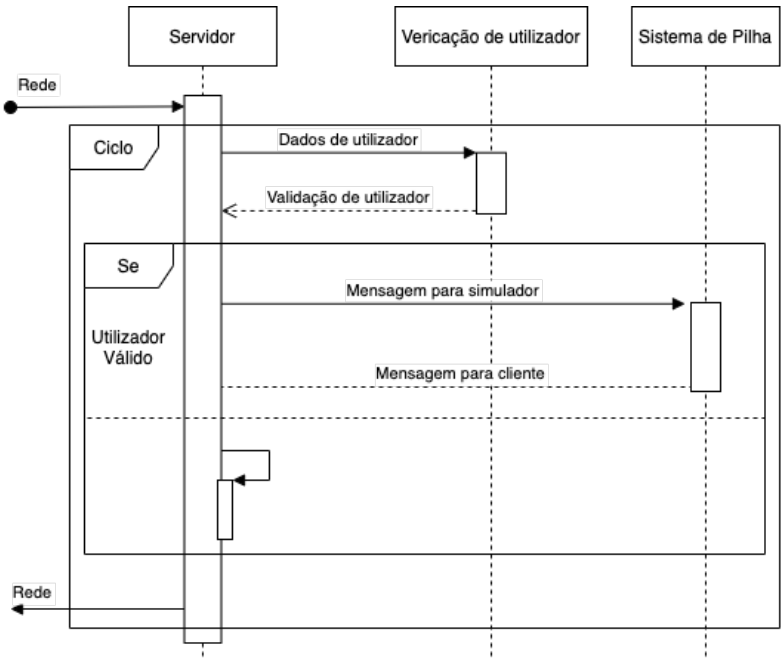


Figura 23 Diagrama de sequência de operação do módulo servidor

O servidor tem um *buffer* de envio de mensagens que muda de dimensões consoante o tamanho da mensagem. Esta versatilidade no buffer permite fazer um envio de uma mensagem completa de cada vez, libertando recursos de processamento que seriam necessários caso o tamanho do buffer fosse estático.

Este servidor recorre a algumas bibliotecas (socket, *time*, *JSON*, *data_ops*) de *Python* de modo a poder executar as funções necessárias.

A biblioteca *socket* permite vários tipos de ligações em domínios de rede diferentes como a rede interna da máquina, a rede local e a *internet*. Esta biblioteca funciona como um objecto do tipo ligação a que associamos o tipo de ligação pretendida, neste caso acesso a *internet*. Define-se o endereço de protocolo de *internet* da máquina, com o mesmo de acesso à rede e colocamos o servidor a aguardar ligações, ficando à espera que seja iniciada uma ligação.

Para realizar a transmissão de mensagens utilizamos o formato *JSON*, que é um formato leve e simples de ler para troca de dados, tanto entre programas, como linguagens ou máquinas diferentes, são estruturas universais e

normalmente organizadas por vectores [75]. Por exemplo para comunicarmos com páginas de *internet*, outros clientes ou até *Labview* ou *Matlab*. Assim considera-se este formato extremamente versátil. Mais se acrescenta que este formato é considerado dos melhores formatos de comunicação disponíveis actualmente e que a *internet* de hoje em dia utiliza em primazia [76].

Um cliente estabelece ligação com o servidor. Após a ligação estar estabelecida e receber uma mensagem, abre-se o seu conteúdo. O conteúdo da mensagem recebida tem que ser em formato *JSON*. São extraídos os itens utilizador e palavra-passe. Na Figura 24, observa-se um exemplo, de uma mensagem recebida através do servidor. Relativamente ao utilizador pode-se distinguir:

-user - Nome de utilizador;

-usernum – Número de utilizador;

-pw – palavra-passe.

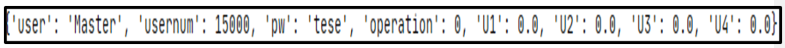


Figura 24 - Mensagem recebida onde se podem verificar o utilizador e password.

Consequentemente o servidor invoca o método *checkuser*, da classe data_ops, que permite verificar na base de dados se o utilizador existe e é válido, existindo é verificada a palavra-passe. Este processo permite criar um certo tipo de segurança e também armazenar registos de ordens associadas a um dado utilizador.

Sendo esta verificação bem-sucedida passa-se o cliente para a parte de gestão do cliente, à classe handle_client, por onde passam a ser feitas todas as comunicações entre servidor e cliente, a ocorrer enquanto o cliente se encontrar ligado. Cada vez que entra uma mensagem, esta é transmitida como argumento a saída da *thread* que corre a classe servidor, e para o gestor do sistema através da pilha. Esta mensagem é dada como argumento à *thread* que corre a classe simulador.

No servidor existem também requisitos de segurança, que controlam a ligação em si, que faz com que se um cliente por algum motivo perder a ligação ao servidor, o servidor fecha a ligação, eliminando assim erros de ligação fechada de forma abrupta e também permite a ligação de novos clientes à mesma porta, se assim for necessário.

Através do método enunciado anteriormente, conseguimos garantir a estabilidade do servidor, nas primeiras tentativas ocorriam erros que terminavam a execução, por ligações abruptas. Esta segurança é garantida através do sistema

de excepções (*try-catch*) de *Python* que quando apanha o erro de ligação, reinicia o processo de espera de ligação, permitindo que um cliente se tente conectar novamente ao sistema.

4.2.6- Simulador

Para implementar o simulador (Figura 25) foi implementada uma classe de *Python* com o mesmo nome, que inicia a correr em *loop* quando sistema é iniciado.

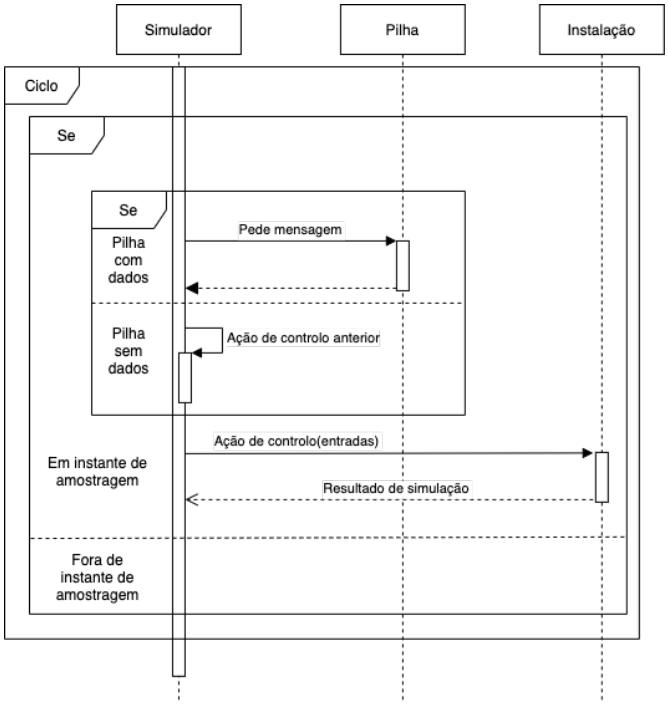


Figura 25 - Diagrama de sequência do simulador.

Este simulador (Figura 25), será responsável por controlar os tempos de execução de cada iteração de modelo de instalação, por interagir com a classe de gestão de dados, para armazenar os dados nos intervalos de amostragem de armazenamento pretendidos. Assim a classe implementada começa por carregar os dados a partir do ficheiro de configuração do sistema, de modo a ter valores

iniciais para realizar as simulações. Depois carrega o modelo da instalação a simular implementado e inicia o mesmo. Após a inicialização entra em ciclo infinito até ao momento em que se desliga o sistema de simulação.

Enquanto o modelo de simulação se encontrar no ciclo, verifica se existem mensagens na pilha em espera de clientes remotos, se existir passa a mensagem para o modelo de simulação, se não recorre aos valores de entrada da iteração anterior e envia-a para o modelo de simulação. Isto faz com que o sistema não seja dependente de utilizadores-clientes para funcionar, o que lhe dá autonomia para funcionamento independente. Ao receber mensagens da pilha, quando retorna resultados de simulação dessa iteração devolve-os á pilha para poderem ser retornados ao cliente. Isto faz com que o cliente possa utilizar estes dados no controlador implementado remotamente para poder gerar a próxima acção de controlo.

4.2.7- Gestão de Dados

Na parte de gestão de dados iremos encontrar toda a parte que contém a interacção do *Python* com o sistema de bases de dados *PostgreSQL*.

Na classe de gestão de dados foi criado um método de autenticação de utilizadores requerendo esta informação à BD (verificando nomes de utilizadores e respectivas palavra-passe). Primeiro testa-se, se o nome recebido consta na tabela da BD, se não constar rejeitamos a mensagem. Se constar, verificamos se a correspondente palavra-passe na BD é igual à da mensagem recebida, se sim deixamos a mensagem entrar no sistema, se não rejeitamos a mensagem.

A verificação da mensagem ocorre através da classe *data_ops*, esta permite operações de utilização da base de dados. Após a validação correcta dos dados de utilizador, a mensagem de operação é passada a *thread* simulador que corre a classe simulador. Este simulador define o sistema temporal e também o intervalo de amostragem, e é aqui que se recorre ao sistema físico. Este pode ser utilizado pelos administradores para definir um sistema físico, esta é a componente a alterar pelo utilizador.

A classe *data_ops* é onde se realiza a escrita na base de dados aquando da execução de simulações, fazendo com que se consigam armazenar dados de simulações de clientes.

O Simulador irá recorrer á classe *data_ops* para poder armazenar os dados de simulação que se considerem importantes. Assim foram definidos dois

critérios para armazenamento dos dados. Os critérios para o sistema realizar o armazenamento de dados na memória de longo prazo são:

- A iteração simulada for executada com valores de entrada enviados por um utilizador externo;
- A diferença entre os últimos valores armazenados e os actuais ultrapassarem um dado limiar. Este limiar é definido no ficheiro de configuração

4.2.8-Instalaçoes de teste

Modelo de simulação

Para testar o sistema foi criado um modelo de uma instalação de 2 tanques, com quatro valores de entrada para o controlar e dois valores de saída. Este modelo pode ter entradas analógicas, digitais, do tipo real ou binárias.

A instalações de teste são as que devem ser desenvolvidas por um administrador do sistema. Estas são a representação do sistema físico, em modelo de simulação.

Para podermos simular um sistema físico com modelos físico-matemáticos programáveis necessita-se de utilizar um conjunto de equações diferenciais.

As equações diferenciais possibilitam o cálculo dos estados - perante um conjunto de entradas, obtêm-se os valores das saídas. Por exemplo, utilizando as derivadas num modelo simples calcula-se a posição ao longo tempo à custa de uma posição inicial, uma velocidade e uma aceleração. Isto gera a capacidade de reescrever uma equação recorrendo às suas derivadas.

Método Solve Ivp

O método *solve_ivp* requer alguns parâmetros de entrada obrigatórios:

- Função iteradora que tem como argumentos o tempo e y, em que y pode ser vector ou um valor fixo. Define um sistema de equações diferenciais que se querem integrar;

- t_{span} intervalo de integração, em vectores no mínimo com duas amostras de tempo;

- y_0 valor inicial das variáveis de estado do sistema;

- Método de integração - existe a possibilidade de escolher entre diferentes métodos de integração numérica, tais como o *Runge-Kutta* de diferentes ordens;

- $args$ – argumentos de entrada e de estado para passar à função iteradora.

Existem outros argumentos de entrada facultativos que permitem configurar opções adicionais e entradas por eventos.

Como argumento de saída retorna um objecto estruturado, pelos seguintes sub-objectos:

- t - vector de tempo;

-y -vector de solução;

- *success* – variável de controlo que permite verificar se a integração foi bem-sucedida[77].

O estudo do modelo de dois tanques, encontra-se representado na Figura 26:

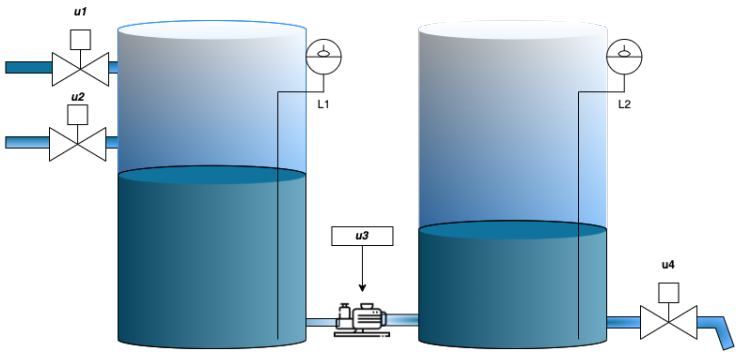


Figura 26 - Modelo de tanques para teste de funcionalidades.

A instalação de tanques foi projectada com três válvulas (u_1, u_2, u_4) e uma bomba (u_3), ou seja, quatro acções de comando e duas saídas que representam o volume dos tanques simulados (L_1, L_2).

Modelo de instalação:

O modelo de instalação (Figura 26) foi definido num método que recebe o instante de amostragem(t), volume inicial dos tanques(vol) e o estado inicial (*state*). Verifica-se também que neste modelo está definido o volume máximo do tanque ($0.5m^3$) e a área de secção dos tanques ($0.5m^2$).

Deleted: 5

```
def d_tq(t, vol, state=(0, 0, 0, 0))
    volumemax = 0.5  # [m^3]
    seccaotq = 0.5  # [m^2]
```

Figura 27 Detalhe do método que define o modelo de tanques.

Na Figura 27 observa-se que o caudal máximo da 1ª válvula é de $0.02m^3/s$, e o caudal máximo da segunda válvula é de $0.01m^3/s$. É de notar que o sistema está dimensionado para que as posições das válvulas só possam apresentar valores entre zero e um. A Figura 28 também descreve a fórmula para obter a altura dos tanques através do quociente do volume, proveniente da iteração anterior, e a área de secção do tanque.

```
q_in_1 = 0.02*valv1+0.01*valv2  # [m^3/s]
altura_1 = vol[0]/seccaotq
```

Figura 28 - Cálculo de caudal de entrada no 1º tanque.

$$q_{in_1}(t) = 0.02 * valv_1 + 0.01 * valv_2$$
$$q_{out_1}(t) = 0.02 * bomba$$
$$Q_{tanque_1}(t) = q_{in_1}(t) - q_{out_1}(t)$$
$$\frac{dV_{tanque_1}}{dt}(t) = q_{in_1}(t) - q_{out_1}(t)$$

A derivada é definida à custa de diferença de caudais como se observa na Figura 29.

```
deriv_tq_1 = q_in_1-q_out_1
```

Figura 29 Detalhe que demonstra como é calculada a derivada.

Simulador de iterações

Na Figura 30 consegue-se observar o método que habilita para a resolução da equação diferencial do sistema de tanques, em cada iteração. Esta função recebe três variáveis de entrada:

- t_{samp} – instante de amostragem;

- y_0 – valores iniciais para resolução de equação inicial;

- u_{array} – valores de posições de entradas do sistema.

Ainda na Figura 30 visualiza-se os passos para a resolução do sistema:

1- Resolução de equação diferencial através da função *solve_ivp*, este recorre ao modelo de instalação de tanques definido;
2- Extrair os valores dos últimos valores das variáveis de estado (y_{ss}), para armazenar e utilizar como valores iniciais novamente;
3- Retornar os valores(simresult) para o sistema de gestão do simulador, já no formato geral de mensagens do simulador, e transversal a todo o sistema. Esta utilização ajuda a transversalidade das mensagens entre todo o sistema, e com poucas alterações aos clientes que se liguem ao sistema.

```
def simulate(self, tsamp, y0, u_array):
    yss = np.array([])
    output1 = solve_ivp(lambda t, y: RtSystem.d_tq(t, y, u_array), (tsamp[-2], tsamp[-1]), y0)
    yss = output1.y[:, -1]
    model_outs = yss
    simresult = {"ts": tsamp[-1], "ins": u_array, "outs": model_outs}
    return simresult
```

Figura 30 - Método para resolução da equação diferencial.

Na função *Simulate* é possível fazer manipulações dos resultados das equações diferenciais. Em alguns casos as saídas ainda requerem manipulação para serem transformadas, ou conjugadas com outros modelos de instalações.

É ainda importante garantir que as saídas deste método se mantenham consistentes. Estas saídas são as já retornadas do modelo físico-matemático, onde o administrador pode actuar, para o sistema de gestão de simulação. Assim na Figura 31 detalham se as variáveis da mensagem *simresult*, que consiste num dicionário:

- ts: $t_{samp}[-1]$, instante de última amostragem, formato número real;
- ins: u_{array}, vector com os valores da última iteração de entradas. Formato de vector que pode ser de dimensão variável consoante o número de entradas;
- outs: $model_{outs}$ é um vector com as saídas do sistema. Este é calculado a partir dos valores de variáveis de estado (y_{ss}) e dos sinais de entrada (u_{array}), após resolução de equação diferencial. Este vector é de dimensão variável consoante o número de saídas que se pretendam devolver para o armazenamento, e cliente se este estiver ligado.

```
simresult = {"ts": tsamp[-1], "ins": u_array, "outs": model_outs}
```

Figura 31 - Saídas do método onde o administrador pode manipular o modelo.

5

Resultados

5.1- Teste de módulos funcionais de plataforma de simulação

De modo a serem realizados testes do sistema utilizou-se o modelo de instalação da Figura 26, a funcionar em anel aberto. Para garantir a robustez do sistema os testes foram realizados de forma compartimentada e modular. Assim é possível testar cada um dos módulos e garantir o seu bom funcionamento.

Os testes realizados foram com a plataforma implementada num *Raspberry Pi 3B*, ligado a uma rede sem fios. Foram utilizadas duas plataformas cliente diferentes, um computador portátil, e um *smartphone* com sistema operativo Android. Ambos os clientes testados se encontravam ligados a mesma rede local que a plataforma local.

Quando o sistema inicia executam-se o modulo servidor e simulador, e são carregados os parâmetros de configuração a partir do respectivo ficheiro. Na Figura 32 consegue-se verificar que os parâmetros são carregados, que ambos os sistemas se inicializam, o simulador fica a correr com o seu tempo de amostragem definido, e o servidor a aguardar ligações.

```
Simulator Configuration Loaded:
Sampling Time:
0.1
Storage Sampling Interval:
10.0
Inputs Startup Values:
[0. 0. 0. 0.]
Outputs Startup Values:
[0.1 0. ]
```

Figura 32 - Leitura dos parâmetros de configuração carregados pelo sistema).

A informação que consta na Figura 32 normalmente não é mostrada, foi apenas produzida para testes de validação.

O passo seguinte é verificar que no simulador é recebido o pedido de ligação do cliente e que a ligação é estabelecida. Em relação a este passo foram considerados alguns testes de resiliência para garantir que se conseguia manter o sistema ligado durante vários dias mantendo a sua funcionalidade. Com isto verificou-se que é possível mantê-lo a correr por um vasto período de tempo, com pouca ou nenhuma manutenção. Como se pode visualizar na Figura 33 conclui-se que o sistema e o cliente estabelecem a ligação com sucesso.

```
[STARTING] server is starting...
[LISTENING] Server is listening on 0.0.0.0
[NEW CONNECTION] ('192.168.1.9', 54658) connected.
```

Figura 33 - Verificação de ligação estabelecida pelo cliente de endereço IP 192.168.1.9 da rede local

No teste seguinte verifica-se que o simulador está a ser executado de acordo com o intervalo de amostragem pré-definido. Para isso foram verificadas as diferenças entre as instâncias de tempo que estão a ser iteradas no simulador. Na Figura 34 apresenta-se o resultado do cálculo do intervalo de amostragem por diferença de duas iterações consecutivas, depreende-se pela análise da variável de tempo do sistema que este se encontra a funcionar conforme o intervalo de amostragem pré-definido.

```
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
```

Figura 34 - Cálculo de tempo de amostragem por diferença de duas iterações consecutivas.

Para se prosseguir nos testes recorre-se ao sistema de cliente que é colocado a executar. Ao estabelecer uma ligação obtém-se um sistema de autenticação funcional que conceda segurança ao sistema, com recurso à base de dados onde estão alojados os utilizadores, dados de acessos e os seus níveis de privilégios. Para isso é necessário verificar que utilizadores estão disponíveis, na Figura 35 observa-se uma tabela de utilizadores populada com alguns dados teste:

user_id	user_name	user_num	pw	level
1	Master	0	tese	1
2	Slave1	10000	tese1	0
3	Slave2	20000	tese2	1
4	Slave3	30000	tese3	1

Figura 35 - Amostra de utilizadores populada com dados exemplares.

Obtendo estes dados pode-se verificar que quando um cliente se liga ao sistema, este executa autenticação correctamente. Nas Figuras 36 e 37 testa-se o módulo de autenticação a funcionar correctamente.

```
[NEW CONNECTION] ('192.168.1.10', 56282) connected.
User found: Master
Password is Correct
```

Figura 36 - Teste de autenticação válida.

```
[NEW CONNECTION] ('192.168.1.10', 56286) connected.
User found: Master
Wrong Password: Master
```

Figura 37 - Teste de autenticação inválida.

Para verificar se o sistema de armazenamento a curto prazo se encontra a funcionar correctamente realiza-se um teste, ligando o simulador e deixando-o a funcionar. O simulador está a funcionar com o número de amostras armazenadas a curto prazo pré-definida como três. Na Figura 38 consegue-se observar que o número de amostras armazenadas é mantido.

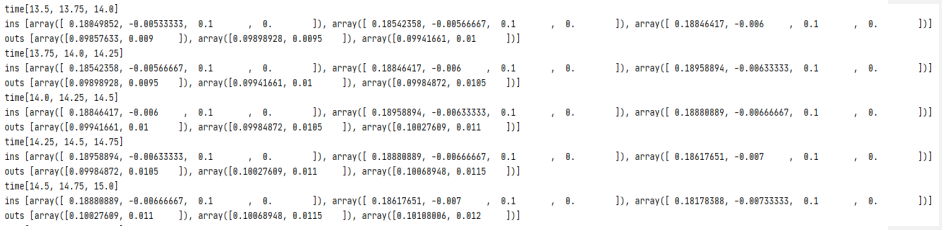


Figura 38 - Teste de memória a curto prazo, verifica-se que os vectores *time, ins* e *outs* se encontram sempre com o tamanho pré-definido para o armazenamento.

Em seguida analisa-se o comportamento do sistema, verificando se o simulador mantém o tempo de amostragem independentemente de o cliente se atrasar perante o próprio tempo de funcionamento do simulador. Na Figura 39 conseguimos verificar que o cliente se encontra a simular com um tempo de amostragem de 0.1 segundos, podemos que em algumas destas iterações o simulador entra em acção com o seu mecanismo para manter os intervalos de amostragem correctos sem esperar por açções de controlo de um cliente.

```
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Auto Mode
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Simulating with client
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
Auto Mode
Tempo de amostragem calculado pela diferencas de duas iteracoes consecutivas
0.1
```

Figura 39 Verificação de manutenção de tempo de amostragem, e que o tempo sistema não depende de um cliente.

Apesar do simulador e o cliente se encontrarem com o mesmo intervalo de amostragem notam-se atrasos no controlo. Após alguns testes e acertos considerou-se que o intervalo de amostragem mínimo, para a plataforma de simulação funcionar correctamente com clientes ligados, é de $0.25s$. Realizaram-se testes em que o intervalo de amostragem de simulação é inferior ao intervalo de comunicação, verifica-se nestes casos que a plataforma de simulação actua bem. O atraso nas comunicações é que faz com que o ritmo com que os comandos do cliente são transmitidos à plataforma não seja constante, ou desejado. Na Figura 37 consegue-se observar esses eventos, ocorrem onde se consegue ver escrito "*Auto Mode*".

De modo a realizar testes em que a interacção com o cliente não requer o modo automático, foi definido o intervalo de amostragem referido e reiniciado o sistema como consta na Figura 40.

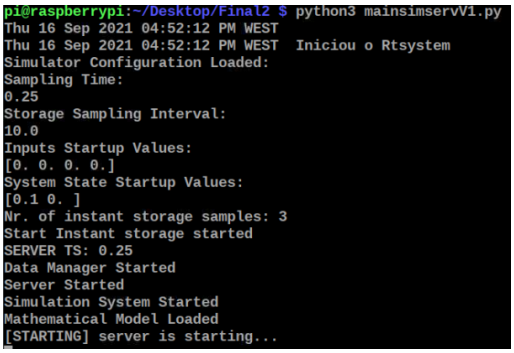


Figura 40 - Inicialização de sistema com configurações diferentes, (de notar tempo de amostragem mínimo a utilizar de $0.25s$).

Foi utilizado um modelo de cliente desenvolvido para testar o sistema. Na Figura 41 observa-se a interface de cliente desenvolvida para executar um teste. Este tem três modos: Teste de simulador(funcional), acesso ao processo real (trabalho futuro), desligar do sistema de simulação.

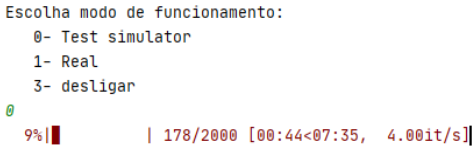


Figura 41 - Interface de cliente a executar controlo do simulador.

5.2- Teste da plataforma com cliente a executar controlos simples em *Windows* num computador

Após a introdução do modelo físico-matemático dos tanques no sistema este foi testado apenas com entradas e saídas, com degraus nas entradas, em anel aberto, enviados pelo cliente. de modo a verificar que o sistema se encontrava a ter o comportamento desejado, garantindo assim a sua validade. Este teste foi realizado apenas enviando vectores que contêm os dados das entradas, que neste caso representam válvulas, do sistema com degraus, para posteriormente obter o comportamento correcto da instalação. Na Figura 42 conseguem-se ver os objectivos do primeiro teste onde se pretende perceber se o sistema se encontra funcional, e se é possível ser controlado remotamente.

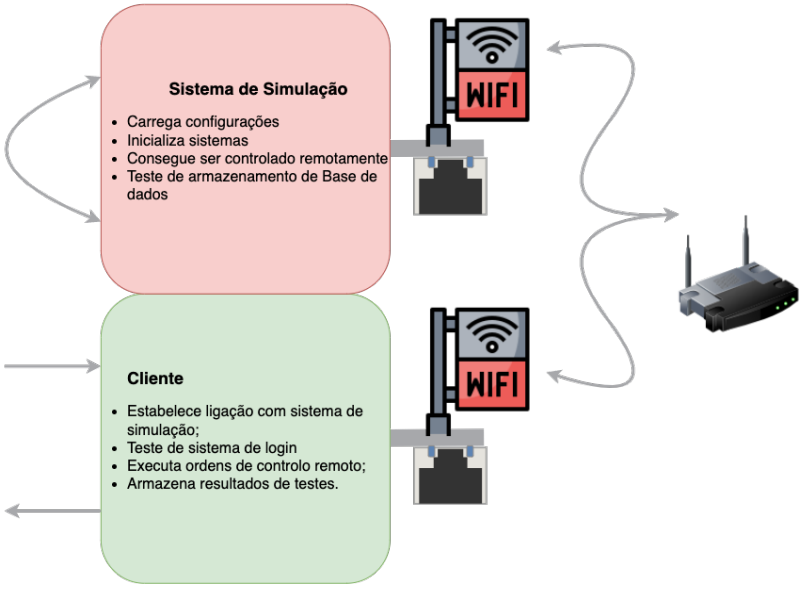


Figura 42 - Teste inicial de verificação de funcionamento do sistema.

Ao executar o teste verificam-se os resultados obtidos na Figura 43. Conseguem-se ver os resultados deste teste, onde no modelo de tanques se verifica o enchimento dos mesmos quando as válvulas de entrada se encontram abertas e

as de saída fechadas, e o esvaziamento quando as válvulas de entrada se encontram fechadas e a de saída se encontra aberta.

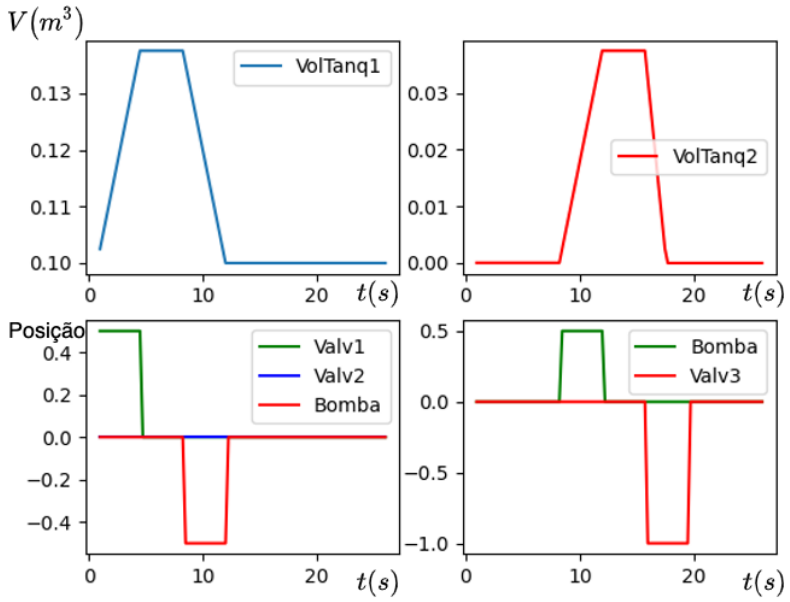
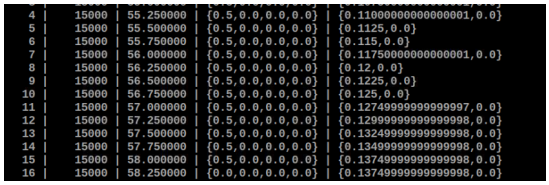


Figura 43 - Resultado de testes simples com vectores de entrada zeros e uns.

O passo seguinte foi testar os mesmos vectores do teste anterior, e verificar se os dados ficaram armazenados correctamente e que o intervalo de armazenamento, definido no ficheiro de configuração inicial é cumprido. Com tal teste conseguimos mostrar também que a base se encontra operacional. Uma vez que ao realizar simulações o sistema armazena no intervalo de amostragem predefinido. Reinicializa-se à base de dados, realiza-se a simulação com condições do teste anterior. Acede-se a base de dados pelo terminal e realiza-se uma pesquisa dos dados armazenados. Na Figura 44 observam-se os dados obtidos onde conseguimos verificar pelas diferenças temporais nos tempos de amostragem de cada linha da base de dados, conseguindo ver que os intervalos estão correctos.



4	15000	55.250000	{0.5,0.0,0.0,0.0}	{0.11000000000000001,0.0}
5	15000	55.500000	{0.5,0.0,0.0,0.0}	{0.1125,0.0}
6	15000	55.750000	{0.5,0.0,0.0,0.0}	{0.115,0.0}
7	15000	56.000000	{0.5,0.0,0.0,0.0}	{0.11750000000000001,0.0}
8	15000	56.250000	{0.5,0.0,0.0,0.0}	{0.12,0.0}
9	15000	56.500000	{0.5,0.0,0.0,0.0}	{0.1225,0.0}
10	15000	56.750000	{0.5,0.0,0.0,0.0}	{0.125,0.0}
11	15000	57.000000	{0.5,0.0,0.0,0.0}	{0.12749999999999997,0.0}
12	15000	57.250000	{0.5,0.0,0.0,0.0}	{0.12999999999999998,0.0}
13	15000	57.500000	{0.5,0.0,0.0,0.0}	{0.13249999999999998,0.0}
14	15000	57.750000	{0.5,0.0,0.0,0.0}	{0.13499999999999998,0.0}
15	15000	58.000000	{0.5,0.0,0.0,0.0}	{0.13749999999999998,0.0}
16	15000	58.250000	{0.0,0.0,0.0,0.0}	{0.13749999999999998,0.0}

Figura 44 - Leitura da base de dados de simulação simples.

Na Figura 44 tem-se por colunas:

- Identificador de registo, formato sequência, inteiro;
- Número de utilizador formato inteiro;
- Instante de amostragem;
- Entradas do modelo de simulação;
- saídas do modelo de simulação. Podemos ver que válvula 1 se encontra com valor de 0.5, e as restantes a nulas.

Verificando que todo o sistema funciona correctamente, ou seja, o simulador corre como é previsto no seu instante e intervalo de amostragem pré-definido, os dados se encontram armazenados, os clientes se conseguem ligar, e utilizar o simulador.

Altera-se o sistema de cliente introduzindo um controlador PID (proporcional integrador derivador), este está controlar a $Valv_1$, que actua no enchimento do primeiro tanque. São definidos valores de referência para podermos testar o sistema e verificar se tanto os dados foram armazenados correctamente, assim como uma resposta correcta do sistema de controlo por parte do simulador. A Figura 45 descreve os testes alvo:

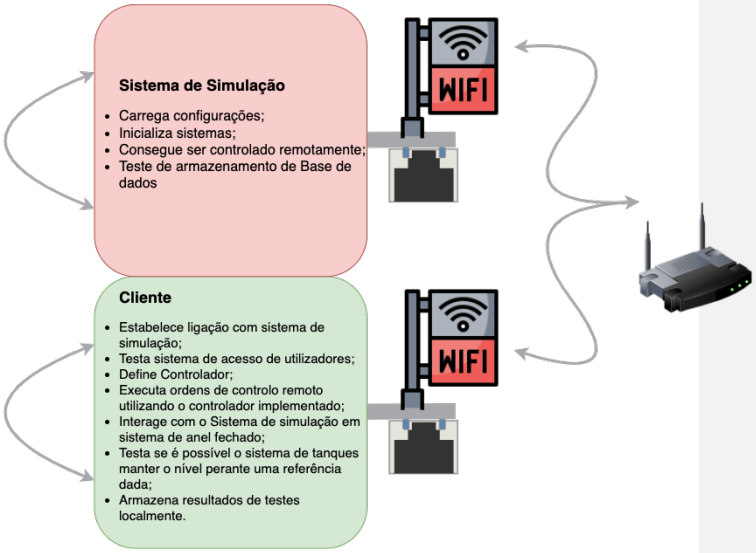


Figura 45 Detalhes dos testes de simulação com controlador PID

Com o controlador PID implementado foi realizado o teste onde se definiram os valores, do lado do cliente, de volume a $0.1m^3, 0.4m^3$ *e* $0.3m^3$. Na Figura 46 consegue-se ver os resultados do teste definido. No primeiro gráfico observa-se a cor de laranja os valores de referência definidos e a azul o volume do tanque no sistema de simulação. No segundo gráfico tem-se a verde a válvula controlada pelo cliente para alcançar a referência, e a azul a válvula dois em modo de descarga com um sinal de rampa. A válvula dois está definida deste modo para ver se o controlador consegue seguir a referência mesmo que existam perturbações.

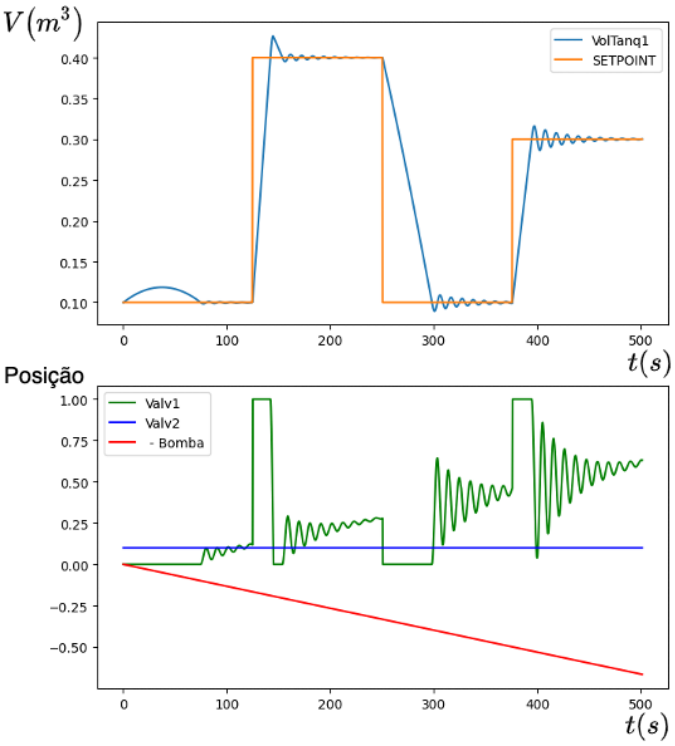


Figura 46 - Teste de sistema com controlador PID remoto no cliente.

É de notar que a $Bomba$ encontra-se com valores em recta decrescente para verificar se o controlador corrige a perturbação com a válvula onde está a actuar. Foi adoptado um valor negativo ($-Bomba$) de modo que os resultados fossem mais legíveis.

Os resultados dos testes com o controlador PID foram os que se pretendiam, o volume do tanque seguiu a referência definida, e respondeu às actuações das válvulas um e dois.

Na Figura 47 consegue-se observar o seguimento da referência com uma escala ampliada, e assim reparar que a diferença entre a referência e o volume dos tanques é pequena.

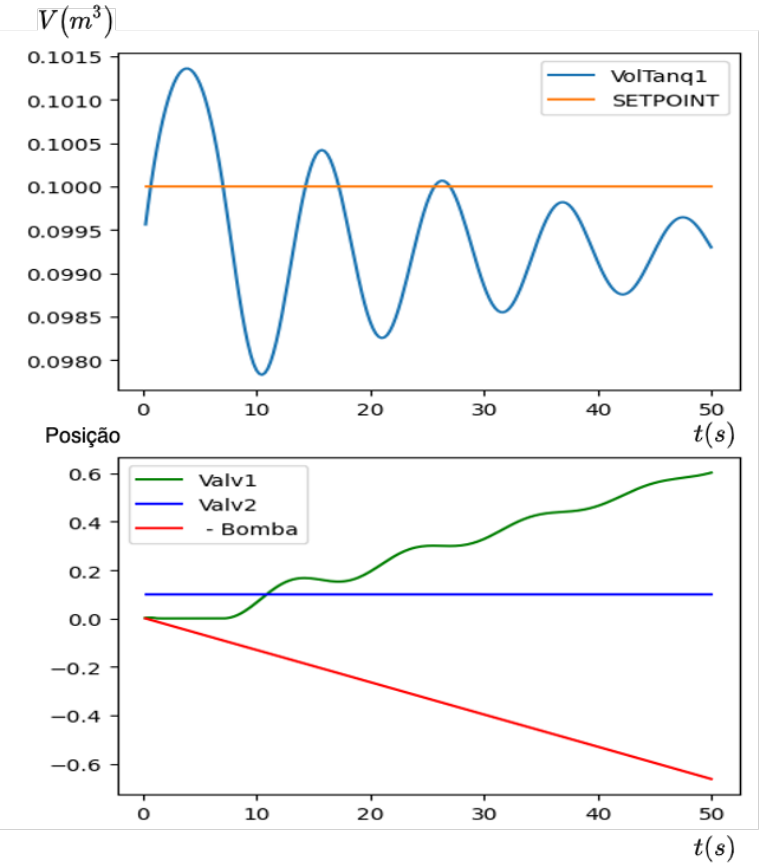


Figura 47 - Teste com 200 amostras de controlador PID.

5.3- Teste da plataforma com controlador PID remoto em cliente em *Android,* num *smartphone*

Adicionalmente foi realizado um teste com um cliente em sistema operativo Android. Este teste demonstra a capacidade de o sistema de simulação implementado interagir com clientes de diferentes sistemas operativos. Na Figura 48 consegue-se ver que o valor do volume é seguido pelo volume definido como

referência do sistema, após a acção de controlo gerada pelo controlador implementado no cliente.

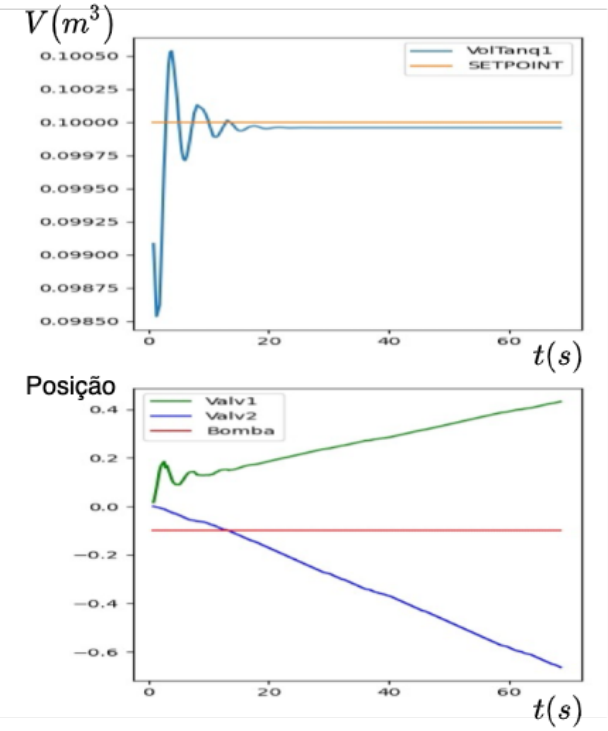


Figura 48 - Teste de cliente com simulador, com valor de referência a 0.1m^3,280 amostras 0.25s.

Neste ensaio o sinal de referência é constante. A abertura da válvula 1 corresponde à reacção do sistema de controlo à abertura da válvula 2.

Foi também realizado um teste mais prolongado com as configurações de cliente definidas na Figura 46. Neste constam diferentes valores de referência (0.1,0.4,0.3):

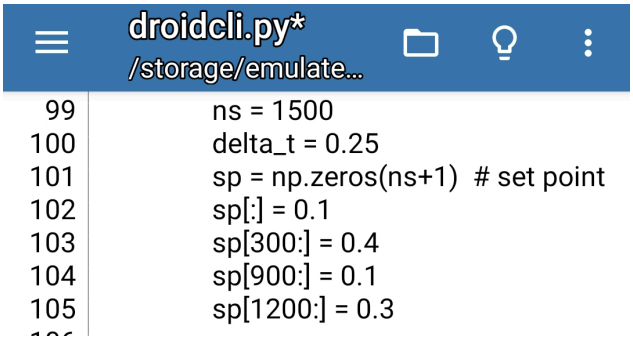


Figura 49 - Configurações em cliente *Android*.

Com as configurações que constam na Figura 49 o sistema de simulação conseguiu executar as acções de comando sobre o modelo matemático de instalação a simular. O processo foi de fácil configuração e adaptação para conseguir o estudo de controladores. Na Figura 50 conseguimos ver os resultados obtidos a partir das configurações no cliente que constam na Figura 49. Estes resultados mostram que é possível executar o controlo remoto do sistema implementado.

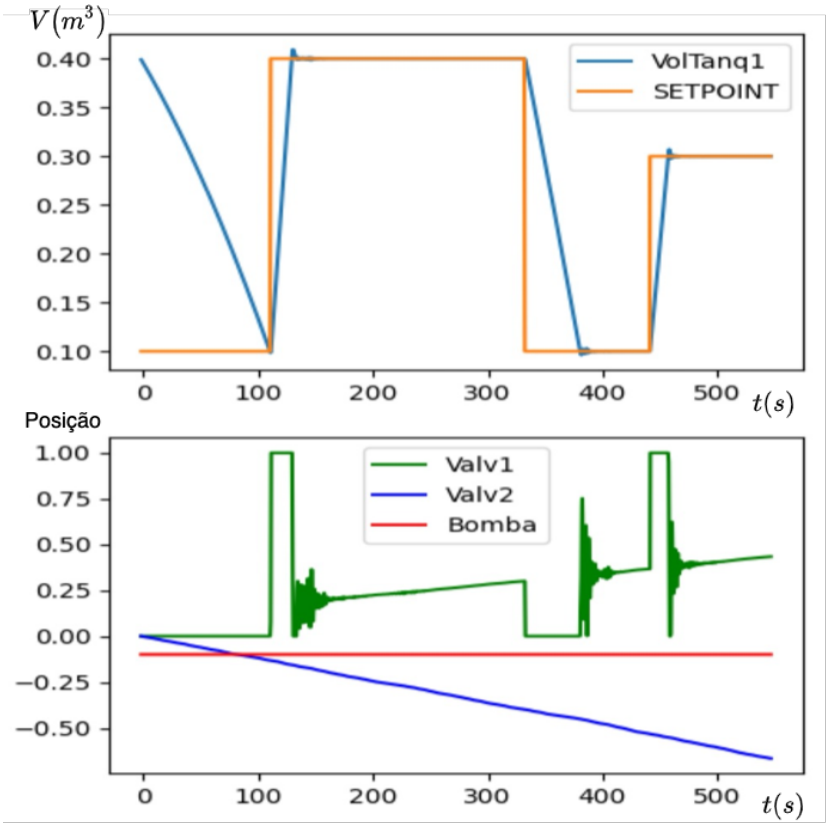


Figura 50 Resultados de simulação de controlador remoto, obtidos a partir do sistema implementado.

Assim se mostra que o sistema pode ser implementado num pequeno microcomputador, e de baixo custo para realizar experiências de controlo remoto.

5.4 Implementação Futura

Não foram implementados os módulos de ligação de hardware e gestão de processo real, e módulo de visualização.

A plataforma encontra-se preparada para interagir com um processo real. No entanto a parte de gestão de entradas e saídas físicas por *software* não se realizou. Não foi possível por não ter sido desenvolvido o módulo de ligação de hardware e dos circuitos conversores AD/DA. Na Figura 51 mostra-se a arquitectura do sistema completo.

As funcionalidades adicionais a implementar serão:

- Gestor de processo real;
- Ligações de cabos e conversores AD/DA;
- Introdução de uma tabela no sistema de bases de dados com os estados da instalação física;
- Adicionar uma funcionalidade em que o gestor de processo real possa carregar configurações do módulo de configuração implementado.

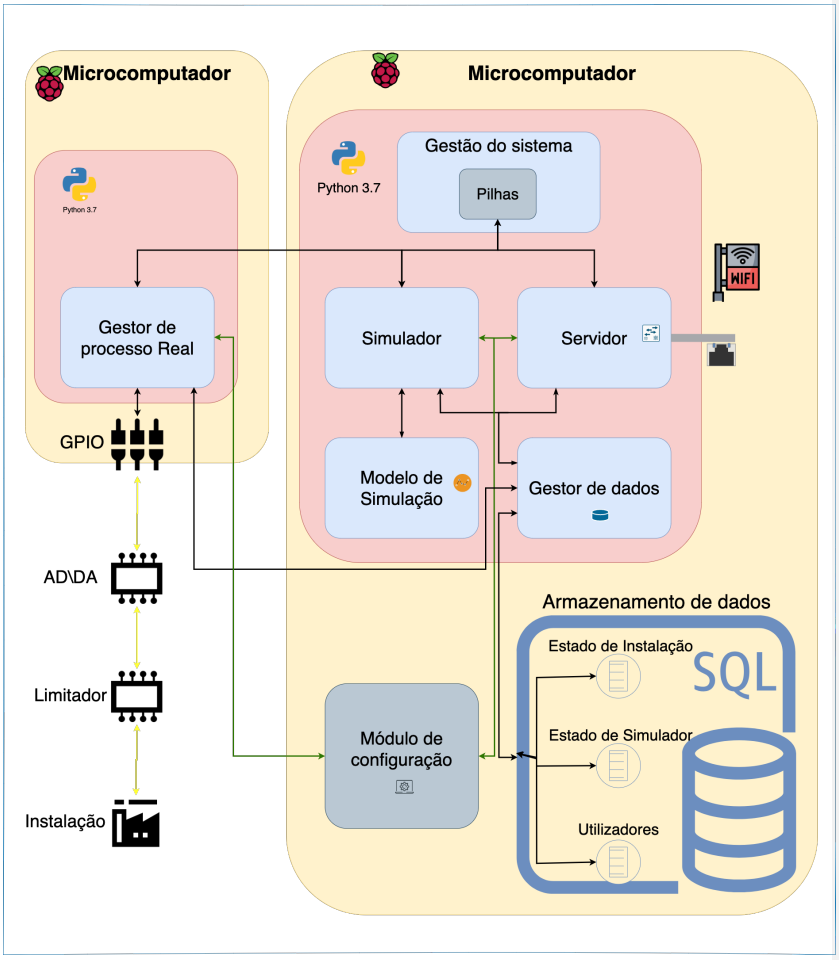


Figura 51 - Arquitectura do sistema completo

Adicionando os módulos referidos o sistema irá ser uma plataforma de simulação e actuação remota, assim como poderá receber acções de controlo através dos seus portas físicas para controlar o modelo em simulação.

6

Conclusões

6.1 - Objectivos alcançados

Neste trabalho foram alcançados os seguintes objectivos:

- Capacidade de comunicação no modelo servidor-clientes;

- Desenvolvimento de uma plataforma para simulação de processos industriais com capacidade de funcionar com ou sem clientes ligados;

- Desenvolvimento de uma interface de armazenamento de dados;

- Desenvolvimento de um sistema de mensagens transversal a todo o sistema, como para os clientes;

- Desenvolvimento de ficheiros de configuração.

Capacidade de comunicação servidor-cliente

Conseguiu-se mostrar que foi executado um sistema de comunicação entre o sistema e o exterior. O acesso que foi desenvolvido foi com base em protocolo TCP/IP. Foi elucidado que o sistema é versátil e consegue comunicar com clientes em diferentes ambientes.

Desenvolvimento de um simulador de processos industriais com capacidade de funcionar com ou sem clientes ligados

A arquitectura realizada neste trabalho gerou a possibilidade de utilizar um sistema desenvolvido com diferentes modelos físico-matemáticos. Este está dimensionado para funcionar de modo autónomo (recorrendo a valores de

iterações anteriores), ou ligado a um cliente. O cliente pode estar, ou não, a controlar o sistema mediante as devidas autorizações.

Esta arquitectura revelou-se funcional com os testes apresentados nos resultados. Foi testada com um modelo de cliente, em que este realizou testes de anel aberto e testes, em anel fechado com controladores PID a controlarem o modelo remotamente.

Desenvolvimento de uma interface de armazenamento de dados

A arquitectura implementada contemplou, em adição ao desenvolvimento de software em Python, a utilização de um sistema de base de dados (*PostgreSQL*). Ao utilizar a BD relacional para interligar o *Python* e garantir o armazenamento consistente, tal revelou-se uma tarefa de grau de dificuldade baixo. Esta também é acessível a partir da rede, com as devidas credencias de acesso. As bases de dados permitem manter os dados armazenados de forma uniforme e consistente.

Desenvolvimento de um sistema de mensagens transversal a todo o sistema, como para os clientes

Neste trabalho foi criado um sistema de mensagens, onde as mensagens que são transmitidas entre os processos tem um formato transversal entre esses mesmos processos. Mais ainda as mensagens enviadas por clientes para comunicar com o sistema devem ter o mesmo formato que as mensagens trocadas entre processos do servidor. Isto facilita a uniformização de dados que circulam pelo sistema. A uniformização de dados garante que o armazenamento de dados, o sistema de autenticação e o sistema de simulação sejam possíveis de implementar, e alem disso executar simulações de diversos modelos físico-matemáticos.

Desenvolvimento de ficheiros de configuração

De modo a conseguir que o sistema implementado tivesse uma fácil configuração foi criado um ficheiro de configuração com os parâmetros referidos anteriormente. Este ficheiro é depois carregado pelos diferentes módulos do sistema. Assim consegue-se que um administrador execute alterações em configurações base do sistema, como intervalos de amostragem para simulações ou portas de acesso do simulador, ou outras configurações que possam ser definidas posteriormente estejam acessíveis num único local.

Estes objectivos conjugados permitiram alcançar o objectivo principal que era conseguir desenvolver um simulador industrial, preparado para executar modelos físico-matemáticos de sistemas industriais, de baixa complexidade controlados via rede, ou em funcionamento autónomo.

Utilizar a linguagem Python para realizar a arquitectura do sistema, a criação de um sistema transversal de mensagens entre cliente e todo o sistema garantiu que fosse possível executar testes com modelo de cliente tanto a partir de um telefone equipado com o sistema operativo Android, como em sistemas Linux, Windows e MacOS, demonstrando assim a versatilidade de funcionamento e interacção do sistema. O objectivo de garantir um armazenamento a longo prazo foi alcançado como elucidado anteriormente, concluindo que é possível realizar diferentes simulações, armazenar na mesma base de dados simulações de diferentes modelos, assim como de diferentes utilizadores.

O método como a arquitectura foi desenvolvida permite com poucas alterações às configurações correr diferentes instâncias do sistema a correr, em paralelo. Tendo múltiplas instâncias será possível que clientes interajam com diferentes modelos ao mesmo tempo. Assim consegue-se aumentar o grau de complexidade do sistema.

6.2-Trabalhos Futuros

Como trabalhos adicionais consideram-se os seguintes:

- Desenvolvimento de módulo de gestão de processo real;
- Desenvolvimento do módulo de hardware para ligação a processo real;
- Desenvolvimento de ambiente gráfico em tempo real;
- Testes com outros modelos.

O desenvolvimento do módulo de gestão considera-se importante para poder ligar o sistema a um processo físico. A arquitectura já contempla um modo no gestor de mensagens para lidar com mensagens de processos físicos. Será necessária desenvolver o módulo de *software* que liga o *hardware* ao sistema de simulação. O desenvolvimento deste módulo gera conectividade ao sistema, uma vez que habilita o sistema de ligação directa com o mundo físico que o rodeia, através de sensores ou actuadores.

De modo a comunicar com o processo real será também necessário desenvolver as ligações de hardware entre o *Raspberri Pi 3* e a instalação. Para isso será necessário estabelecer as ligações entre conversores analógico-digitais. Isso é possível recorrendo aos protocolos SPI e I2C, e os AD/DA referidos no Estado de Arte.

Conjugando o módulo de gestão com as ligações de hardware pode -se criar um sistema físico controlável, ou controlar o modelo da instalação a simular por portos físicos.

O desenvolvimento de um ambiente gráfico em tempo real é uma adição que trará uma maior legibilidade a quem está a administrar o sistema em tempo real. O sistema foi desenvolvido com uma memória de curto prazo que corre enquanto o simulador do sistema estiver em execução, e também um sistema de BD, com armazenamento a longo prazo. Ambos os sistemas permitem o acesso e retorno de dados. No caso da base de dados é possível extrair dados da mesma para fazer estudos posteriores, de modo a observar os comportamentos do sistema NCS.

O sistema deverá ainda ser testado com outros modelos físico-matemáticos para poderem ser verificadas que melhorias de desempenho poderão ser obtidas. A capacidade do sistema de mensagens desenvolvido deve ser explorada para serem desenvolvidas novas funcionalidades.

Formatted: Portuguese

6

Bibliografia

[1] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou, "Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1531–1539, 2013, doi: 10.1016/j.jnca.2012.12.028.

[2] P. Galkin, L. Golovkina, and I. Klyuchnyk, "Analysis of Single-Board Computers for IoT and IIoT Solutions in Embedded Control Systems," *2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings*, pp. 297–302, 2019, doi: 10.1109/INFOCOMMST.2018.8632069.

[3] J. Sobota, M. Goubej, J. Königsmarková, and M. Čech, "Raspberry Pi-based HIL simulators for control education," *IFAC-PapersOnLine*, vol. 52, no. 9, pp. 79–84, 2019, doi: 10.1016/j.ifacol.2019.08.126.

[4] "What Is MATLAB? - MATLAB & Simulink." https://www.mathworks.com/discovery/what-is-matlab.html (accessed Mar. 12, 2021).

[5] R. Barber, D. R. Rosa, and S. Garrido, "Adaptive control of a dc motor for educational practices," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 10, no. PART 1, pp. 244–249, 2013, doi: 10.3182/20130828-3-UK-2039.00060.

[6] "BeagleBone Black Support from Embedded Coder - Hardware Support - MATLAB & Simulink." https://www.mathworks.com/hardware-support/beaglebone-black.html?s_tid=srchtitle (accessed Aug. 09, 2021).

[7] "Arduino Support from Simulink - Hardware Support - MATLAB & Simulink." https://www.mathworks.com/hardware-support/arduino-simulink.html?s_tid=srchtitle (accessed Aug. 09, 2021).

[8] E. Basil and S. D. Sawant, "IoT based traffic light control system using Raspberry Pi," *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017*, pp. 1078–1081, 2018, doi: 10.1109/ICECDS.2017.8389604.

[9] "Simulink Support Package for Raspberry Pi Hardware - File Exchange - MATLAB Central." https://www.mathworks.com/matlabcentral/fileexchange/40313-simulink-support-package-for-raspberry-pi-hardware (accessed Mar. 12, 2021).

[10] D. Quaglia, R. Muradore, R. Bragantini, and P. Fiorini, "A SystemC/Matlab co-simulation tool for networked control systems," *Simulation Modelling Practice and Theory*, vol. 23, pp. 71–86, 2012, doi: 10.1016/j.simpat.2012.01.003.

[11] "Raspberry Pi DC Motor Driver » File Exchange Pick of the Week - MATLAB & Simulink." https://blogs.mathworks.com/pick/2013/05/03/raspberry-pi-dc-motor-driver/ (accessed Mar. 12, 2021).

[12] "Pricing and Licensing - MATLAB & Simulink." https://www.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=comm (accessed Mar. 19, 2021).

[13] S. Kumar, "Overview of Modeling of Delays in Networked Control Systems," vol. 1, no. 11, pp. 74–79, 2014.

[14] R. M. Reck and R. S. Sreenivas, "Developing a new affordable DC motor laboratory kit for an existing undergraduate controls course," *Proceedings of the American Control Conference*, vol. 2015-July, pp. 2801–2806, 2015, doi: 10.1109/ACC.2015.7171159.

[15] Chugani and L. Mahesh, *LabVIEW Signal Processing*. Prentice Hall PTR, 1998.

[16] "LabVIEW Front Panel Explained - NI." https://www.ni.com/en-us/support/documentation/supplemental/08/labview-front-panel-explained.html (accessed Mar. 19, 2021).

[17] I. González, A. J. Calderón, A. Mejías, and J. M. Andújar, "Novel networked remote laboratory architecture for open connectivity based on PLC-OPC-Lab VIEW-EJS integration. Application in remote fuzzy control and sensors data acquisition," *Sensors (Switzerland)*, vol. 16, no. 11, 2016, doi: 10.3390/s16111822.

[18] "LabVIEW Release Notes - NI." https://www.ni.com/pt-pt/support/documentation/release-notes/product.labview.html (accessed Aug. 09, 2021).

[19] W. Zeng and M. Y. Chow, "Optimal tradeoff between performance and security in networked control systems based on coevolutionary algorithms," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 7, pp. 3016–3025, 2012, doi: 10.1109/TIE.2011.2178216.

[20] "Arduino™-Compatible Compiler for LabVIEW Download - NI." https://www.ni.com/pt-pt/support/downloads/tools-network/download.arduino--compatible-compiler-for-labview.html#379003 (accessed Aug. 09, 2021).

[21] "LabVIEW Hobbyist Toolkit Download - NI." https://www.ni.com/pt-pt/support/downloads/tools-network/download.labview-hobbyist-toolkit.html#376574 (accessed Aug. 09, 2021).

[22] "Advantages and Disadvantages of LabVIEW - Viewpoint Systems." https://www.viewpointusa.com/labview/advantages-and-disadvantages-of-labview/ (accessed Mar. 22, 2021).

[23] R. Kolla, Z. Wang, Z. Miao, and L. Fan, "Realization of Enhanced Phase Locked Loop using Raspberry Pi and LabVIEW," *51st North American Power Symposium, NAPS 2019*, pp. 1–6, 2019, doi: 10.1109/NAPS46351.2019.9000278.

[24] "Select Your LabVIEW Edition - NI." https://www.ni.com/en-us/shop/labview/select-edition.html (accessed Mar. 19, 2021).

[25] P. K. Karmore and G. L. Girhe, "Programming Language Python : a Review," *Ijariie*, no. 2, pp. 1634–1637, 2020.

[26] "General Python FAQ — Python 3.9.2 documentation." https://docs.python.org/3/faq/general.html#why-is-it-called-python (accessed Mar. 19, 2021).

[27] K. R. Srinath, "Python – The Fastest Growing Programming Language," *International Research Journal of Engineering and Technology*, pp. 354–357, 2017.

[28] K. J. Millman and M. Aivazis, "Python for scientists and engineers," *Computing in Science and Engineering*, vol. 13, no. 2, pp. 9–12, 2011, doi: 10.1109/MCSE.2011.36.

[29] G. van Rossum and J. de Boer, "Interactively Testing Remote Servers Using the Python Programming Language," *CWI Quarterly*. pp. 283–303, 1991.

[30] J. Xu and M. Frydenberg, "Python Programming in an IS Curriculum: Perceived and Outcomes," *Communications of the Association for Information Systems*, vol. 44, no. January, pp. 123–155, 2019.

[31] M. Sahani, C. Nanda, A. K. Sahu, and B. Pattnaik, "Home Security System Based on Face Recognition," *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pp. 1–6, 2015.

[32] B. G. Olivier, J. M. Rohwer, and J. H. S. Hofmeyr, "Modelling cellular systems with PySCeS," *Bioinformatics*, vol. 21, no. 4, pp. 560–561, 2005, doi: 10.1093/bioinformatics/bti046.

[33] B. Delhaisse, L. Rozo, and D. G. Caldwell, "PyRoboLearn : A Python Framework for Robot Learning Practitioners," *CoRL*, no. CoRL, pp. 1–11, 2019.

[34] G. Fedel, D. Beniz, L. do Carmo, and J. Piton LNLS, "Python for User Interfaces at Sirius; Python for User Interfaces at Sirius," 2017, doi: 10.18429/JACoW-ICALEPCS2017-THAPL04.

[35] "designer-multiple-screenshot.png (907×515)." https://doc.qt.io/qt-5/images/designer-multiple-screenshot.png (accessed Mar. 29, 2021).

[36] H. Abubakar, A. A. Bello, and H. Dandakouta, "Development and Validation of a Pythonbased Simulation Program for Energetic and Exergetic Analysis of Heat Exchangers," *International Journal of Mechanical Engineering*, vol. 7, no. 7, pp. 1–9, 2020, doi: 10.14445/23488360/ijme-v7i7p101.

[37] K. R. Thorp and K. F. Bronson, "A model-independent open-source geospatial tool for managing point-based environmental model simulations at multiple spatial locations," *Environmental Modelling and Software*, vol. 50, pp. 25–36, 2013, doi: 10.1016/j.envsoft.2013.09.002.

[38] G. Van Rossum and F. L. Drake, "The Python Library Reference," *October*, pp. 1–1144, 2010.

[39] C. Kumar Sahu and P. Behera, "A low cost smart irrigation control system," *2nd International Conference on Electronics and Communication Systems, ICECS 2015*, no. Icecs, pp. 1146–1151, 2015, doi: 10.1109/ECS.2015.7124763.

[40] "Psycopg – PostgreSQL database adapter for Python — Psycopg 2.8.7.dev0 documentation." https://www.psycopg.org/docs/ (accessed Mar. 23, 2021).

[41] J. Jacky, "EPICS-BASED CONTROL SYSTEM FOR A RADIATION THERAPY MACHINE," pp. 922–925.

[42] J. N. H, A. Jerine, R. K. D, B. Joseph, and M. E. A, "Implementation of a keypoint database for Diwata-1 image autogeoreferencing : key ideas , experiences and issues," 2015.

[43] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[44] J. R. Dormand and P. J. Prince, "A reconsideration of some embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 15, no. 2, pp. 203–211, 1986, doi: 10.1016/0377-0427(86)90027-0.

[45] M. Morshed and T. Mahmud, "source Packages," 2020.

[46] "configparser — Configuration file parser — Python 3.9.2 documentation." https://docs.python.org/3/library/configparser.html (accessed Mar. 29, 2021).

[47] R. May, "Porting Radar Simulation Software to Python : A Case Study in the Benefits of Python".

[48] "gpiozero — GPIO Zero 1.6.2 Documentation." https://gpiozero.readthedocs.io/en/stable/index.html (accessed Apr. 01, 2021).

[49] V. Margapuri, "Smart Motion Detection System Using Raspberry Pi."

[50] B. Balon and M. Simic, "Using raspberry Pi computers in education," *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, no. March 2018, pp. 671–676, 2019, doi: 10.23919/MIPRO.2019.08756967.

[51] Maiti and Bidinger, *済無No Title No Title*, vol. 53, no. 9. 1981.

[52] R. Neves-Silva, "Controlo de Sistemas Dinamicos - Tempo contínuo e tempo discreto," pp. 1–318, 2008.

[53] G. Bianchini, "Synthesis of robust strictly positive real discrete-time systems with l2 parametric perturbations," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 8, pp. 1221–1225, 2002, doi: 10.1109/TCSI.2002.801271.

[54] B. D. O. Anderson, "A System Theory Criterion for Positive Real Matrices," *SIAM Journal on Control*, vol. 5, no. 2, pp. 171–182, May 1967, doi: 10.1137/0305011.

[55] C. V. Hollot, L. Huang, and Z.-L. Xu, "Designing Strictly Positive Real Transfer Function Families: A Necessary and Sufficient Condition for Low Degree and Structured Families," in *Robust Control of Linear Systems and Nonlinear Control*, Birkhäuser Boston, 1990, pp. 215–227. doi: 10.1007/978-1-4612-4484-4_19.

[56] P. Naghshtabrizi and J. P. Hespanha, "Designing an observer-based controller for a network control system," *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, vol. 2005, pp. 848–853, 2005, doi: 10.1109/CDC.2005.1582263.

[57] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Suvey of Recent Results In Networked Control Systems," *Bernoulli*.

[58] "2560px-Raspberry_Pi_4_Model_B_-_Side.jpg (2560×1507)." https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Raspberry_Pi_4_Model_B_-_Side.jpg/2560px-Raspberry_Pi_4_Model_B_-_Side.jpg (accessed Aug. 10, 2021).

[59] R. Foundation, "raspberrypi." https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (accessed Jan. 13, 2020).

[60] RASPBERRY PI FOUNDATION, "Raspberry Pi 4 Model B specifications – Raspberry Pi." https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ (accessed Mar. 19, 2021).

[61] "Microcomputador Raspberry Pi 3 Model B 1.2GHz CPU, 1GB RAM, WiFi/BLE, 40 GPIO Pins." https://mauser.pt/catalog/product_info.php?products_id=096-6085 (accessed Apr. 06, 2021).

[62] "BeagleBoard.org - AI." https://beagleboard.org/ai (accessed Nov. 21, 2020).

[63] Cdn.antratek, "Channel with Programmable Gain Amplifier." https://cdn.antratek.nl/media/product/675/ads1115-16-bit-adc-4-channel-with-pga-ada-1085-f92.jpg (accessed Jan. 09, 2020).

[64] D. Information and S. B. Diagrams, "Ads1113," 2018.

[65] "e9b4e3a7d78d3de81d7da1449a4b18d8.png (684×460)." https://www.microchip.com/_images/products/medium/e9b4e3a7d78d3de81d7da1449a4b18d8.png (accessed Aug. 16, 2021).

[66] Microchip Technology Inc., "Mcp4902/4912/4922 8/10/12-Bit Dual Voltage Output Digital-to-Analog Converter with SPI Interface," pp. 1–48, 2010.

[67] "What Is MongoDB? | MongoDB." https://www.mongodb.com/what-is-mongodb (accessed Mar. 23, 2021).

[68] S. Agarwal and K. S. Rajan, "Performance analysis of MongoDB versus PostGIS/PostGreSQL databases for line intersection and point containment spatial queries," *Spatial Information Research*, vol. 24, no. 6, pp. 671–677, 2016, doi: 10.1007/s41324-016-0059-1.

[69] J. S. Van Der Veen, B. Van Der Waaij, and R. J. Meijer, "Sensor data storage performance: SQL or NoSQL, physical or virtual," *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, pp. 431–438, 2012, doi: 10.1109/CLOUD.2012.18.

[70] S. Daniel-Berhe and H. Unbehauen, "Physical parameters estimation of the nonlinear continuous-time dynamics of a DC motor using Hartley modulating functions method," *Journal of the Franklin Institute*, vol. 336, no. 3, pp. 481–501, 1999, doi: 10.1016/S0016-0032(98)00043-X.

[71] P. Dostál, V. Bobál, and J. Vojtěšek, "Simulation of cascade control of a continuous stirred tank reactor," *Proceedings - 29th European Conference on Modelling and Simulation, ECMS 2015*, no. June 2009, pp. 266–272, 2015, doi: 10.7148/2015-0266.

[72] H. Kaneriya, "Design Of I2C Master With Multiple Slave," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 5, pp. 2890–2893, 2015.

[73] M. Maemunah and M. Riasetiawan, "The Architecture of Device Communication in Internet of Things Using Inter-Integrated Circuit and Serial Peripheral Interface Method," *Proceedings - 2018 4th International Conference on Science and Technology, ICST 2018*, pp. 19–22, 2018, doi: 10.1109/ICSTC.2018.8528663.

[74] "3.9.2 Documentation." https://docs.python.org/3/ (accessed Mar. 16, 2021).

[75] "JSON." https://www.json.org/json-en.html (accessed Mar. 16, 2021).

[76] M. S. N. E. T. Core and H. K. Dhalla, "A Performance Analysis of Native JSON Parsers in," vol. 2, p. 8, 2020.

[77] "scipy.integrate.solve_ivp — SciPy v1.6.1 Reference Guide." https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html#r179348322575-2 (accessed Mar. 29, 2021).