Universidade do Minho
Escola de Engenharia

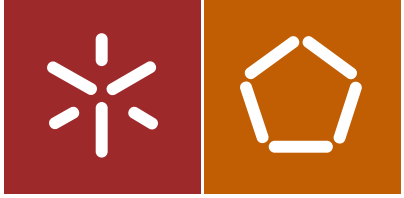
Fábio Rafael Freitas Morais

Desenvolvimento de um Sistema de Gestão de Cargas para Smart Grids

novembro de 2015

Fábio Rafael Freitas Morais Desenvolvimento de um Sistema de Gestão de Cargas para Smart Grids

UMinho | 2015



Universidade do Minho
Escola de Engenharia

Fábio Rafael Freitas Morais

Desenvolvimento de um Sistema de Gestão de Cargas para Smart Grids

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Eletrónica e de Computadores

Trabalho efectuado sob a orientação do
Professor Doutor João Luiz Afonso

e coorientação do
Professor Doutor Adriano Tavares

novembro de 2015

DECLARAÇÃO

Fábio Rafael Freitas Morais

Endereço eletrónico: a55712@alunos.uminho.pt Telefone: 932782582

Número do Bilhete de Identidade: 13725332

Título da Tese:

Desenvolvimento de um Sistema de Gestão de Cargas para *Smart Grids*

Orientador:

Doutor João Luiz Afonso

Coorientador:

Doutor Adriano Tavares

Ano de conclusão: 2015

Dissertação submetida na Universidade do Minho para a obtenção do grau de

Mestre em Engenharia Eletrónica e de Computadores

Universidade do Minho, ___/___/_____

Assinatura: _____

Ao meu Pai.

Agradecimentos

A realização do trabalho aqui apresentado não teria sido possível sem o apoio e a contribuição de algumas pessoas, às quais transmito os meus mais sinceros agradecimentos:

Ao meu orientador Doutor João Luiz Afonso, pela dedicação, empenho e interesse demostrado durante a realização deste trabalho, bem como pelo apoio, concelhos sugestões e críticas apresentadas durante a orientação.

Ao meu coorientador Doutor Adriano Tavares, pela dedicação, empenho e interesse demostrado durante a realização deste trabalho, bem como pelo apoio, concelhos sugestões e críticas apresentadas durante a orientação.

Aos meus colegas e amigos, bolseiros de investigação Gabriel Pinto, Vítor Monteiro, Delfim Pedrosa, Bruno Exposto e Raul Almeida, pelo excelente ambiente vivido no laboratório bem como por toda a ajuda prestada ao longo da elaboração deste trabalho.

A todos os alunos que realizaram a dissertação de mestrado no Laboratório de Eletrónica de Potência, pelo ambiente de respeito e de amizade proporcionados.

Um agradecimento muito especial ao meu pai Carlos Morais, pelo incondicional apoio, compreensão e sacríficos realizados para que eu pudesse completar esta etapa da minha vida.

Resumo

A energia elétrica é um bem cada vez mais essencial dado que está quase omnipresente nas sociedades atuais. Por isso a necessidade de tornar a produção, transporte e distribuição mais fiável, eficiente e sustentável.

As redes elétricas são atualmente a base sobre a qual as sociedades assentam, fornecendo a estas a energia necessária para todas as suas atividades, sejam elas domésticas ou industriais. No entanto, as novas realidades e desafios que estas enfrentam requerem que estas evoluam para que sejam capazes de se adaptarem a esses desafios. A evolução das redes atuais apresenta-se sob a forma das *Smart Grids*, redes elétricas inteligentes que irão integrar as mais recentes tecnologias para poderem satisfazer as necessidades de energia ao mesmo tempo que melhoram a fiabilidade, eficiência e sustentabilidade das redes que já existem atualmente.

Uma das inovações que as *Smart Grids* virão trazer é o controlo do consumo, sendo este um dos aspetos importantes neste tipo de redes, uma vez que vai permitir à rede elétrica alterar até um certo ponto o consumo de energia elétrica de forma que este se adapte à capacidade de produção da rede. Isto irá criar muitas vantagens quer para a rede quer para os consumidores.

O trabalho desta dissertação irá incidir no desenvolvimento de um sistema de gestão de cargas para operar numa *Smart Grid*. Tendo por isso como objetivo que este seja capaz de interagir com o utilizador, rede elétrica e cargas de forma que o funcionamento destas últimas vá de encontro às condições da rede ao mesmo tempo que reduz o custo que o funcionamento destas cargas trazem para o utilizador, fazendo assim com que o sistema implementado seja capaz de ser integrado em futuros programas de gestão de consumo.

Nesta dissertação é apresentada uma análise do que atualmente se entende por gestão de consumo e quais são os objetivos principais deste tipo de gestão. É também apresentado um levantamento sobre as *Smart Grids* onde estes estão inseridos, sendo depois projetado o sistema de gestão de cargas de acordo com as informações recolhidas. Este é então simulado, implementado e testado de forma a demostrar que este é capaz de desempenhar a função para o qual foi concebido.

Palavras-Chave: Sistema de Gestão de Cargas, *Smart Grids*, *Field Programmable Gate Array* (FPGA), Filtros Ativos, Carros elétricos, Energias Renováveis.

Abstract

The electric network is becoming much more essential since it´s almost omnipresent in today´s society. Because of that the need to make production, transportation and distribution more reliable, efficient and sustainable.

The electric grids are, nowadays, the base in which the societies are sustained, giving them the necessary energy for all the activities, being them domestic or industrial. However, the new realities and challenges that they face require them to evolve, so that they can be capable of adaption to those challenges. The evolution of today's grids present themselves as Smart Grids, that are intelligent electric grids that will integrate the newest technology to satisfy the energy requirements, at the same time that they improve reliability, efficiency and sustainability of the grids that already exist.

One of the improvements that the Smart Grids will bring is the consumption control, being this one of the most relevant aspects of this type of grids, since it will allow the electric grid to alter to a certain degree the electric energy consumption in doing so, it will adapt to the production ability of the grid. Bringing a lot of advantages to the grid as well to the consumers.

This dissertation project will focus on the development of a charge's management system to operate in a Smart Grid. It pursues two main objectives: (1) friendly interaction with the user, the electric grid and changes in order to improve the grids' conditions while reducing the cost to the user and (2) implementation of module to be integrated in the future programs of consumption management.

In this dissertation is presented an analysis of what, nowadays, is understood as consumption management and what are the main objectives of this type of managemet. It will be also presented a research about Smart Grids, where they are inserted, being then projected a charge's management system according to the found information. This will be simulated, implemented and testes to demonstrate that it is capable of doing what it was designed to do.

Keywords: Load Management System, Smart Grid, Field Programmable Gate Array (FPGA), Active Filters, Electric Vehicles, Renewable Energy.

Índice

Agradecimentos ..... **v**

Resumo ..... **vii**

Abstract ..... **ix**

Lista de Figuras ..... **xiii**

Lista de Tabelas ..... **xvi**

Lista de Siglas e Acrónimos ..... **xvii**

CAPÍTULO 1 Introdução **1**

1.1. Futuro da Rede Elétrica 1
1.2. Motivações 2
1.3. Objetivos e Contribuições 2
1.4. Organização e Estrutura da Dissertação 3

CAPÍTULO 2 Sistemas de Gestão de Cargas **5**

2.1. Introdução 5
2.2. Gestão de Cargas 6
2.2.1. Vantagens da Gestão de Cargas 6
2.3. Requisitos 8
2.3.1. Sistemas de Gestão de Cargas 8
2.3.2. Sistemas de Gestão de Cargas Existentes no Mercado 10
2.3.2.1. G Smart 11
2.3.2.2. SACE Emax 2 with Ekip Power Controller 12
2.3.2.3. Wiser Home Management 14
2.3.3. Infraestruturas de Comunicação 15
2.3.4. Medidores Avançados 16
2.3.5. Algoritmos de Otimização e Previsão 18
2.4. *Smart Grids* 19
2.4.1. Caraterísticas das *Smart Grids* 19
2.4.2. Vantagens das *Smart Grids* 21
2.4.3. Desafios à Implementação das *Smart Grids* 22
2.5. Conclusão 23

CAPÍTULO 3 Projeto do Sistema de Gestão de Cargas para *Smart Grids* **25**

3.1. Introdução 25
3.2. Descrição das Funcionalidades Pretendidas 25
3.3. Descrição do Sistema 26
3.4. Aplicação de Interface com o Utilizador 27
3.4.1. Sistema Operativo Android 28
3.4.2. Android Studio 29
3.4.3. Linguagem de Programação Java 30
3.4.4. Especificação da Aplicação 31
3.5. Servidor 35
3.5.1. Raspberry Pi 35
3.5.2. Linguagem de Programação Python 37
3.5.3. IDLE 38
3.5.4. Especificação do Servidor 38
3.6. Sistema de Gestão de Cargas (SGC) 40
3.6.1. Basys 2 40
3.6.2. Verilog 41
3.6.3. Especificação do Sistema de Gestão de Cargas (SGC) 42
3.7. Conclusão 54

CAPÍTULO 4 Simulação do Sistema de Gestão de Cargas Desenvolvido **57**

4.1. Introdução 57
4.2. Simulação do Módulo de Controlo de Cargas 57
4.3. Simulação do Sub-Módulo Responsável pelas Prioridades 58
4.3.1. Verificação da Entrada de comando DEV (Device) 58
4.3.2. Verificação da Entrada de comando LOC (Local) 59
4.3.3. Verificação da Entrada de comando NET (Network) 61
4.3.4. Verificação da Entrada de comando RT_CLOCK (Real Time Clock) 61

4.3.5. Verificação do funcionamento do sub-módulo das prioridades 62
4.4. Simulação do Módulo Responsável pela Descodificação de Instruções 64
4.5. Simulação do Sub-Módulo Responsável pelo Real Time Clock 71
4.6. Simulação do módulo para controlo de cargas com todos os sub-módulos já integrados 74
4.7. Conclusão 80

CAPÍTULO 5 Implementação do Sistema de Gestão de Cargas para *Smart Grids* 83

5.1. Introdução 83
5.2. Implementação da Aplicação de Interface com o Utilizador 83
5.3. Implementação do Servidor 86
5.4. Implementação do Sistema de Gestão de Cargas 89
5.5. Conclusão 91

CAPÍTULO 6 Teste do Sistema de Gestão de Cargas para *Smart Grids* 93

6.1. Introdução 93
6.2. Teste à Aplicação de Interface com o Utilizador 93
6.3. Teste ao Servidor 100
6.4. Teste ao Sistema Gestão de Cargas 107
6.4.1. Teste ao Módulo de Comunicação 107
6.4.2. Teste ao Módulo Completo 109
6.5. Conclusão 113

CAPÍTULO 7 Conclusão 115

7.1. Conclusões 115
7.2. Sugestões para Trabalho Futuro 117

Referências 119

Lista de Figuras

Figura 2.1 - Interações do gestor de cargas. 9
Figura 2.2 - Interações do gestor de cargas servindo de mediador para o utilizador. 10
Figura 2.3 - *G Smart* [16] 11
Figura 2.4 – Disjuntor *SACE Emax 2* [17]. 13
Figura 2.5 – Sistema de gestão de consumo *Wiser Home Management*, figura adaptada de [18]. . 14
Figura 2.6 - Infraestruturas de comunicação. 16
Figura 2.7 - Distribuição dos medidores ao longo da rede. 17
Figura 2.8 - Interação entre medidor e gestor de cargas a nível local. 18
Figura 3.1 - Sistema a Implementar. 27
Figura 3.2 - Diagrama de sequência do processo de *login*. 32
Figura 3.3 - Diagrama de sequência do processo de *logout*. 32
Figura 3.4 - Formato da *string* a enviar para o servidor. 33
Figura 3.5 - Diagrama de sequência do processo de enviar. 33
Figura 3.6 - Diagrama de sequência do processo de pedir informação. 34
Figura 3.7 - Diagrama de sequência do processo de definir dia e horas atuais. 35
Figura 3.8 - Placa Basys2 [46]. 40
Figura 3.9 - Sub-módulo das prioridades. 42
Figura 3.10 - Atribuição de comandos aos registos. 43
Figura 3.11 - Sub-módulo de descodificação e processamento. 44
Figura 3.12 - Formato do comando de alteração de estado. 45
Figura 3.13 - Fluxograma do comando de alteração de estado. 46
Figura 3.14 - Formato do comando de alteração de prioridade. 46
Figura 3.15 - Fluxograma do comando de alteração de prioridade. 47
Figura 3.16 - Formato do comando de alteração de horário. 47
Figura 3.17 - Fluxograma do comando de alteração de horário. 48
Figura 3.18 - Formato do comando para o preço de alteração. 48
Figura 3.19 - Fluxograma do comando de preço de alteração. 49
Figura 3.20 - Formato do comando de alteração do tipo de controlo. 49
Figura 3.21 - Fluxograma do comando de alteração do tipo de controlo. 49
Figura 3.22 - Fluxograma dos comandos de informação. 50
Figura 3.23 - Formato do comando de comparação de preço. 50
Figura 3.24 - Fluxograma do comando de comparação de preço. 51
Figura 3.25 - Formato do comando para a definição do dia e da hora. 51
Figura 3.26 - Fluxograma do comando de definição de dia e hora. 51
Figura 3.27 - Sub-Módulo do relógio. 52
Figura 3.28 - Sub-módulo das comunicações. 53
Figura 4.1 - Simulação da entrada dev fase um. 59
Figura 4.2 - Simulação da entrada dev fase dois. 59

Figura 4.3 - Simulação da entrada loc fase um. 60
Figura 4.4 - Simulação da entrada loc fase dois. 60
Figura 4.5 - Simulação da entrada net fase um. 61
Figura 4.6 - Simulação da entrada net fase dois. 61
Figura 4.7 - Simulação da entrada rt_clk. 62
Figura 4.8 – Simulação do módulo completo fase um. 63
Figura 4.9 - Simulação do módulo completo fase dois. 63
Figura 4.10 - Simulação do módulo completo fase três. 64
Figura 4.11 - Simulação do módulo completo fase quatro. 64
Figura 4.12 - Simulação do módulo de descodificação de instruções fase um. 65
Figura 4.13 - Simulação do módulo de descodificação de instruções fase dois. 66
Figura 4.14 - Simulação do módulo de descodificação de instruções fase três. 67
Figura 4.15 - Simulação do módulo de descodificação de instruções fase quatro. 67
Figura 4.16 Simulação do módulo de descodificação de instruções fase cinco. 68
Figura 4.17 - Simulação do módulo de descodificação de instruções fase seis. 68
Figura 4.18 - Simulação do módulo de descodificação de instruções fase sete. 69
Figura 4.19 - Simulação do módulo de descodificação de instruções fase oito. 69
Figura 4.20 Simulação do módulo de descodificação de instruções fase nove. 70
Figura 4.21 - Simulação do módulo de descodificação de instruções fase dez. 70
Figura 4.22 - Simulação do módulo de descodificação de instruções fase onze. 71
Figura 4.23 - Simulação do módulo RT_CLK fase um. 72
Figura 4.24 – Simulação do módulo RT_CLK fase dois. 72
Figura 4.25 - Simulação do módulo RT_CLK fase três. 73
Figura 4.26 - Simulação do módulo RT_CLK fase quatro. 73
Figura 4.27 - Simulação do módulo RT_CLK fase cinco. 74
Figura 4.28 - Simulação do módulo completo - fase um. 75
Figura 4.29 - Simulação do módulo completo - fase dois. 75
Figura 4.30 - Simulação do módulo completo - fase três. 76
Figura 4.31 - Simulação do módulo completo - fase quatro. 76
Figura 4.32 - Simulação do módulo completo - fase cinco. 77
Figura 4.33 - Simulação do módulo completo - fase seis. 77
Figura 4.34 - Simulação do módulo completo - fase sete. 78
Figura 4.35 – Simulação do módulo completo - fase oito. 78
Figura 4.36 - Simulação do módulo completo - fase nove. 79
Figura 4.37 - Simulação do módulo completo - fase dez. 79
Figura 4.38 - Simulação do módulo completo - fase onze. 80
Figura 5.1 - Seleção do Hardware a utilizar. 84
Figura 5.2 - Seleção da versão do sistema operativo. 84
Figura 5.3 - Seleção das caraterísticas da máquina virtual. 85
Figura 5.4 - Ecrã de desenho da aplicação. 85
Figura 5.5 - Gestor do SDK. 86

Figura 5.6 - Assistente de instalação NOOBS. 87
Figura 5.7 - Consola de configuração do Raspi-config. 88
Figura 5.8 – *Login* no ambiente de trabalho remoto utilizando o VNC Viewer. 88
Figura 5.9 - Definições do projeto no ISE. 89
Figura 5.10 - Utilização de recursos da FPGA. 90
Figura 5.11 – Mapeamento das entradas e saídas utilizando o PlanAhead. 90
Figura 5.12 – Programação da FPGA utilizando o Diligent Adept. 91
Figura 6.1 - Ícone da aplicação. 94
Figura 6.2 - Ecrã inicial. 94
Figura 6.3 – Ecrã de autenticação. 95
Figura 6.4 – Mensagem de erro na palavra-chave. 95
Figura 6.5 – Mensagem de erro no utilizador. 95
Figura 6.6 - Ecrã de seleção de cargas. 96
Figura 6.7 - Mensagem de confirmação de *logout*. 96
Figura 6.8 – Exemplo de *NumberPicker* utilizado para introdução de valores. 97
Figura 6.9 – Exemplo de *Spinner* utilizado para seleção de opções. 97
Figura 6.10 - Ecrã de gestão do carro elétrico. 98
Figura 6.11 - Mensagem de erro na definição das prioridades. 98
Figura 6.12 - Mensagem com a seleção do comando a enviar. 99
Figura 6.13 - Mensagem de notificação do estado da execução do comando. 99
Figura 6.14 – Novos valores dos elementos recebidos através do comando de informação. 100
Figura 6.15 – Ecrã de definição de dia e hora. 100
Figura 6.16 - Tabela dos utilizadores. 101
Figura 6.17 – Processo de *login* e *logout*. 102
Figura 6.18 - Execução dos comandos alterar estado e alterar prioridade. 103
Figura 6.19 - Execução dos comandos alterar horário e alterar preço. 104
Figura 6.20 - Execução dos comandos alterar tipo de controlo e alterar estado de carga mínimo. 105
Figura 6.21 - Execução dos comando de definição de dia e hora. 106
Figura 6.22 - Execução de um pedido de informação ao SGC. 107
Figura 6.23 - Resultados da comunicação na placa com 100MHz de relógio. 108
Figura 6.24 - Resultados da comunicação na placa com 50MHz de relógio. 109
Figura 6.25 - *Byte* 3 (mais significativo) do comando de alteração de estado que se encontra à saída do SGC. 110
Figura 6.26 - *Byte* 2 do comando de alteração de estado que se encontra à saída do SGC. 111
Figura 6.27 - *Byte* 1 do comando de alteração de estado que se encontra à saída do SCG. 111
Figura 6.28 - *Byte* 0 do comando de alteração de estado que se encontra à saída do SGC. 111
Figura 6.29 - *Byte* referente ao Estado obtido através do comando de informação. 112
Figura 6.30 - *Byte* referente ao Estado de Carga Atual obtido através do comando de informação. 112
Figura 6.31 - *Byte* referente ao Capacidade de Carga Utilizada obtido através do comando de informação. 113
Figura 6.32 - *Byte* referente ao Fator de Potência obtido através do comando de informação. 113

Lista de Tabelas

Tabela 3.1 - Lista das versões do sistema operativo Android. 29
Tabela 3.2 - Lista das diferentes versões da Raspberry Pi. 36
Tabela 3.3 - Organização do *Byte* de prioridades. 43
Tabela 3.4 - Lista de Instruções. 45
Tabela 3.5 - Comandos específicos das cargas. 53
Tabela 3.6 - Estados das cargas. 54

Lista de Siglas e Acrónimos

SGC	Sistema de Gestão de Cargas
FPGA	*Field Programmable Gate Array*
V2G	*Vehicle-to-Grid*
API	*Application Programming Interface*
ADT	*Android Development Tools*
IDE	*Integrated Development Environment*
SDK	*Software Development Kit*
AVD	*Android Virtual Devices*
USB	*Universal Serial Bus*
SHA	*Secure Hash Algorithm*
ROM	*Read-Only Memory*
LED	*Light Emitting Diode*
VGA	*Video Graphics Array*
XML	*eXtensible Markup Language*
NOOBS	*New Out of the Box Software*
SSH	*Secure Shell*
IP	*Internet Protocol*
UCF	*User Constraints File*
PROM	*Programmable Read-Only Memory*

CAPÍTULO 1

Introdução

1.1. Futuro da Rede Elétrica

O conceito de *Smart Grid* tem sido cada vez mais abordado e, por essa razão, a ideia de uma rede de distribuição de energia elétrica inteligente é cada vez mais apelativa e, acima de tudo, necessária. A liberalização dos mercados energéticos, o crescimento das formas de energia renováveis, a geração distribuída de energia e o advento dos veículos elétricos tornam necessária uma evolução na rede de distribuição de energia. Esta evolução apresenta-se na forma de uma rede de energia elétrica flexível, capaz de promover a eficiência (operacional e económica), a sustentabilidade, a robustez, assim como a capacidade de aumentar os serviços prestados ao utilizador e ao mesmo tempo promover uma maior contribuição deste, para o bom funcionamento da rede [1].

Para que isto seja possível, é necessário que este novo tipo de rede de distribuição seja dotado de uma maior capacidade de recolha de dados, de capacidade de transferência de informação de forma bidirecional e de gestão automatizada e remota.

A recolha de dados deve abranger todos, ou quase todos, os aspetos que influenciam a rede, tais como, o consumo, como estes são distribuídos ao longo do tempo, a qualidade da energia ao longo da rede, a quantidade de energia que está a ser produzida, o preço da energia, os tipos de cargas ligadas à rede, dados meteorológicos para ajudar a prever os níveis de produção dos diferentes tipos de energia renováveis e até o estado dos componentes da própria rede. Toda esta informação deve ser adquirida em períodos de tempo curtos, para que os agentes que atuam sobre a rede o possam fazer em tempo útil, não só para manter o bom funcionamento da rede mas também para tirar o máximo proveito possível desta.

No cenário em que a distribuição de energia é feita através de uma *Smart Grid*, a gestão que é feita sobre a rede terá de sofrer alterações em relação ao que é feito atualmente. Esta passará a ser baseada numa quantidade de informação muito maior do que anteriormente e a frequência com que a informação é adquirida será também aumentada. Isto permitirá novas possibilidades de intervenções na rede, uma vez que ao

ter informação acerca de um maior conjunto de parâmetros também será possível estes mesmos parâmetros serem controlados. Com este aumento no fluxo de informação e em parâmetros a controlar, torna-se necessário que a gestão sobre a rede seja não só remota, mas também automatizada para que as ações de gestão possam ser realizadas de forma mais rápida e eficiente. Como exemplo, do que foi mencionado anteriormente, existe a gestão de cargas ligadas à rede. As cargas ligadas à rede podem ser geridas de forma a tirar o melhor proveito possível dos preços de eletricidade, mas também para aliviar a carga da rede em momentos em que o consumo atinge o seu pico. Esta estratégia irá evitar gastos em capacidade de produção para satisfazer esses picos de consumo e este tipo de gestão tem de ter em conta não só as necessidades da rede, mas principalmente as preferências do cliente ao qual a carga pertence [1][2][3][4].

A segurança será um dos aspetos fundamentais nos equipamentos das redes inteligentes, visto que estes serão responsáveis pela monitorização da rede e de grande parte da atuação, por isso é absolutamente necessário que se tomem medidas para os proteger [4]. Devido a isto, todos os dispositivos envolvidos nas operações deste tipo de rede, terão obrigatoriamente de possuir comunicações seguras de forma que os dados que circulam não possam ser intercetados, acedidos ou até mesmo alterados de forma indevida. Os próprios dispositivos devem apresentar também um elevado nível de segurança pelas mesmas razões.

1.2. Motivações

As *Smart Grids* oferecem várias oportunidades e benefícios para a sociedade. No entanto, ainda há muito trabalho a ser realizado para que estas possam ser implementadas de forma que todos os benefícios idealizados se tornem realidade. Um desses trabalhos é o desenvolvimento de equipamentos capazes de fazer a gestão de cargas ligadas à rede para que estas possam contribuir para o bom funcionamento da mesma. Para além destes dispositivos oferecerem à rede novas oportunidades para melhorar a sua eficiência, também vão promover a proliferação das energias renováveis ao ajudar a colmatar algumas das suas desvantagens. O consumidor irá igualmente beneficiar ao melhorar a eficiência do seu consumo ou mesmo tempo que poupa na fatura da eletricidade.

1.3. Objetivos e Contribuições

Tal como sugere o título, o objetivo global desta dissertação consiste no desenvolvimento de um Sistema de Gestão de Cargas para uma *Smart Grid*, que tem como

propósito gerir cargas de forma que estas possam contribuir para o bom funcionamento da rede. Assim, os principais objetivos a atingir podem ser resumidos nos seguintes tópicos:

- Projeto do Sistema de Gestão de Cargas para uma *Smart Grid*;
- Seleção das cargas a gerir;
- Seleção dos parâmetros a gerir nessas cargas;
- Seleção das comunicações a usar;
- Simulação do sistema a desenvolver;
- Desenvolvimento da interface com o utilizador;
- Desenvolvimento de um servidor para receber pedidos dos utilizadores e da rede; Desenvolvimento do sistema de gestão de cargas;
- Realização de testes ao sistema desenvolvido.

Com esta dissertação pretende-se contribuir para o desenvolvimento de equipamentos capazes de atuar numa *Smart Grid* na área de gestão de consumo, de forma a tornar real este novo aspeto das redes elétricas. Contribuindo assim, para o desenvolvimento das *Smart Grids* e através destas, para uma melhoria na eficiência, sustentabilidade e rentabilidade do mercado da energia elétrica.

1.4. Organização e Estrutura da Dissertação

Esta dissertação encontra-se dividida em sete capítulos que se encontram estruturados da seguinte forma:

No primeiro capítulo é feito o enquadramento desta dissertação, onde são expostas as motivações para a realização desta, assim como os seus objetivos e contribuições.

O segundo capítulo é onde é efetuado o levantamento do estado da arte sobre os sistemas de gestão de cargas e as *Smart Grids*, onde estes estarão inseridos. São também, aqui inclusivamente apresentados, exemplos de sistemas de gestão de cargas já existentes no mercado.

O terceiro capítulo é onde é efetuado o projeto do sistema, usando como base as informações recolhidas no capítulo anterior sobre sistemas de gestão de cargas e o meio onde estes vão estar inseridos.

No quarto capítulo é realizada a simulação do sistema, de forma a verificar que este está de acordo com o que foi projetado e pronto para ser implementado, sendo as simulações efetuadas por partes de forma isolar possíveis problemas.

A implementação do sistema é abordada no quinto capítulo, onde são apresentadas todas as fases concretizadas para implementar as diferentes partes do sistema, sendo os passos apresentados divididos de acordo com a parte do sistema a que pertencem.

Os testes realizados ao sistema são apresentados no sexto capítulo, sendo que estes testes também estão segmentados pelas diferentes partes que fazem parte do sistema, de forma a isolar possíveis erros e para validar cada parte individualmente, antes de se proceder ao teste de integração.

Por fim, as conclusões e sugestões de trabalho futuro são apresentadas no sétimo capítulo, finalizando assim esta dissertação.

CAPÍTULO 2

Sistemas de Gestão de Cargas

2.1. Introdução

As redes elétricas são sistemas complexos que apareceram pela primeira vez no final do século dezanove e com o passar do tempo têm assumido cada vez mais um papel fundamental. Estes sistemas representam hoje em dia a base sobre a qual a sociedade se apoia. Não é possível para alguém atualmente desenvolver as suas atividades quotidianas sem recorrer ao uso de eletricidade, sejam elas atividades lúdicas ou profissionais. Tal como as sociedades onde estão inseridas as redes elétricas têm vindo a evoluir para melhor responder às necessidades destas que se encontram em constante mutação. Como tal, novos desafios surgem constantemente e as redes elétricas têm de ser capazes de se adaptar de forma eficiente e sólida, pois a importância destas na sociedade não permite margem para erros [5].

Uma das evoluções que se prevê para as redes elétricas é a incorporação de um sistema de gestão de consumo nas mesmas. A gestão de consumo é a capacidade da rede ou dos seus gestores, de influenciar e direcionar o consumo de energia elétrica de forma que este seja mais eficiente e sustentável, trazendo assim vantagens não só para a rede, mas também para o consumidor que colabora neste esforço No entanto, esta evolução está inserida num conjunto de alterações mais abrangentes às redes, que é o migrar das redes atuais para as denominadas *Smart Grids* que irão conter bastantes mais evoluções, para além da gestão de consumo.

Ao longo deste capítulo vai ser descrito o que é a gestão de consumo, as suas vantagens e os requisitos para a sua correta implementação. Serão também apresentadas características para sistemas capazes de realizar esta tarefa, bem como exemplos deste tipo de sistemas que já se encontram disponíveis no mercado Por fim, será feita a descrição das redes em que estes vão estar inseridos (*Smart Grids*) e que outros elementos as constituem de forma a contextualizar o tipo de rede elétrica em que se prevê inserir a gestão de consumo.

2.2. Gestão de Cargas

No panorama atual é cada vez mais importante uma rede elétrica eficiente. Existem diversos métodos que contribuem para este objetivo, no entanto a maioria destes são postos em prática pelas companhias que produzem ou distribuem a energia elétrica. Todavia, torna-se cada vez mais vantajoso e necessário que se incluam neste esforço os clientes, uma vez que a colaboração destes pode ter um enorme impacto na eficiência da rede.

O contributo dos consumidores para a eficiência e o bom funcionamento da rede elétrica ainda não é significativo [6], mas começam a ser feitos esforços para mudar esta realidade. Cada vez mais têm surgido preocupações em tornar o consumidor mais ativo na gestão da rede. Um dos métodos que se tem evidenciado para atingir este objetivo é controlar a procura de energia elétrica de forma a tornar esta mais flexível com o intuito de assim poderem ser melhor acomodadas as necessidades e características da rede [7]. Uma vez que nos sistemas de produção e distribuição de energia elétrica a produção tem de ser sempre igual à procura, tornar esta última mais flexível traz diversas vantagens para todos os intervenientes.

2.2.1. Vantagens da Gestão de Cargas

A grande vantagem de realizar a gestão das cargas ligadas à rede é a possibilidade de dar uma maior flexibilidade ao consumo de energia elétrica, que até agora é caracterizada por padrões fixos que obrigatoriamente têm de ser correspondidos pela produção de energia. Outra das grandes vantagens é a capacidade de coordenação do funcionamento das cargas de forma a promover a máxima eficiência da rede.

Ao flexibilizar o consumo de energia torna-se possível alterá-lo de forma a satisfazer mais eficientemente as necessidades da rede ou simplesmente para que o uso das capacidades desta seja mais eficiente [8][9]. Isto pode ser alcançado movendo a utilização de algumas cargas das horas de maior consumo para horas de menor, reduzindo assim a diferença entre estes períodos.

Desta forma, é possível reduzir picos de procura de energia, e os períodos em que a energia a ser produzida ultrapassa a procura. Os picos de procura de energia são prejudiciais porque obrigam a investimentos em infraestruturas e capacidade de produção para que estes possam ser atendidos [7]. Este facto diminuí a eficiência da rede, uma vez que esta vai estar sobredimensionada numa grande parte do seu tempo de funcionamento, com a exceção dos períodos em que se verificam estes picos de consumo [8].

Este problema pode ser resolvido, ou pelo menos minimizado, se a utilização das cargas for gerida de uma forma que o consumo acompanhe, dentro do possível, a capacidade de produção [10].

No entanto, com o aumento da geração através de energias renováveis a própria capacidade de produção varia ao longo do tempo, devido a fontes de energias como eólica e solar que não são constantes. Para além disto, é difícil prever a capacidade de produção destas, e apenas o é possível fazer a curto prazo num espaço de dias [11], o que torna a flexibilidade oferecida pela gestão de cargas ainda mais importante e vantajosa, visto que esta irá permitir tirar um melhor proveito das energias renováveis [10]. Através da gestão de cargas será possível dar prioridade ao consumo de energia nos momentos em que a produção desta através de fontes renováveis é maior [12], o que reduz a dependência de combustíveis fósseis para a geração de energia, fator que ajuda a proteger o ambiente.

A flexibilidade que a gestão de cargas vem trazer proporciona várias vantagens e oportunidades para as intervenções que são feitas na rede. Contudo, para que se possa tirar o máximo partido desta flexibilidade, esta tem ser coordenada ao nível de toda a rede mas também a nível local. A coordenação é essencial para garantir que todas as cargas são geridas de forma que os objetivos gerais são atingidos, sendo os principais entre estes a eficiência e bom funcionamento da rede.

A vantagem de poder coordenar todas, ou quase todas, as cargas existentes na rede é que estas passarão a ser utilizadas da forma mais eficiente possível, ou seja, a sua utilização será orientada por princípios comuns, sendo os mais importantes o bom funcionamento da rede e a priorização do consumo de energia produzida através de energias renováveis [13]. Estes princípios têm como objetivo assegurar o correto funcionamento da rede sem que sejam prejudicadas as funções que as cargas devem desempenhar. A coordenação necessária nesta gestão passa por organizar o funcionamento das cargas para este ir de encontro às necessidades da rede. É também necessário fazer a coordenação com os utilizadores a quem as cargas pertencem.

A interação com os utilizadores é muito importante porque para alterar o funcionamento das cargas que lhes pertencem é necessário a aceitação destes. Para isto, é fundamental que os requisitos destes sejam respeitados. Isto traduz-se em cumprir as instruções dos utilizadores para a gestão das cargas, limitando assim as ações que podem ser executadas nas mesmas mas assegurando a colaboração dos utilizadores. E garantir que a gestão de cargas não irá interferir de forma negativa nos processos que estas precisam de executar [14].

A vantagem que esta coordenação traz aos consumidores é que estes conseguem poupar no custo da energia, ao mesmo tempo que contribuem para a eficiência e bom funcionamento da rede, uma vez que os preços serão estabelecidos consoante as necessidades da mesma [10], isto é, os preços da eletricidade serão mais altos quando a rede se encontra mais sobrecarregada, ou com menos capacidade de produção, e será mais barata em horas de consumo reduzido. Os preços podem também ser estabelecidos de forma a promover o uso de energias renováveis, ou seja, em alturas em que a produção de energia de fontes renováveis, como eólicas e fotovoltaicas, é elevada os preços serão mais favoráveis, do que nas alturas em que a produção deste tipo de energia não é tão significativo [2].

Assim os preços da eletricidade servirão para fazer a coordenação entre a gestão da rede e os utilizadores, e proporcionarão a estes uma oportunidade de reduzir os seus custos com a eletricidade ao mesmo tempo que participam na gestão da rede. Gerando assim vantagens para consumidores ao mesmo tempo que beneficia a rede elétrica e até mesmo o ambiente.

2.3. Requisitos

Para implementar uma gestão de cargas ao longo de toda a rede, de forma a conseguir um controlo da procura eficiente, é necessário criar infraestruturas capazes de suportar todo o processo associado.

Conseguir que todos os consumidores possam contribuir de forma ativa no funcionamento da rede é uma tarefa complexa que requer meios adequados para a por em prática. Estes meios passam pela existência de dispositivos ou sistemas capazes de fazer a gestão das cargas em cada residência ou indústria, capacidade de recolha de informação para orientar a gestão a efetuar, meios de comunicação ao longo da rede para possibilitar a troca de informação entre todos os intervenientes e algoritmos de otimização e previsão para ajudar ao processo de decisão dos clientes e dos gestores das redes elétricas.

2.3.1. Sistemas de Gestão de Cargas

Os Sistemas de Gestão de Cargas (SGC) são importantes pois permitem que o controlo da procura seja possível de se por em prática, através da gestão das cargas associadas a estes e tornam todo o processo automático e prático para os utilizadores [2] [15]. A rede elétrica também beneficia com a existência destes sistemas, pois estes permitem descentralizar várias operações, reduzindo assim o esforço de processamento

que terá que ser realizado pela gestão da rede, ao mesmo tempo que criam algum nível de independência para realizar intervenções a nível local. Esta solução é preferível a ter todas as cargas ligadas à rede a enviar informação para um sistema de gestão central, visto que se este fosse o caso, seria necessário que o centro de processamento contivesse uma capacidade de processamento irrealista, e todos os canais de comunicação até este, teriam de permitir fluxos de informação muito maiores do que se existir algum grau de independência e autonomia. Para além destes fatores, é preciso considerar que no caso de a gestão ser feita de forma centralizada, o tempo que a informação leva a ser transmitida e processada é maior, e em caso de avaria do centro de processamento toda a gestão de cargas ficaria comprometida, ao passo que, se esta for feita com um certo grau de independência, apenas parte das funções ficariam comprometidas. Assim, os SGC servirão principalmente para fazer o interface entre a rede e as próprias cargas que cada sistema individual está encarregue de gerir [15] (Figura 2.1).

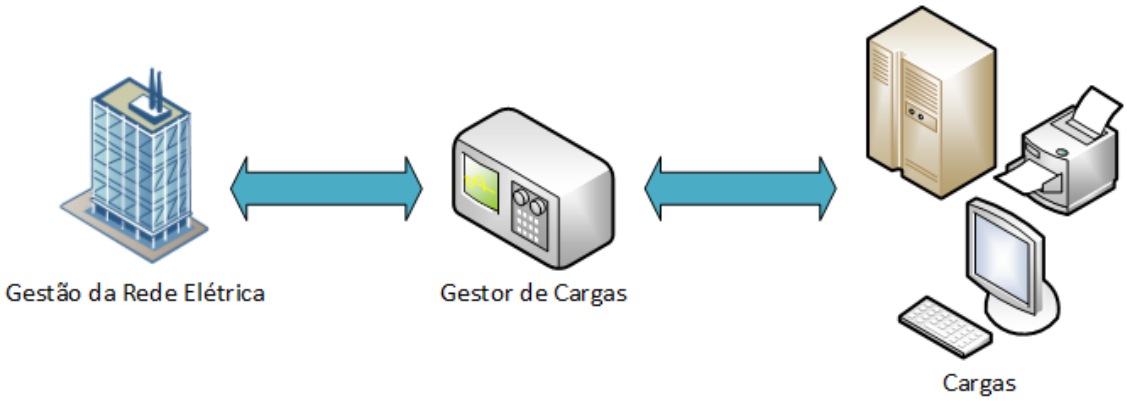


Figura 2.1 - Interações do gestor de cargas.

Os SGC serão responsáveis por receber informações provenientes da rede elétrica, como preços a serem praticados, condições da rede e necessidades desta. Do outro lado é também necessário recolher informações sobre o funcionamento e estado das cargas que estão a ser geridas, de forma a saber com antecedência em que condições estas se encontram e de que forma o seu atual funcionamento pode ser alterado. Para além destas informações, é necessário interagir com o utilizador, que é o proprietário das cargas, enquanto é importante ter algum controlo sobre as mesmas, para que, como foi dito anteriormente, se possam usar estas para benefício da rede, tendo em conta as preferências dos consumidores [14]. Assim sendo, uma das funções dos SGC será decidir como reagir às necessidades da rede enquanto respeita as preferências dos consumidores, ficando a cargo do sistema de gestão de cargas encontrar a melhor solução tendo em conta os requisitos de ambas as partes. Para que isto aconteça é necessário, para além da rede, o SGC seja capaz de interagir com os utilizadores e guardar as preferências dos mesmos

(Figura 2.2), de forma que cada pedido da rede seja comparado com essas preferências e verificar se pode ser aceite ou se as indicações do utilizador não permitem que este seja atendido.

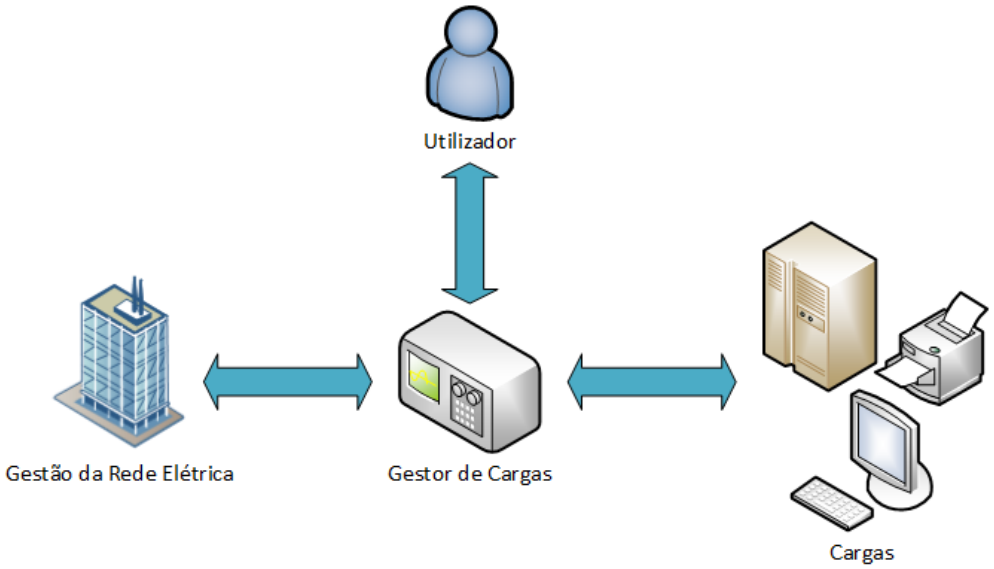


Figura 2.2 - Interações do gestor de cargas servindo de mediador para o utilizador.

Outros fatores que tornam os SGC importantes são o tempo de resposta e a disponibilidade, isto é, para que o controlo do consumo seja eficiente é necessário que a resposta às necessidades da rede elétrica seja rápida, pois um período de resposta demasiado longo pode significar numa redução das vantagens provenientes da atuação nas cargas ou até mesmo fazer com que esta solução deixe de ser vantajosa [6]. A disponibilidade é igualmente importante, uma vez que a qualquer hora pode ser necessário a rede intervir nas cargas, logo se estas não estiverem disponíveis através do sistema que as gere, a contribuição destas deixe de ser possível, o que também reduz a eficiência do controlo de consumo. Tudo isto torna evidente a necessidade de se ter um sistema automático dedicado, uma vez que seria impossível pôr em prática um controlo de consumo ao longo de toda a rede se a cada pedido da rede ou a qualquer alteração de preços fosse necessária a intervenção direta dos consumidores.

2.3.2. Sistemas de Gestão de Cargas Existentes no Mercado

Existem já disponíveis no mercado alguns SGC, embora estes podem não seguir exatamente todos os critérios aqui apresentados, devido às especificidades de cada produto e da visão que cada fabricante tem para a gestão de cargas a implementar pelos seus produtos e de que forma esta pode se distinguir da concorrência, no entanto pode ser facilmente constatado que muitos dos pontos mencionados anteriormente já estão

presentes nestas soluções. Os produtos apresentados servirão para demostrar casos concretos dos conceitos apresentados anteriormente e para demonstrar que tais sistemas são possíveis de ser concretizados e são cada vez mais alvo de interesse por parte de grandes companhias.

2.3.2.1. G Smart

O *G Smart* (Figura 2.3) é produzido pela empresa EFACEC e tem como objetivo controlar e monitorizar redes de baixa e média tensão. Este dispositivo é mais direcionado para a automatização de estações de transformação e automatização de alimentadores. No entanto, quando combinado com as suas funcionalidades de *Smart Metering* o *G Smart* pode ser também utilizado em aplicações para *Smart Grids*.

Figura 2.3 - *G Smart* [16].

Ao integrar múltiplas funções de automação com a capacidade de recolha de informação, gestão e vários standards de comunicação, este produto permite desempenhar funções como controlo de iluminação pública, gestão do consumo, carregamento de veículos elétricos e controlo de microgeração, sendo todas estas aplicações relevantes para as *Smart Grids*.

Para o controlo de iluminação o G Smart providencia um módulo de software opcional que permite, não só o controlo manual, mas também opções de escalonamento e gestão, através de uma tabela existente para este efeito em que o utilizador define os períodos em que pretende que a iluminação se encontre ligada. Um outro tipo de escalonamento é associar o desligar e ligar das luzes à hora do nascer e pôr-do-Sol, respetivamente. Existe ainda a opção de associar o escalonamento das luzes a medidores

de iluminação e ao consumo de energia. Estas configurações podem ser modificadas localmente pelo *web interface* ou remotamente por plataformas de gestão de iluminação pública como o *eLumem*.

Para implementar a gestão de consumo o *G Smart* consegue remover de funcionamento cargas de forma iterativa e seletiva de acordo com a importância das mesmas e fazendo o restauro do funcionamento quando não seja mais necessário que estas estejam paradas. Utilizando isto com um algoritmo que gere o consumo de energia, o *G Smart* consegue fazer o controlo de consumo ao monitoriza-lo este e mantendo-o dentro dos valores pretendidos com base nas capacidades da rede.

Uma outra funcionalidade presente é a capacidade de monitorização de estações de carregamento para veículos elétricos, podendo assim ser responsável pela interação destes com a rede. Para além de gerir a informação referente à operação o *G Smart* pode também implementar estratégias de carregamento inteligentes em que os ciclos de carregamento são coordenados de forma a minimizar o impacto na rede de distribuição de energia. Além disto, quando a função de V2G (*Vehicle-to-Grid*) for suportada pelos construtores este modo de operação também pode ser gerida pelo G Smart.

Também é possível o funcionamento como gestor de unidades de microgeração graças aos seus algoritmos de controlo inteligentes e à informação recolhida dos dispositivos inteligentes espalhados pela rede de baixa tensão. Isto permitirá que sejam encontradas soluções que satisfaçam da melhor forma os requisitos técnicos e económicos da rede, e que estas sejam comunicadas em tempo real às unidades de microgeração e de armazenamento de energia, para que a reposta seja o mais rápido possível.

Para dar suporte a todas estas funcionalidades o *G Smart* possui um *web server*, portas de I/O, capacidade de armazenamento de informação, comunicações, capacidade de monitorização de auto monitorização, monitorização e medição da qualidade de energia elétrica a nível local e diversas API (*Aplicattion Programming Interface*) de software que permitem que a as aplicações mais avançadas sejam implementadas pelo o utilizador [16].

2.3.2.2. SACE Emax 2 with Ekip Power Controller

O *Sace Emax 2* (Figura 2.4) é um disjuntor produzido pela empresa ABB que quando equipado com a função *Ekip Power Controller* que já está integrada na unidade eletrónica que é usada para o funcionamento normal do disjuntor para proteção contra sobre correntes, permite que este seja capaz de realizar o controlo de cargas sem que seja

necessário sistemas de controlo complexos ou software dedicado adicional para este efeito.

Figura 2.4 – Disjuntor *SACE Emax 2* [17].

Esta função é baseada num algoritmo de cálculo que permite que a carga seja controlada através de comandos remotos dos circuitos de comutação relevantes ou circuitos de controlo, de acordo com as prioridades definidas pelo utilizador, prioridades estas que são baseadas nos seus requisitos e nos tipos de cargas.

Os comandos a serem enviados para os componentes a serem controlados podem ser transmitidos de duas formas: através de uma comunicação com fios ou através da utilização de uma comunicação dedicada para este efeito.

O objetivo deste sistema é limitar o consumo médio de energia num determinado período de tempo. Este objetivo é atingido ao desconectar simultaneamente algumas cargas consideradas menos prioritárias pelo utilizador quando o algoritmo o considera necessário. As cargas só voltam a ser conectadas quando o algoritmo verificar que já não existe a necessidade destas estarem desligadas. As prioridades das cargas são definidas pelo utilizador de acordo com as suas preferências ou necessidades. De forma semelhante ao que faz às cargas, este sistema também pode fazer o controlo de geradores de reserva. O algoritmo tem também como objetivo a otimização do número de cargas que se encontram ligadas, de forma a minimizar o impacto das alterações efetuadas por este.

O algoritmo consiste em quatro passos fundamentais para a obtenção destes resultados, sendo o primeiro passo a medição da potência que passa pelo disjuntor. O segundo passo é a sincronização, que com base no relógio interno define os intervalos a que a potência média é medida, durante o tempo de referência definido. Em intervalos regulares é dado início ao passo seguinte que é a avaliação, onde a sincronização também pode ser feita por um sinal externo. A avaliação é baseada na energia medida e no tempo passado desde o início do período de referência, o algoritmo avalia se o consumo é demasiado alto, se está dentro dos valores definidos ou se está demasiado abaixo destes.

Com base nesta avaliação o número de cargas ligadas vai ser reduzido, mantido ou aumentado, respetivamente. O resultado deste passo vai servir como base à fase de gestão de cargas que vai decidir que cargas devem ser desligadas ou ligadas de acordo com a sua prioridade, no entanto este passo tem em conta o tempo em que cada carga não pode alterar o seu estado devido ao risco de causar dano [17].

2.3.2.3. Wiser Home Management

O *Wiser Home Management* (Figura 2.5) é um sistema de gestão de consumo produzido pela Schneider Electric. Como o nome indica é direcionado para o mercado doméstico e apresenta várias soluções *plug-and-play*. Este sistema permite ao utilizador ver, monitorizar e gerir o uso de energia e configurar as suas preferências para a gestão de consumo a implementar. Este sistema é composto por vários componentes que em conjunto implementam a gestão de consumo nas habitações.

Figura 2.5 – Sistema de gestão de consumo *Wiser Home Management*, figura adaptada de [18].

Para interface com o utilizador está disponível um *site* e uma aplicação que permite ao utilizador visualizar e controlar a utilização de energia. Um coordenador faz o *upload* da informação para a *cloud* dos equipamentos existentes na habitação, de forma que este esteja disponível através do *website* e da aplicação. Termostatos inteligentes fazem ajustes automáticos de acordo com as preferências do utilizador, tomadas inteligentes desempenham uma função semelhante nos eletrodomésticos da casa. Existem também soluções para o carregamento de veículos elétricos e inversores para fazer a integração dos painéis solares. Para a recolha de informação sobre o consumo de energia existe um medidor responsável por monitorizar todo o consumo de energia da casa e enviar os dados para a aplicação e portal *web*. Para controlo e visualização de informação a nível local existe um terminal dedicado. Por fim, existe um controlador de cargas que fica responsável por monitorizar, controlar e escalonar as cargas maiores. Este componente consiste num disjuntor controlável com comunicações por *ZigBee*.

Para além das funcionalidades referidas o *Wiser Home Management* pode ainda ser alterado para também fazer a gestão e monitorização dos sistemas de segurança e trancas das portas [18].

2.3.3. Infraestruturas de Comunicação

Os SGC em si só não são suficientes para implementar um programa de controlo de consumo. Como já foi mencionado anteriormente, o sistema de gestão de cargas irá fazer a comunicação com a rede, cargas e utilizadores. Desta forma, é necessário que sejam criados os meios para transmitir informações ao longo da rede, dotar os equipamentos intervenientes com formas de comunicação (SGC e cargas) e criar os interfaces para os utilizadores poderem definir as suas preferências e interagir com os restantes intervenientes.

A estrutura de comunicação da rede vai ser responsável por grande parte do fluxo da informação, sendo o resto da informação trocada localmente, e é através desta que vão ser comunicados os dados provenientes dos diversos medidores para a gestão da rede. Por sua vez, a rede utiliza esta estrutura para comunicar os preços a ser praticados e as necessidades da rede aos sistemas que fazem a gestão das cargas. O *feedback* resultante destas informações será transmitido pelo mesmo meio (Figura 2.6).

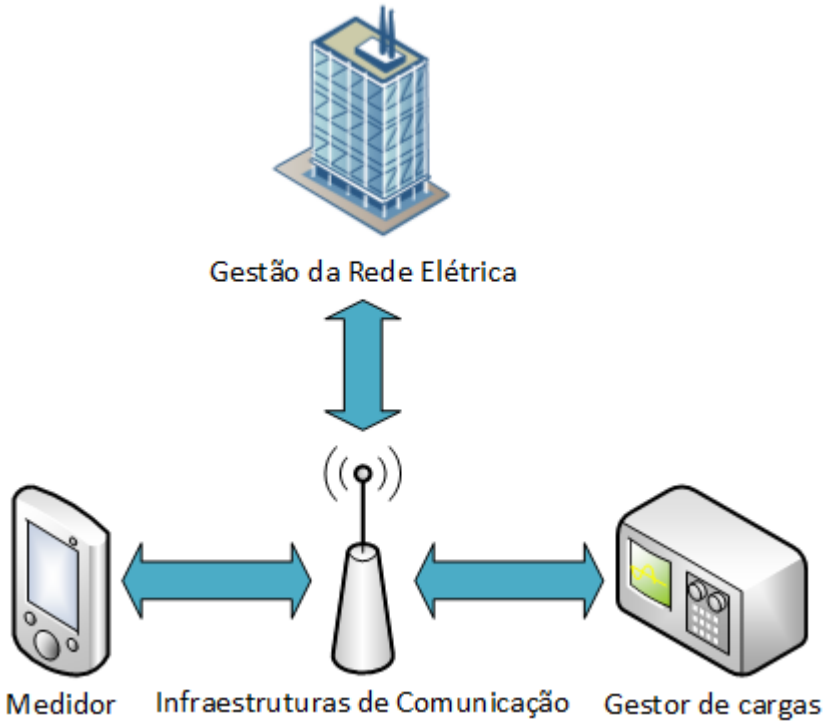


Figura 2.6 - Infraestruturas de comunicação.

Os utilizadores realizarão a sua interação com os restantes através dos sistemas de gestão que deverão estar equipados com meios para tal, como por exemplo, interface local, ou interface remoto. Pode ser ainda utilizada uma combinação destes dois tipos de interface para oferecer ainda mais possibilidades aos utilizadores.

2.3.4. Medidores Avançados

Para efetuar a recolha de informação ao longo da rede elétrica serão necessários medidores para monitorizar o estado da mesma. No entanto, estes medidores precisam de ser diferentes do que hoje em dia se encontram na maioria das habitações e indústrias. Os medidores terão de recolher informação que vai para além do consumo de cada um dos clientes.

Outra característica que deverá ser incluída nos medidores é a capacidade de comunicação com a rede, isto é, a possibilidade de receber e enviar informação para esta. Por fim, e não menos importante, estes medidores devem incorporar a capacidade de enviar alertas para a rede se estes registarem algum problema, mesmo sem que a rede tenha previamente requerido algum tipo de informação (Figura 2.7).

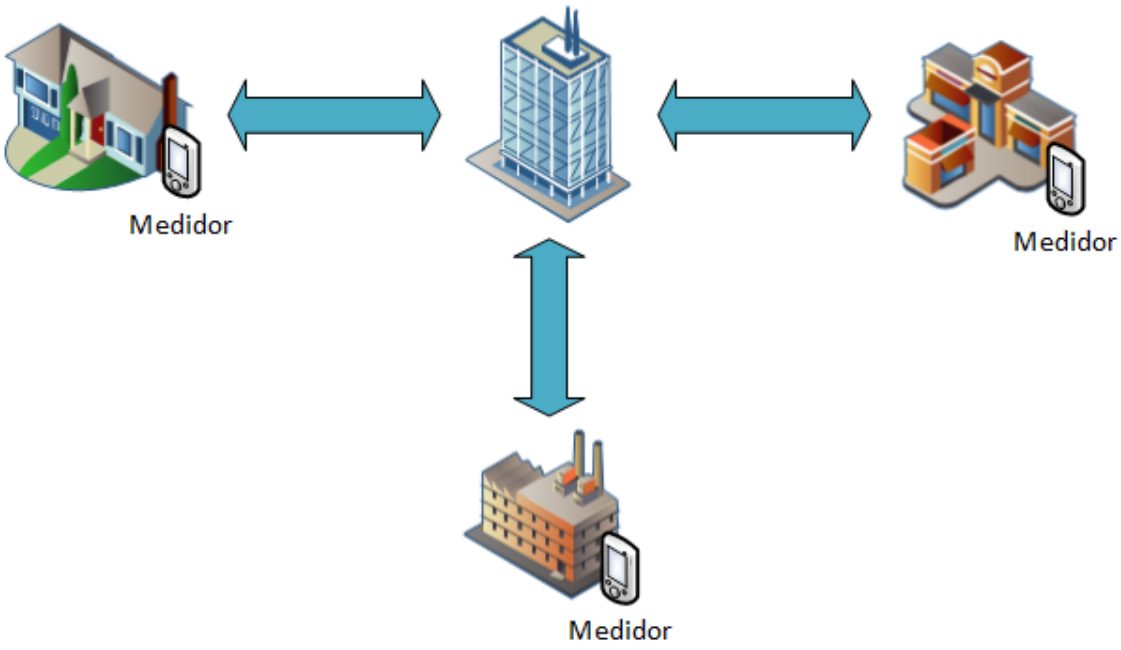


Figura 2.7 - Distribuição dos medidores ao longo da rede.

A existência destes dispositivos pode também beneficiar os SGC, visto que assim é possível a troca de informação entre os mesmos. Através desta interação é possível fazer a gestão das cargas de uma forma mais independente da rede, pois parte da informação necessária seria adquirida localmente pelos dispositivos orientados para esse efeito (Figura 2.8). Desta forma, o fluxo de informação ao longo da rede será mais reduzido uma vez que parte da informação necessária seria adquirida e transmitida localmente entre estes dois tipos de dispositivos.

Este tipo de intervenção local será também mais rápida uma vez que o número de intervenientes será menor, assim como, a distância percorrida pela informação e o tempo de espera pela decisão. Todavia, é preciso coordenar estas intervenções locais com as necessidades da rede, para evitar que esta independência não se torne prejudicial.

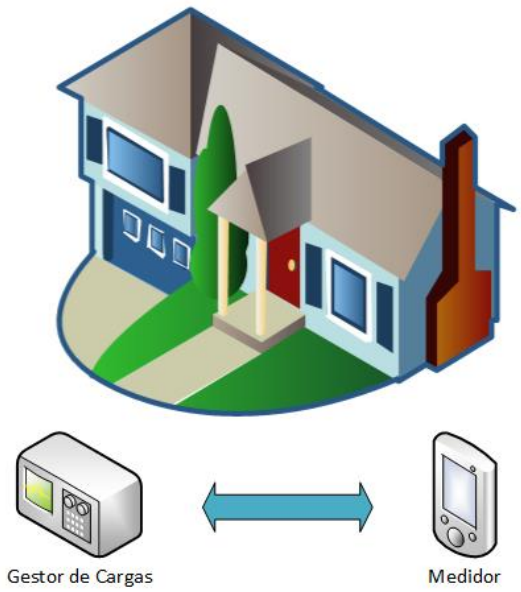


Figura 2.8 - Interação entre medidor e gestor de cargas a nível local.

2.3.5. Algoritmos de Otimização e Previsão

Para além dos medidores, os SGC podem também beneficiar de algoritmos capazes de prever com alguma antecedência o comportamento de diversos fatores da rede, como por exemplo a geração de energia, consumo e o preço a pagar pela energia consumida. Além dos algoritmos de previsão, também podem ser utilizados algoritmos de otimização de forma a ajudar o utilizador do sistema a encontrar a melhor solução e assim reduzir o mais possível a sua conta de eletricidade sem comprometer o seu conforto ou as suas necessidades [19].

Os algoritmos de previsão permitirão à gestão da rede e aos utilizadores planificar com antecedência as suas ações. Para a gestão da rede será importante prever o consumo para saber quanta produção será necessária, se a produção através de energias renováveis será suficiente ou se será necessário mobilizar fontes de geração adicionais [20]. Para os utilizadores será importante saber com antecedência os preços a serem praticados no fornecimento de eletricidade para planificarem o uso das suas cargas.

Feita a previsão dos diversos fatores críticos da rede, serão executados os algoritmos de otimização para encontrar as soluções mais vantajosas das condições e recursos existentes. Tendo em conta o elevado número de fatores a ter em conta, esta tarefa é demasiada complexa para ser efetuada sem o auxílio de algoritmos desenvolvidos para este efeito.

2.4. *Smart Grids*

O controlo de consumo apesar de ser uma inovação na forma de gestão e funcionamento das redes elétricas, principalmente no que diz respeito ao papel que os consumidores têm no funcionamento destas redes, está longe de ser a única mudança que se prevê que irá acontecer nas redes elétricas atuais.

2.4.1. Caraterísticas das *Smart Grids*

As *Smart Grids* irão apresentar características diferentes das redes de distribuição de energia elétrica atuais, para isto estas irão tirar proveito de avanços tecnológicos em diversas áreas, tais como, a produção de energia, transporte, monitorização, comunicação e controlo [21]. Todas estas áreas serão utilizadas em conjunto para poder criar um novo tipo de rede elétrica capaz de responder de forma mais eficiente aos novos desafios que surgem nesta área.

Uma das alterações das *Smart Grids* em relação às redes atuais é que estas serão mais dinâmicas, isto é, terão a capacidade de responder a eventos de forma mais rápida e autónoma. Logo, estas serão capazes de detetar e responder a eventuais falhas de forma muito mais rápida. Este aspeto poderá ser mesmo levado um passo mais à frente, ou seja, fazer a previsão de possíveis falhas [22]. Nas redes atuais as falhas são de difícil deteção, o que leva a um tempo de resposta mais longo, e a resposta a estas implica obrigatoriamente intervenção humana. Para além disto qualquer tipo de previsão de falhas é praticamente impossível [23].

O que irá permitir que as *Smart Grids* tenham a capacidade de previsão de falhas são as infraestruturas de monitorização que estarão integradas nestas. Estas infraestruturas para além de irem permitir a implementação do controlo de consumo, como foi mencionado em pontos anteriores, também permitirão a monitorização do estado de funcionamento da rede e até do estado dos seus componentes, dando assim a capacidade de prever quando estes irão falhar para que medidas possam ser tomadas antes que isso aconteça [22].

A informação que é recolhida servirá para a criação de modelos que possam representar e prever o comportamento da rede e dos seus componentes. Estes modelos servirão para a rede ajustar o seu funcionamento com base nas previsões mas também para melhor perceber de que forma a rede vai ter de evoluir para responder ao crescente consumo de energia elétrica [24].

Uma outra característica das *Smart Grids*, que apenas agora começa a ser observado nas redes elétricas tradicionais, é o fluxo bidirecional de energia, isto é, os consumidores deixam de ser apenas consumidores mas passam também a produzir a sua própria energia que irá servir para satisfazer as suas próprias necessidades, mas também pode ser vendida à rede. Esta característica já está presente nas redes atuais mas será muito mais significante nas *Smart Grids* do futuro, pois irá fazer com que as redes elétricas deixem de ser tão controladas pelos produtores de energia, e se tornem mais interativas para o consumidor, uma vez que este vai poder decidir o que fazer com a energia que produz e em que condições é mais vantajoso vender ou não a energia produzida [25].

Este fator aliado à crescente expansão das energias renováveis vai fazer com que a produção de energia se torne cada vez mais descentralizada, ao contrário do que é feito atualmente.

As energias renováveis, apesar de serem fontes viáveis de energia têm problemas associados, pois como foi anteriormente referido, estas não têm uma produção de energia constante e esta também não pode ser prevista a longo prazo, logo a flexibilidade que as *Smart Grids* vêm trazer vão ajudar a que este tipo de energias possam crescer ainda mais sem afetar negativamente a rede elétrica, bem como a qualidade e disponibilidade do serviço prestado por esta [26].

Um outro fator que irá marcar de forma positiva as *Smart Grids*, será o avanço nas tecnologias de armazenamento de energia e o aumento da proliferação deste tipo de tecnologias, uma vez que prevê-se um aumento do número de carros elétricos e estes poderão servir como armazenadores de energia quando não se encontram em circulação. Isto permitirá não só à rede mas também aos consumidores/produtores armazenar a energia nas alturas em que a produção esteja em alta e utilizar esta quando a produção não é tão abundante, ou então para satisfazer os picos de consumo [19].

Todos estes fatores farão com que o mercado da energia, sob o paradigma das *Smart Grids*, se torne muito diferente do atual, uma vez que deixarão de haver apenas os grandes produtores, e passarão a haver muitos pequenos produtores e produtores domésticos. Isto irá fazer com que a competitividade dos preços aumente, o que por sua vez fará com que novos tarifários e ofertas sejam criados para cativar mais consumidores e tirar melhor proveito destas novas características das *Smart Grids* [27].

2.4.2. Vantagens das *Smart Grids*

As novas características associadas às *Smart Grids* trazem consigo vantagens a vários níveis, entre os quais se destacam o económico e o ambiental. As vantagens no foro económico já foram mencionadas no ponto anterior com a referência ao surgimento de mais concorrência nos preços e de novos tarifários e ofertas, mas estas são apenas parte dos benefícios económicos que estas vêm trazer.

O aumento de fiabilidade, disponibilidade e eficiência que as *Smart Grids* oferecem vão também trazer outros ganhos económicos, para além dos associados à compra e venda da energia, podendo os benefícios económicos associados a cada um destes fatores ser analisados individualmente.

O aumento da fiabilidade irá fazer com que existam menos avarias na rede elétrica o que irá reduzir os custos de manutenção, reparação e substituição dos componentes desta. Relacionado com este ponto está a disponibilidade da rede, ou seja, os períodos de tempo em que a rede está operacional e com capacidade de responder à procura de energia elétrica.

As falhas na disponibilidade da rede elétrica têm custos diretos e indiretos. Os custos diretos são aqueles que estão associados ao tempo em que a energia não está a ser fornecida, logo os produtores e distribuidores estão a perder dinheiro pois não conseguem vender o seu produto. Por outro lado, os custos indiretos estão associados aos consumidores que não podem desenvolver as suas atividades, e numa economia altamente dependente da energia elétrica uma breve interrupção no fornecimento desta pode significar prejuízos elevados, sendo por isso este o fator mais significativo [28].

A poupança relacionada à eficiência da rede traduz-se tanto a nível da redução das perdas de energia durante o seu transporte entre o local de produção e os consumidores, como também na redução da necessidade, ou adiamento, da expansão da capacidade da rede elétrica [29], o que evita ou adia o investimento em novas infraestruturas de produção e transporte necessárias para satisfazer um constante aumento na procura de energia elétrica.

Para além dos benefícios económicos que as *Smart Grids* oferecem, estas também têm benefícios a nível ambiental, uma vez que, tal como já foi referido anteriormente, as *Smart Grids* irão permitir uma maior expansão das energias renováveis, logo haverá uma diminuição da dependência de combustíveis fósseis para a produção de energia elétrica, o que irá reduzir as emissões de CO2. A possibilidade de criar programas, ou tarifários que promovam o uso de energia quando a produção através das energias renováveis está

em alta, é também um fator que irá aumentar ainda mais os benefícios para o ambiente [26].

2.4.3. Desafios à Implementação das *Smart Grids*

Como pode ser observado, a mudança entre o panorama atual da produção, transporte e distribuição da energia elétrica para o panorama das *Smart Grids* irá ser radical, e várias alterações terão de ser feitas para que estas possam funcionar de forma a trazer as vantagens pretendidas.

Um dos primeiros desafios à implementação das *Smart Grids* será provar que o custo da sua implementação pode ser recuperado, e que os custos de manutenção a estas associadas serão compensados pelas vantagens que estas virão trazer durante o seu ciclo de funcionamento [30]

Como foi visto no ponto anterior, as vantagens económicas e ambientais das *Smart Grids* são grandes, no entanto o custo associado à investigação, desenvolvimento, instalação, manutenção e funcionamento das tecnologias necessárias para a operação destas também é elevado, logo é necessário a realização de estudos que comprovem que esta solução será de facto economicamente rentável como atualmente se prevê [30].

Para além dos estudos de rentabilidade económica, é também necessário realizar estudos sobre a participação e comportamento dos utilizadores nas *Smart Grids* [31]. Como ficou claro nos pontos anteriores o controlo do consumo é um ponto fulcral no futuro das redes de energia elétrica, logo será importante prever o nível de participação dos consumidores nestes programas e até que ponto eles estão dispostos a contribuir para a necessidades da rede e quais os incentivos necessários para motivar esta participação.

Relacionado também com isto está a necessidade de informar e esclarecer os utilizadores das vantagens que as *Smart Grids* e os programas de controlo de consumo associados a estas vêm trazer, para eles e para a rede, de forma que estes percebam que os investimentos (em medidores, dispositivos de gestão e comunicação) e alterações no funcionamento das suas cargas (e.g., carros elétricos a devolverem energia elétrica à rede em vez de apenas a consumirem durante o carregamento das baterias) serão compensados pelas vantagens que a participação nestes programas trarão [32].

Para além dos desafios sociais e económicos, a implementação das *Smart Grids* terá obviamente de ultrapassar também desafios de cariz mais técnico. Estes desafios passarão pela recolha e processamento da informação obtida sobre os vários fatores da rede de forma que se possa atuar em tempo útil sobre esta [23] [22]. Para isso será necessário o desenvolvimento de novas tecnologias que sejam capazes não só de fazer a

leitura destes dados, mas também de os processar e comunicar de forma segura para os responsáveis pela gestão da rede.

O desafio seguinte será dotar todos os intervenientes da rede de um maior grau de automatização e capacidade de controlo remoto, para que seja reduzido o tempo que a rede necessita para se adaptar a novas circunstâncias. É neste área que se encontram os SGC ao conferirem esta capacidade às cargas que se encontram do lado do consumidor. No entanto, será também preciso criar soluções tecnológicas semelhantes para outros componentes que fazem parte do processo de produção, transporte e distribuição da energia.

Toda esta troca de informação, instruções de comando, controlo de cargas e componentes da rede é considerado informação extremamente sensível e crítica para o funcionamento dos sistemas de energia do futuro. Qualquer tipo de intrusão no sistema ou captura de informação pode levar a consequências catastróficas, o que torna óbvio que outro dos desafios à implementação das *Smart Grids*, é a capacidade de dotar todos os dispositivos que fazem parte destas de soluções de segurança informática fortes o suficiente para assegurar um sector da sociedade tão importante como é o setor da energia [33].

Todas estas alterações a nível da produção levará a que apareçam novos desafios no que toca a manter a qualidade da energia ao longo da rede, como por exemplo níveis de tensão e frequência constantes. Manter estes parâmetros torna-se mais complexo num cenário em que a produção de energia é distribuída, uma vez que há mais intervenientes em jogo, e até mesmo a limitação das correntes de defeito se torna mais complexa [26].

2.5. Conclusão

Ao longo deste capítulo foi abordado o conceito de gestão de consumo que será utilizado para melhorar o desempenho das redes elétricas e conferindo a estas uma maior flexibilidade, ao mesmo tempo que inclui os clientes da rede no esforço para melhorar o seu funcionamento, sendo dentro desta gestão de cargas que os Sistemas de Gestão de Cargas (SGC) desempenham a sua função como intermediário entre todas as partes envolvidas.

Foram apresentadas as diversas vantagens que a gestão de cargas vem trazer para a rede e utilizadores: Para a rede as vantagens são provenientes da maior flexibilização do consumo de energia e da possibilidade de coordenação do funcionamento de cargas, fatores que permitirão resolver alguns problemas que as redes atuais enfrentam e criar

novas oportunidades para o melhoramento da rede elétrica, bem como possibilitar uma maior penetração das energias renováveis, uma vez que possibilita contornar alguns dos problemas associados a estas.

Os utilizadores poderão beneficiar da gestão de consumo através dos incentivos oferecidos pela rede como contrapartida pela participação nos programas de gestão de consumo. Uma outra forma de os utilizadores beneficiarem com isto é aproveitando os diferentes tarifários oferecidos, mudando o funcionamento das suas cargas para horários em que a energia não é tão cara.

Os requisitos para a implementação da gestão de cargas também foram abordadas, uma vez que é nesta categoria que se enquadram os sistemas de gestão de cargas, que são o tema desta dissertação.

As características e as funções dos SGC são abordadas de forma a compreender como estes contribuem para a gestão de consumo. Foram também apresentados exemplos já implementados de SGC de forma a perceber o que atualmente já é feito nesta área. São abordados também outros requisitos para a gestão de consumo, como as infraestruturas de comunicação necessárias para a circulação de informação que irá alimentar a gestão de consumo, os medidores avançados que vão ser responsáveis por recolher essa informação e os algoritmos de previsão e otimização, quer a nível da rede quer a nível individual, de forma a tirar melhor partido da informação recolhida.

Por fim são abordadas as *Smart Grids*, que são o tipo de rede onde se prevê que a gestão de cargas será incorporada. As características deste tipo de rede são abordadas de forma a perceber que outras contribuições vão advir destas redes, para além da gestão de consumo, seguindo para as vantagens proporcionadas pelas *Smart Grids*, para além das que foram mencionadas na gestão de consumo. Por fim, são abordados os desafios à implementação das *Smart Grids*, de forma a perceber o percurso que ainda falta percorrer até que estas sejam uma realidade, e que contribuição o desenvolvimento de sistemas de gestão de cargas vem trazer para este esforço.

CAPÍTULO 3

Projeto do Sistema de Gestão de Cargas para *Smart Grids*

3.1. Introdução

Este capítulo aborda o projeto do sistema de gestão de cargas, onde são feitas as descrições das diversas partes deste, bem como os componentes e ferramentas que foram utilizados na sua elaboração. As funcionalidades gerais pretendidas para o sistema também serão abordadas neste capítulo.

A descrição das partes do sistema é realizada através de uma abordagem dos componentes e as ferramentas com que estas foram elaboradas de forma a perceber o porquê da escolha destas e os seus impactos no sistema em geral. Terminada a descrição dos componentes é iniciada a exposição dos diferentes módulos e sub-módulos que constituem este sistema.

Os módulos são especificados em termos de funcionalidades desempenhadas e os processos que são efetuados para que estas sejam realizadas, sendo também descritos de forma a dar uma visão mais completa do sistema e da forma como este funciona.

3.2. Descrição das Funcionalidades Pretendidas

Através do capítulo anterior é possível perceber os requisitos básicos necessários para que um Sistema de Gestão de Cargas (SGC) consiga ser uma mais-valia para as *Smart Grids* , sendo estes requisitos a linha orientadora do projeto do SGC. Como tal o SGC será capaz de comunicar com os utilizadores e com a rede de forma a poder gerir a carga a este associada segundo as necessidades destes intervenientes. Deve também conseguir armazenar informação referente às preferências dos utilizadores e do estado de funcionamento da carga a gerir. Por fim, deve ser capaz de comunicar com a carga de forma a efetuar a sua gestão.

A capacidade de gestão de mais do que uma carga também será importante, por isso são desenvolvidas soluções para acomodar vários tipos de cargas. Nomeadamente, tipos de bancos de condensadores, filtros ativos e carros elétricos.

Os bancos de condensadores foram selecionados por serem um tipo de carga bastante comum na indústria com potencial para regulação do fator de potência e tensão na rede, ambos fatores de interesse a ter em conta na gestão da rede.

Os filtros ativos foram escolhidos pelas razões anteriormente apresentadas nos bancos de condensadores e também devido à sua capacidade de melhorar a qualidade da energia da rede, fator cada vez mais importante.

Por sua vez os carros elétricos foram escolhidos porque, quando equipados com carregadores inteligentes, podem também fazer a correção do fator de potência ao consumir reativos o que também pode ajudar a fazer o ajuste da tensão. Para além disto estes também podem fornecer a energia armazenada à rede, como foi mencionado anteriormente, e um fator muito importante para a gestão de consumo.

Tendo em conta as características das cargas escolhidas e apesar de se fazer um projeto para o controlo de cada tipo de carga, o principal foco será a gestão do carro elétrico, uma vez que a contribuição que este poderá fazer para o esforço de controlo do consumo é maior.

3.3. Descrição do Sistema

O sistema a implementar usará para interface com o utilizador uma aplicação móvel que permitirá a este controlar o SGC à distância. Esta aplicação estará ligada a um servidor que por sua vez estará ligado ao SGC, de forma a comunicar a este as informações e comandos recebidos do utilizador e da rede. Após a receção destas informações e comandos por parte do SGC este irá proceder com a ação de gestão mais adequada com base nas informações armazenadas e comunicar a ação que a carga deve efetuar a seguir.

Na Figura 3.1 é possível verificar o esquema de interações entre os diversos intervenientes, onde se é capaz de observar que o utilizador interage com a aplicação, que por sua vez serve de intermediária entre este e o servidor, ao passo que as interações com a rede são feitas diretamente entre esta e o servidor. Este por sua vez comunica também com o SGC. Por fim o SGC comunica com as cargas.

Figura 3.1 - Sistema a Implementar.

O SGC deve conseguir de forma independente a gestão das cargas na ausência de comandos, isto quer dizer que caso não existam novos comandos este deve executar as ações previamente definidas, mesmo na eventualidade de a comunicação com o servidor e a aplicação estiver indisponível. Esta ação independente passa essencialmente por fazer cumprir os horários de funcionamento da carga que já estão definidos.

3.4. Aplicação de Interface com o Utilizador

Tal como já foi mencionado anteriormente, foi escolhida uma aplicação móvel para fazer a interface com o utilizador, pois este tipo de aplicações têm-se tornado cada vez mais comuns. Uma vez que alcançam um elevado número de pessoas, graças à expansão dos *smartphones* e *tablets*, e visto que nos tempos atuais a grande maioria da população possuí pelo menos um, foram fatores decisivos na implementação de uma aplicação móvel para fazer a interface com o utilizador. Dos diversos sistemas operativos que hoje em dia correm nos dispositivos móveis (*smartphones*, *tablets*, etc) destaca-se o Android pela sua grande taxa de penetração no mercado, tendo sido esta a principal razão para a aplicação de interface ter sido desenvolvida para correr neste sistema operativo.

3.4.1. Sistema Operativo Android

O sistema operativo Android é um sistema operativo *open source* desenhado para dispositivos móveis. Apoiado pela Google e propriedade da Open Handset Alliance, é uma plataforma aberta que separa o hardware do software que corre nestes, o que permite que um número muito maior de aparelhos corra as mesmas aplicações criando um ecossistema mais rico para desenvolvedores e consumidores.

Para desenvolvedores, o Android providencia todas as ferramentas necessárias para desenvolver aplicações de forma rápida e fácil, algo que também contribuiu para a escolha do mesmo.

Para os utilizadores, o Android funciona de forma simples e oferece a estes a possibilidade de personalizarem a sua experiência substancialmente. Do ponto de vista dos fabricantes, este sistema operativo é vantajoso visto ser uma solução completa para correr nos seus dispositivos. Para além de alguns *drivers* específicos para o hardware, o Android providencia tudo o resto para que os dispositivos possam funcionar [34]. Ao longo dos anos este sistema operativo foi apresentando várias versões, que podem ser vistas na Tabela 3.1.

A versão mais atual, o Android Lollipop, e o nível de API (*Application Programming Interface*) correspondente, foi o escolhido para o desenvolvimento da aplicação. A escolha desta versão deveu-se ao facto de que ao dar suporte para versões mais antigas deste sistema operativo, as opções de componentes que podem ser utilizados são mais limitadas, e o aspeto de alguns deles já se encontra ultrapassado aos olhos dos utilizadores. Devido a estas razões optou-se por incluir componentes mais atuais de forma a conseguir um visual mais contemporâneo para a aplicação, em vez de providenciar suporte a versões mais antigas do Android.

Tabela 3.1 - Lista das versões do sistema operativo Android.

Versão	**Nome de Código**	**Data Lançamento**	**Nível de API**	**Logótipo**
5.0	Lollipop	Novembro de 2014	Nível de API 21 a 22	
4.4	KitKat	Outubro de 2013	Nível de API 19 a 20	
4.1/4.2/4.3	Jelly Bean	Julho de 2012	Nível de API 16 a 18	
4.0	Ice Cream Sandwich	Dezembro de 2011	Nível de API 14	
3.x	Honeycomb	Fevereiro de 2011	Nível de API 11 a 13	
2.3	Gingerbread	Dezembro de 2010	Nível de API 9	
2.2	Froyo	Maio de 2010	Nível de API 8	
2.0/2.1	Eclair	Outubro de 2009	Nível de API 5 a 7	
1.6	Donut	Setembro de 2009	Nível de API 4	
1.5	Cupcake	Abril de 2009	Nível de API 3	

3.4.2. Android Studio

Para desenvolver a aplicação para este sistema operativo existem duas ferramentas que se destacam, o Android Studio e o Eclipse com o ADT (*Android Development Tools*) um *plugin* que estende as capacidades deste IDE (*Integrated Development Environment*)

para ser possível criar aplicações Android neste [35]. O Android Studio foi selecionado por ser um IDE especificamente desenvolvido para a criação de aplicações para Android.

O Android Studio é o IDE oficial para o Android, baseado no IDEA da InelliJ. O Android Studio tem ganho mais espaço ao Eclipse com o ADT que até agora era também utilizado para desenvolver aplicações em Android [36].

A programação neste IDE é feita em Java, logo a utilização deste requer que exista uma versão do Java previamente instalada. O Android Studio instala automaticamente a versão mais recente do Android SDK (*Software Development Kit*), no entanto também é importante saber alterar e configurar este para se adaptar a especificidades de alguns projetos [37].

Para além das funcionalidades comuns de um IDE o Android Studio apresenta ainda:

- Suporte integrado para a plataforma Google Cloud;
- Possibilidade de Simular as aplicações em máquinas virtuais ou em dispositivos reais;
- Gestor para AVD (*Android Virtual Devices*);
- Gestor para o SDK.

3.4.3. Linguagem de Programação Java

Como foi mencionado no ponto anterior o Android Studio usa como linguagem de programação o Java, logo também será feita uma descrição desta linguagem de programação uma vez que esta será parte integrante do projeto.

As origens do Java remontam a 1990 na Sun Microsystems, empresa que foi comprada pela Oracle, que é a atual detentora desta linguagem de programação. O Java é considerado uma linguagem interpretada e compilada ao mesmo tempo, pois o código escrito em Java é compilado para instruções binárias que vão ser executadas pela máquina virtual. Logo o Java é uma plataforma em que se divide a máquina virtual do ambiente de execução. A máquina virtual é um processador baseado em software que apresenta um conjunto de instruções designado normalmente por Java Virtual Machine.

O ambiente de execução consiste em bibliotecas para correr os programas e interagir com o sistema operativo subjacente ou com a plataforma nativa. A unidade fundamental do código em java é a classe, uma vez que se trata de uma linguagem de programação orientada a objetos [38].

As principais características desta linguagem de programação são a simplicidade uma vez que oferece funcionalidades como a libertação automática de memória, enquanto os programas ainda estão a correr, e ser independente da plataforma, devido a ser uma mistura entre uma linguagem interpretada e compilada, o que permite executar programas em hardwares diferentes tornando-a bastante portável. Uma outra vantagem já foi referida anteriormente que é ser uma linguagem de programação orientada a objetos.

É uma linguagem segura devido às restrições que impõe em termos de acesso a recursos e a uma supervisão cuidada da alocação de memória. Apresenta também funcionalidades de programação paralela. E por fim é uma linguagem dinâmica que permite a adição de bibliotecas dinâmicas para serem adicionadas na altura de correr os programas [39].

3.4.4. Especificação da Aplicação

A aplicação terá como objetivo fornecer ao utilizador um meio para interagir com o SGC, embora esta interação não será feita diretamente mas sim através de um servidor. A aplicação deve então fornecer ao utilizador os meios para este selecionar e introduzir as configurações pretendidas para a carga, no entanto antes que um utilizador possa controlar uma carga este tem de passar por um processo de autenticação para verificar se este deve ter ou não acesso à mesma.

Para além de um ecrã inicial a aplicação possuí um ecrã de *login* para permitir a um possível utilizador introduzir as suas credenciais de acesso. Por sua vez estas vão ser enviadas ao servidor para verificação. Caso a validade seja verificada irá ser dado ao utilizador acesso ao ecrã de seleção de cargas a gerir, caso contrário o acesso é negado e indicado o porquê de o ter sido. Este processo pode ser visualizado no diagrama de sequência presente na Figura 3.2.

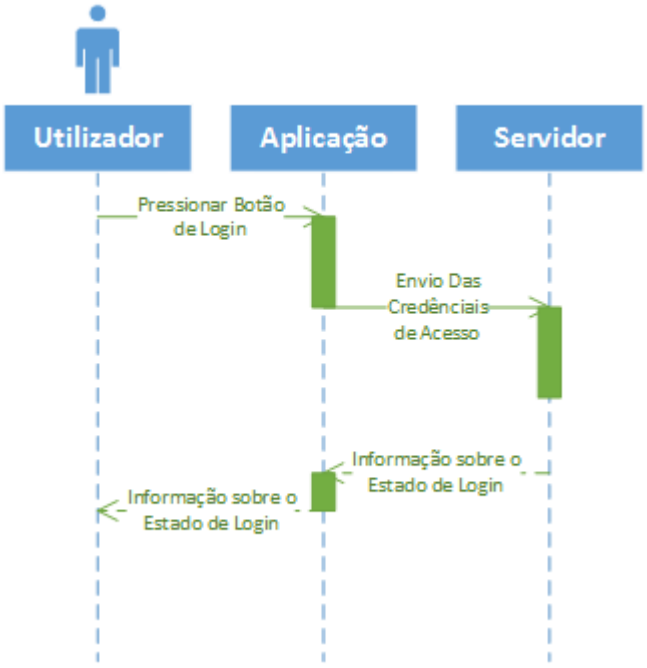


Figura 3.2 - Diagrama de sequência do processo de *login*.

O ecrã de seleção de cargas irá apresentar os tipos de cargas disponíveis para gerir e também a opção de fazer *logout*. Enquanto a seleção de uma carga leva o utilizador para um novo ecrã, a opção de *logout* envia para o servidor a informação que o utilizador terminou a sua sessão e retorna para o ecrã de login. Isto pode ser observado no diagrama de sequência que se encontra na Figura 3.3. É possível observar também que este processo é semelhante ao processo de *login* em termos de fluxo de informações.

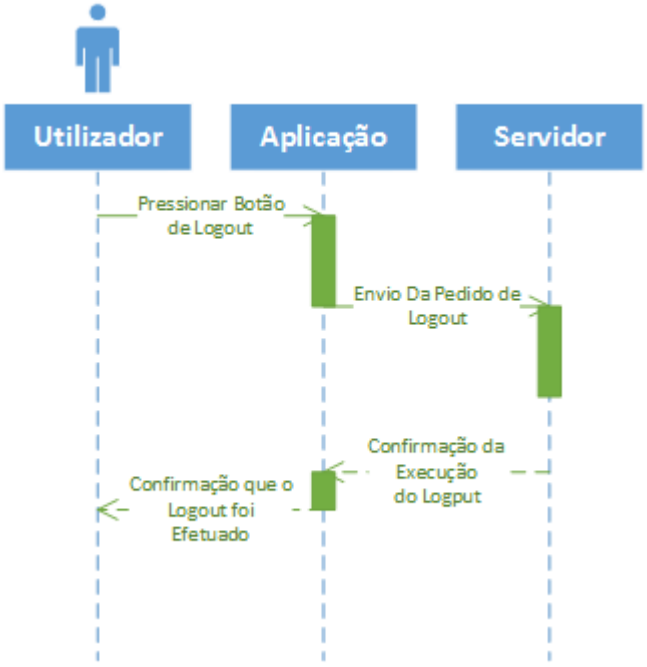


Figura 3.3 - Diagrama de sequência do processo de *logout*.

O ecrã de gestão de cargas irá conter objetos para a introdução de informação por parte do utilizador e botões para realização das diferentes operações. Enquanto os objetos para a inserção de dados variam consoante a carga a controlar, os botões e as suas funções mantêm-se inalterados. Tendo em conta as funcionalidades pretendidas e as características do SGC, que serão descritas mais tarde, é necessário a existência de três botões distintos, "Enviar", "Informação" e "Definir Dia e Hora".

O Botão "Enviar" faz com que seja enviado um comando para ser executado pelo SGC. Antes de o comando ser enviado, as informações necessárias são recolhidas dos elementos presentes nesse ecrã e são posteriormente enviadas para o servidor, que serão processadas juntamente com a informação da ação a realizar e do utilizador que pediu a execução do comando. No entanto, antes de estas serem enviadas são primeiro formatadas numa *string* com um formato que segue o padrão presente na Figura 3.4.

Figura 3.4 - Formato da *string* a enviar para o servidor.

A formatação das *strings* a enviar será a mesma para todas as informações enviadas para o servidor, até mesmo para as operações de *login* e *logout* que foram mencionadas anteriormente, sendo a única variação o número de informações que são adicionadas à *string*. O carater de separação escolhido foi os dois pontos (:).

A sequência de ações relacionadas com o premir do botão "Enviar" podem ser observadas em detalhe na Figura 3.5, no respetivo diagrama de sequência.

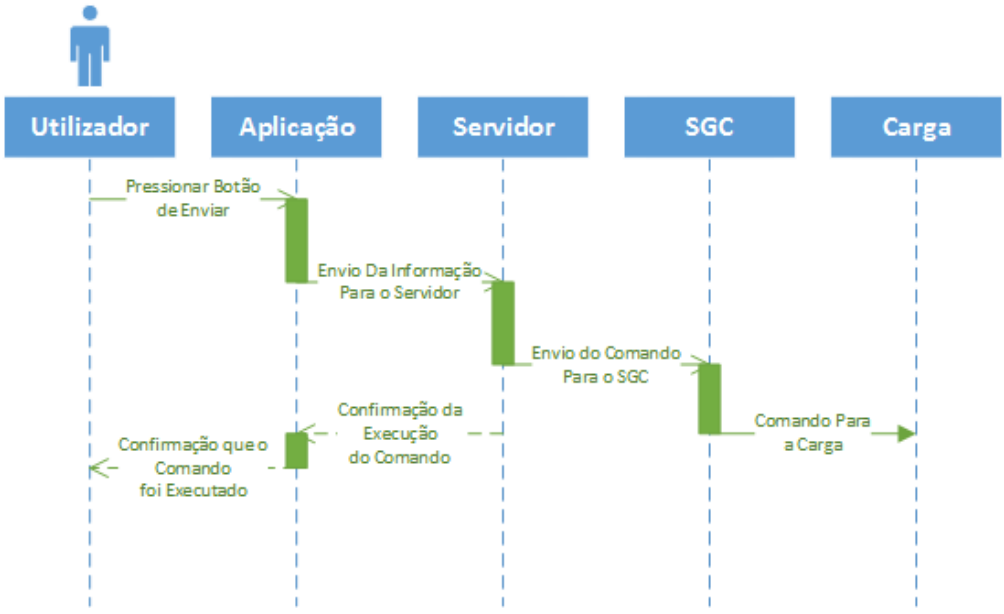


Figura 3.5 - Diagrama de sequência do processo de enviar.

O botão "Informação" envia um pedido de informação ao servidor, este apenas contém o nome do utilizador responsável por este pedido e o tipo de ação a realizar pelo servidor. As informações são recebidas num formato idêntico ao da Figura 3.4, com a exceção de que não se encontra presente o tipo de ação a realizar pelo servidor nem o nome do utilizador que pediu esta ação, uma vez que estas não são necessárias. Na Figura 3.6 é possível observar o diagrama de sequência ilustrativo do pedido de informação.

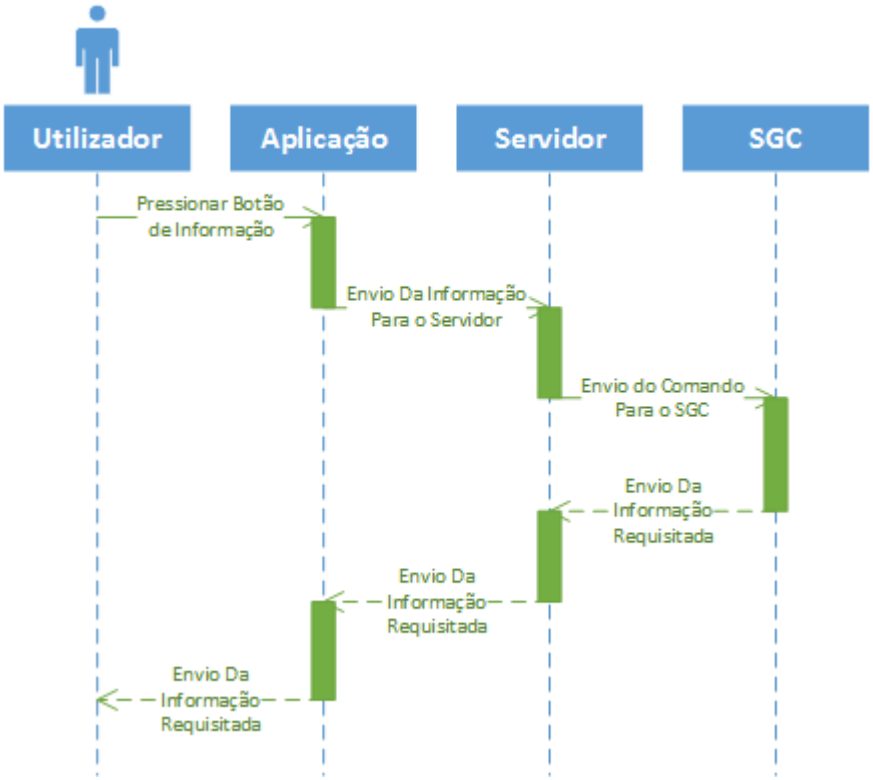


Figura 3.6 - Diagrama de sequência do processo de pedir informação.

Existe ainda o botão que vai levar o utilizador para o ecrã de definição de dia e hora atuais para o SGC, sendo esta a única função deste botão, uma vez que o processo de envio de informação vai ser tratado nesse ecrã separadamente.

Para além dos botões relacionados com as funcionalidades do SGC existe um outro que é o botão "Cancelar" que serve para voltar ao ecrã de seleção de cargas permitindo consequentemente voltar atrás na navegação da aplicação.

O ecrã de definição de dia e hora irá conter um elemento para a introdução das horas e outro para a seleção do dia, e ainda um botão para confirmar o envio desta informação, que tal como anteriormente foi dito será acompanhada por o nome de utilizador e a operação que se pretende que o servidor execute. Também este ecrã possuirá um botão "Cancelar" para voltar atrás na navegação da aplicação.

A formatação da informação a enviar irá seguir o modelo que foi apresentado previamente e a sequência das operações desencadeadas pelo premir deste botão podem ser observadas na Figura 3.7 onde está presente o diagrama de sequência desta operação.

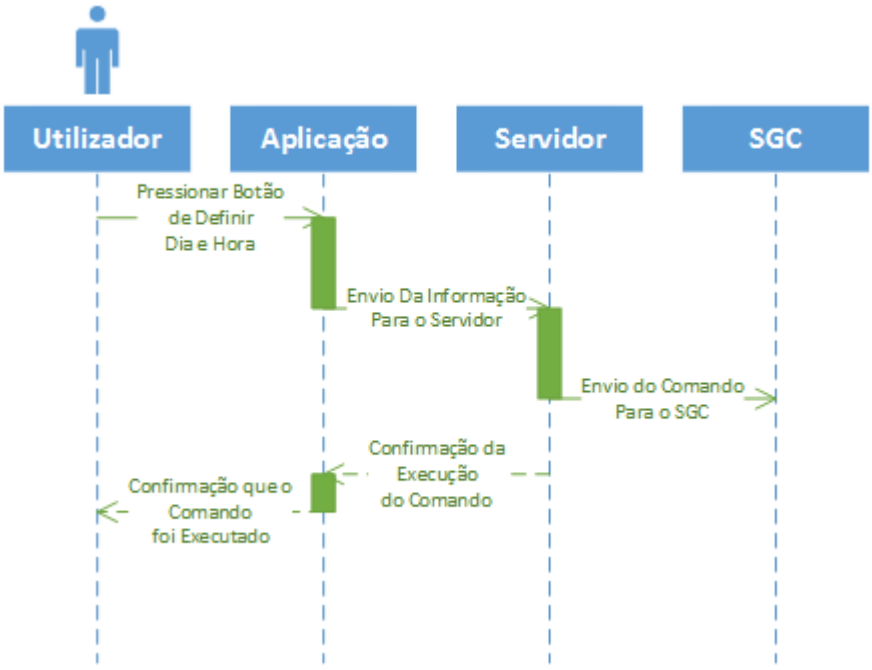


Figura 3.7 - Diagrama de sequência do processo de definir dia e horas atuais.

Uma vez descritos os vários processos a realizar pela aplicação, é importante também mencionar que o método de comunicação a utilizar entre esta e o servidor são *sockets*, e será através destes que a informação irá ser comunicada.

3.5. Servidor

O servidor que recebe as informações da aplicação e da rede elétrica é implementado através de uma Raspberry Pi. O servidor foi programado na linguagem de programação Python e a sua principal função é receber as informações provenientes da aplicação, rede e formatar estas para comandos que possam ser comunicados ao SGC. Para além disto é neste servidor que está albergada a base de dados com os utilizadores que podem aceder ao SGC.

3.5.1. Raspberry Pi

A Raspberry Pi é um computador de dimensões reduzidas desenvolvido pela Raspberry Pi Foundation, uma fundação criada com o principal propósito de melhorar a educação de crianças e adultos no campo da ciência da computação e campos relacionados [40].

A ideia por trás da Raspberry Pi começou a surgir em 2006 quando os seus criadores lançaram a ideia da necessidade de um computador barato em que fosse possível às pessoas treinarem e desenvolverem as suas competências de programação. Várias tentativas foram realizadas para o desenvolvimento de um computador que possuísse estas características, mas só em 2008 com a expansão do mercado de dispositivos móveis, que fez com que os microprocessadores se tornassem mais pequenos e baratos, o projeto se tornou mais viável. Em três anos o Model B entrou em produção em massa e em dois anos venderam-se mais de dois milhões de unidades [41].

Tabela 3.2 - Lista das diferentes versões da Raspberry Pi.

	Model B+	**Model B**	**Model A+**	**Model A**
Imagem				
Chip	Broadcom BCM2835 SoC full HD multimedia applications processor	Broadcom BCM2835 SoC full HD multimedia applications processor	Broadcom BCM2835 SoC full HD multimedia applications processor	Broadcom BCM2835 SoC full HD multimedia applications processor
Processador	700 MHz Low Power ARM1176JZ-F Applications Processor	700 MHz Low Power ARM1176JZ-F Applications Processor	700 MHz Low Power ARM1176JZ-F Applications Processor	700 MHz Low Power ARM1176JZ-F Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor	Dual Core VideoCore IV® Multimedia Co-Processor	Dual Core VideoCore IV® Multimedia Co-Processor	Dual Core VideoCore IV® Multimedia Co-Processor
RAM	512MB SDRAM	512MB SDRAM	256MB SDRAM	256MB SDRAM
Armazenamento	Micro SD	SD	Micro SD	SD
USB 2.0	4x USB Ports	2x USB Ports	1x USB Ports	1x USB Ports
Consumo de Energia/ Tensão	2A @ 5V	1.2A @ 5V	2A @ 5V	1.2A @ 5V
GPIO	40 pinos	26 pinos	40 pinos	26 pinos
Conetor Ethernet	onboard 10/100 Ethernet RJ45 jack	onboard 10/100 Ethernet RJ45 jack	Nenhum	Nenhum

A Rasberry Pi tornou-se desde então um grande sucesso, uma vez que proporciona uma plataforma para a realização de projetos, barata, poderosa e versátil. Este grande sucesso também fez com que surgisse uma grande comunidade de desenvolvedores, que levou a que a Raspberry Pi e os conteúdos para esta tenham-se desenvolvido e diversificado imenso. Esta comunidade também proporciona um apoio muito maior para o desenvolvimento de novos projetos.

Um dos exemplos desta grande variedade de conteúdos existentes para a Raspberry Pi é a existência de vários sistemas operativos capazes de correr nesta placa, sendo que o mais conhecido é o Raspbian.

O Raspbian é um sistema operativo otimizado para o hardware da Raspberry Pi. Este oferece mais de 35000 pacotes de software que podem ser facilmente instalados na Raspberry Pi, tendo este pacote inicial de software sido completado em Junho de 2012. No entanto, o Raspbian continua em desenvolvimento com especial foco na melhoria da estabilidade e da performance [42].

A Raspberry Pi apresenta também vários modelos com especificações de hardware diferentes. Na Tabela 3.2, podem ser observados os diferentes modelos da Raspberry Pi e as suas diferenças a nível de hardware.

Para este projeto foi escolhido o Modelo B+, pois é o que apresenta o maior número de portas USB (*Universal Serial Bus*) e é o que mais potência consegue fornecer para alimentar os componentes ligados a estas portas, evitando a necessidade de utilização de *hubs* USB com alimentação externa.

3.5.2. Linguagem de Programação Python

A linguagem Python é uma linguagem de programação de propósito geral, de código aberto que oferece suporte para programação orientada a objetos, programação funcional, e para estruturas de código imperativas. É geralmente utilizada para programas ou para aplicações de *scripting* [43].

As vantagens desta linguagem são, tal como foi referido acima, o suporte para vários paradigmas de programação o que imediatamente a torna muito versátil. Uma outra vantagem desta linguagem de programação é que é gratuita, para além de código aberto como já foi referido.

Tem uma comunidade *online* ampla que oferece bastante apoio e que está sempre a trabalhar para o seu desenvolvimento. É portável, os programas em Python correm numa grande variedade de plataformas e sistemas operativos como Linux, Windows e Mac OS. É um híbrido entre as linguagens de *scripting* tradicionais e as linguagens utilizadas no

desenvolvimento de sistemas, como o C, C++ e Java, que oferece a simplicidade das linguagens de *scripting* e as funcionalidades mais avançadas das linguagens que passam por um processo de compilação.

Outra das vantagens é poder ser usada em conjunto com outras linguagens de programação, para estender as funcionalidades desta ou para servir como ligação em aplicações desenvolvidas em mais que uma linguagem de programação. É fácil de usar e aprender razão pela qual foi escolhida como linguagem oficial do projeto Raspberry Pi [44]. Possui uma sintaxe simples e não necessita de um processo de compilação, o que aumenta a velocidade de desenvolvimento.

A principal desvantagem do Python é não atingir velocidades de execução como é o caso das linguagens C e C++, e apesar de a velocidade dos computadores atuais reduzir a importância desta desvantagem continuam a existir aplicações em que é necessário este tipo de performance melhorada.

No geral o Python é uma linguagem de programação utilizada em diversas áreas devido às suas características. Em particular, esta linguagem foi selecionada para o desenvolvimento do servidor instalado na Raspberry Pi, por se tratar da linguagem oficial da plataforma, o que faz com que exista uma quantidade de informação abundante sobre a utilização de ambos em conjunto, mas principalmente pela redução do tempo de desenvolvimento obtido pela simplicidade e facilidade de aprendizagem da linguagem.

3.5.3. IDLE

O IDLE é um IDE para Python. Este é dividido em duas janelas, a janela da linha de comandos e a janela de edição, sendo a linha de comandos onde se pode executar linhas de código em Python de forma individual ou observar o resultado da execução de programas, e a janela de edição onde se pode escrever o código dos mesmos [45].Para além disto este IDE apresenta ainda as seguintes características:

- Programado 100% em Python;
- Multiplataforma, corre em Windows, Unix e Mac OS X;
- Depurador, que ainda não se encontra completo.

3.5.4. Especificação do Servidor

O servidor terá como objetivo servir de intermediário entre o utilizador, rede e SGC, para isto este deve ser capaz de receber e enviar informações entre os intervenientes.

A principal função do servidor será receber as informações da aplicação e da rede e converter estas em comandos que possam ser executados pelo SGC, e por sua vez gerir o acesso às funcionalidades deste, permitindo apenas que utilizadores autorizados possam executar comandos.

A gestão do acesso às funcionalidades do SGC é feita através do processo de *login* e *logout*. Durante o processo de *login* o servidor recebe da aplicação uma *string* que indica o nome de utilizador e palavra passe, estes dados vão ser comparados com as informações armazenadas na base de dados de utilizadores, para verificar se é um utilizador válido.

Por razões de segurança a palavra passe será armazenada num formato codificado, tendo sido o algoritmo de encriptação escolhido o SHA-1 (*Secure Hash Algorithim*). Isto vai fazer com que antes de a palavra-passe recebida possa ser comparada com a que está armazenada na base de dados, vai ter de ser encriptada usando o algoritmo de encriptação e só depois comparar o resultado com o que está presente na base de dados. O resultado da verificação destes dados vai ser transmitido à aplicação para ser depois exibido ao utilizador.

No caso em que as informações são validadas, o nome do utilizador vai ser inserido numa lista para utilizadores ativos. Esta lista vai ser posteriormente utilizada no momento em que o servidor recebe algum pedido da aplicação, de forma a validar se o utilizador que a pediu tem uma sessão ativa ou não.

O processo de *logout* no servidor consiste em retirar o nome de utilizador recebido da lista de utilizadores ativos para que este não possa mais executar ações sem ter que passar outra vez pelo processo de *login*. Quando o servidor executa uma operação de envio para o SGC, a informação recebida da aplicação é dividida e convertida para binário. Posteriormente, estas informações são juntas em uma única *string* que será convertida para um valor inteiro, sendo este valor o que vai ser enviado para o SGC. Uma vez enviado o comando para o SGC o servidor envia para a aplicação a informação que o comando foi executado corretamente. Deve-se destacar que este processo só acontece caso o nome de utilizador que acompanha esta informação estiver na lista de utilizadores ativos, caso isto não aconteça nenhuma ação é efetuada e é enviada para a aplicação a informação que ocorreu um erro na execução do comando.

Por fim, a execução do processo de pedir uma informação também faz a verificação do utilizador que fez o pedido e caso este exista na lista de utilizadores ativos vai ser enviado ao SGC um pedido de informação. Este por sua vez vai enviar as informações armazenas para o servidor. O servidor terá como função formatar a

informação recebida e enviar esta para a aplicação de forma a ser visualizada pelo utilizador.

O método de comunicação entre a aplicação e o servidor já foi especificado anteriormente, mas falta mencionar que o método de comunicação entre o servidor e o SGC é comunicação através de porta série. Para isto, foi utilizada uma porta série existente na Raspberry Pi. O pino de envio de informação (TxD) é o pino oito do módulo GPIO, enquanto o pino de receção de informação (RxD) é o pino dez deste mesmo módulo.

3.6. Sistema de Gestão de Cargas (SGC)

O SGC foi implementado usando a tecnologia FPGA (*Field Programmable Gate Array*), a placa utilizada para implementação deste foi uma Basys 2 que foi programada através da linguagem de descrição de hardware Verilog. O SGC vai ser o elemento responsável por processar as informações recebidas do servidor e indicar à carga que operações deve realizar mediante as informações recebidas. Este vai também ser responsável por armazenar as informações necessárias para a gestão das cargas.

3.6.1. Basys 2

A Basys2 (Figura 3.8) é uma placa para o desenho, implementação e teste de circuitos digitais, que poder ir desde simples funções lógicas até controladores mais complexos. A placa é alimentada pelo conetor USB existente na própria, no entanto esta também disponibiliza um conetor para alimentação via uma bateria. É também por ligação USB que a placa é programada.

Figura 3.8 - Placa Basys2 [46].

Esta placa foi desenhada para trabalhar com o ISE WebPack da Xilinx, que é a ferramenta de software utilizada para desenvolver os módulos a serem programados na FPGA e também para simular esses módulos e gerar os ficheiros de programação das FPGAs [46].

Em termos de hardware esta placa tem as seguintes características:

- A FPGA Xilinx Spartan 3E com 100000 portas;
- Atmel AT90USB2;
- Xilinx Plataform Flash ROM (*Read-Only Memory*) para guardar as configurações da FPGA;
- 8 LED (*Light Emitting Diode*);
- 4 Displays de 7 segmentos;
- 4 Botões;
- 8 Interrutores;
- Porta PS/2;
- Porta VGA (*Video Graphic Array*) de 8 bits;
- Seletor de Clock de 25/50/100MHz;
- Socket para segundo clock;
- 4 Expansores de seis pinos;
- Proteção contra curto circuitos em todos os I/O.

3.6.2. Verilog

A linguagem de descrição de hardware Verilog permite descrever um sistema digital com um elevado nível de abstração. Nos níveis mais iniciais de desenvolvimento são utilizadas descrições comportamentais e nos estágios mais avançados descrições estruturais. Estas podem ser combinadas ao longo do processo de desenho dos circuitos.

A descrição comportamental não mostra os detalhes da implementação, mas confere maior flexibilidade com medições de desempenho pouco fiáveis, ao passo que a descrição estrutural é menos flexível para alterações, mas as suas medições de desempenho são mais fiáveis.

Esta linguagem descreve um sistema digital como um conjunto de módulos, sendo que cada um destes módulos tem um interface para interagir com outros módulos e possui uma descrição dos seus componentes.

A razão da escolha desta linguagem, está relacionada com a experiência prévia na utilização desta durante a realização de outros projetos.

3.6.3. Especificação do Sistema de Gestão de Cargas (SGC)

O sistema de gestão de cargas vai ser onde os comandos vão ser processados e comparados com a informação já armazenada de forma a decidir que ação irá ser efetuada sobre a carga. No entanto, as funcionalidades que este vai realizar foram divididas por sub-módulos, sendo que vão existir quatro destes, um que vai ser responsável por organizar os comandos recebidos por ordem de prioridade, outro que vai fazer a decodificação e processamento dos comandos recebidos, um outro que vai fazer a gestão do tempo e por fim o sub-módulo responsável pelas comunicações.

O sub-módulo responsável por organizar os comandos por ordem de prioridade vai ter as seguintes entradas:

- Clock;
- Reset;
- Comando da rede;
- Comando do Utilizador;
- Comando da carga;
- Comando do relógio;
- *Byte* das prioridades.

Este sub-módulo possui apenas uma saída que é para o comando mais prioritário. O esquema de entradas e saídas deste módulo pode ser observado na Figura 3.9.

Figura 3.9 - Sub-módulo das prioridades.

O funcionamento deste módulo assenta nos valores presentes no *byte* das prioridades. Este valor vai ser armazenado no sub-módulo responsável pela

descodificação e processamento das instruções, e passado a este através da entrada para esse efeito. O *byte* de prioridades é dividido em quatro partes, cada uma referente a uma das entradas de comando possíveis (rede, utilizador, dispositivo e relógio), sendo possível ver na Tabela 3.3 como está feita esta divisão. O valor presente nos dois bits associados a cada entrada vai ser a sua prioridade, sendo os valores mais altos os mais prioritários.

Tabela 3.3 - Organização do *Byte* de prioridades.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rede		Utilizador		Dispositivo		Relógio	

Neste sub-módulo vão existir ainda quatro registos que vão ser usados para albergar os comandos provenientes de cada uma das entradas. É neste ponto que entra em funcionamento o valor de prioridade associada a cada uma das entradas, pois o valor do registo que tem maior prioridade é o registo de valor mais alto, neste caso o registo quatro, ou seja, mesmo que todos os registos tenham comandos para ser executados o comando presente neste é que vai ser executado primeiro, sendo os restantes executados seguindo uma ordem decrescente. O valor do comando que vai ser atribuído ao registo quatro vai corresponder à entrada de maior prioridade, e os outros seguiram o mesmo método de atribuição.

Enquanto existir um valor no registo mais prioritário, vai ser sempre este comando que irá ser posto na saída do sub-módulo. Para que o valor de saída corresponda ao de outro registo, é necessário que o mais prioritário esteja a zero, ou seja, não possua nenhum comando para ser executado. Devido a isto, o registo de prioridades precisa ter um valor por defeito que seja diferente de zero, logo o valor pré-definido é 11100100, em que a ordem de prioridade é rede, utilizador, dispositivo e relógio. Na Figura 3.10 é possível observar como os comandos presentes em cada entrada são atribuídos aos registos quando a ordem de prioridades é rede, utilizador, dispositivo e relógio.

Figura 3.10 - Atribuição de comandos aos registos.

A saída deste módulo com o comando mais prioritário vai ser uma das entradas do sub-módulo responsável por descodificar e processar os comandos recebidos que pode ser observado na Figura 3.11. Para além desta entrada existem ainda outras entradas, sendo estas, uma entrada para o clock, o reset, e o endereço do horário de funcionamento que se pretende consultar. Como saídas existe a saída com o horário de funcionamento, uma saída com o estado atual, uma com a ação a realizar pela carga, outra com a ação a realizar pelo relógio e a saída com o *byte* de prioridade que vai ser utilizado no sub-módulo da prioridade.

Figura 3.11 - Sub-módulo de descodificação e processamento.

Para além das operações associadas aos comandos recebidos, este módulo também vai armazenar as informações necessárias para o funcionamento do SGC. Como as informações necessárias variam consoante o tipo de carga estas vão ser discutidas mais tarde.

Os comandos a serem interpretados por este sub-módulo vão ter um tamanho de trinta e dois bits e o seu formato vai ser diferente consoante o tipo de comando uma vez que as informações que este vão conter irão ser diferentes. Cada comando vai ter associado a si um valor que serve para o identificar. Na Tabela 3.4 é possível visualizar todos os comandos possíveis de serem efetuados e os seus valores associados.

Tabela 3.4 - Lista de Instruções.

Lista de Instruções					
Decimal	Binário				Instrução
1	0	0	0	1	Alterar Estado
2	0	0	1	0	Alterar Prioridade
3	0	0	1	1	Alterar Horário
4	0	1	0	0	Alterar Preço
5	0	1	0	1	Alterar Tipo de Controlo
6	0	1	1	0	Informação
7	0	1	1	1	Comparar Preço
8	1	0	0	0	Definir Horas
9	1	0	0	1	Específico da carga
10	1	0	1	0	Específico da carga
11	1	0	1	1	Informação 2

O comando para a alteração de estado vai ter o formato presente na Figura 3.12, este vai ser constituído pelo valor associado ao comando de alterar estado, que se encontra nos quatro bits mais significativos do comando como indica a figura, pelo valor do novo estado, por o bit que indica se o comando é proveniente do utilizador ou da rede e pelos dados específicos da carga. No caso de um carro elétrico pode ser a percentagem de carregamento e o fator de potência, por exemplo.

Figura 3.12 - Formato do comando de alteração de estado.

Na Figura 3.13, é possível ver o fluxograma da execução do comando de alteração de estado onde está descrito como este processo funciona. É importante referir que a verificação do preço de alteração só ocorre quando é um comando proveniente da rede, uma vez que para esta poder fazer alterações na carga é preciso que o valor de incentivo tenha sido previamente aceite. Caso isto não aconteça a rede não poderá fazer alterações ao funcionamento da carga.

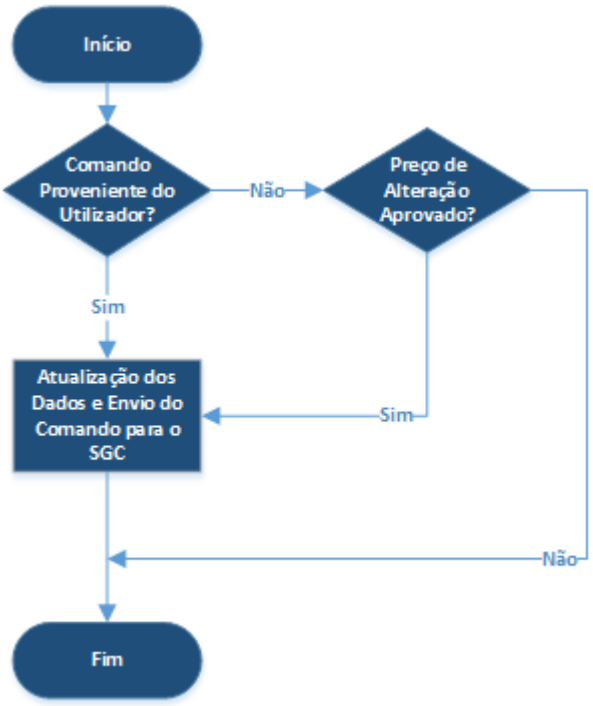


Figura 3.13 - Fluxograma do comando de alteração de estado.

O formato do comando de alteração de prioridade pode ser visualizado na Figura 3.14. O campo para identificar o tipo de comando a executar e bit que identifica se este é proveniente do utilizador ou não, também estão presentes neste comando, mais oito bites que vão conter o novo valor para as prioridades do sistema.

Alterar Prioridade																														
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Prioridade								U/R																		

Figura 3.14 - Formato do comando de alteração de prioridade.

O fluxograma para este comando pode ser observado na Figura 3.15. Ao contrário do comando de alteração de estado este não faz a verificação da aceitação do preço de alteração, porque apenas pode ser executado pelo utilizador, logo, se o comando não originar deste nenhuma operação irá ser executada.

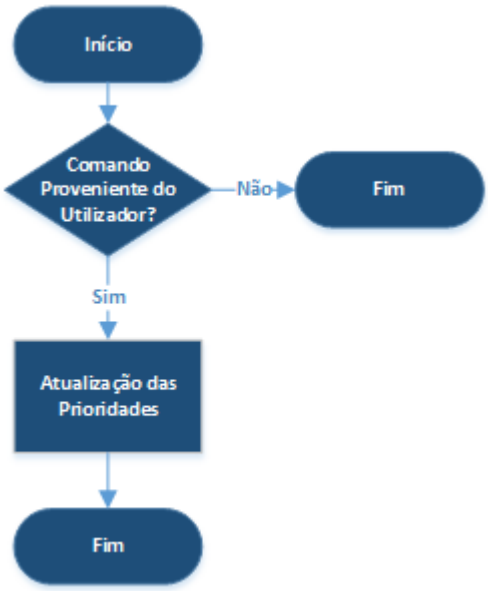


Figura 3.15 - Fluxograma do comando de alteração de prioridade.

A instrução para a alteração do horário de funcionamento apresenta o formato ilustrado na Figura 3.16. O valor representativo do comando a efetuar e o que indica se este veio do utilizador ou não, também se encontram presentes tal como acontece nos exemplos anteriores. Este comando no entanto vai ter ainda um campo para identificar o dia em que se se pretende fazer a alteração do horário e o campo para indicar o novo horário.

O campo para o novo horário vai ter um tamanho de vinte quatro bits e cada bit representa uma hora do dia, assim sendo, se o um bit estiver a um significa que durante a hora correspondente a carga vai estar a funcionar, se ele estiver a zero a carga vai estar desligada.

Alterar Horário																															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Dia			U/R	Novo Horário																							

Figura 3.16 - Formato do comando de alteração de horário.

Na Figura 3.17 é possível ver o fluxograma para o comando de alteração de horário de funcionamento. Este é bastante parecido com o comando de alteração de estado, com a diferença que como este apenas afeta os registos internos não vai existir um envio de informação para a carga apenas uma atualização destes mesmos registos.

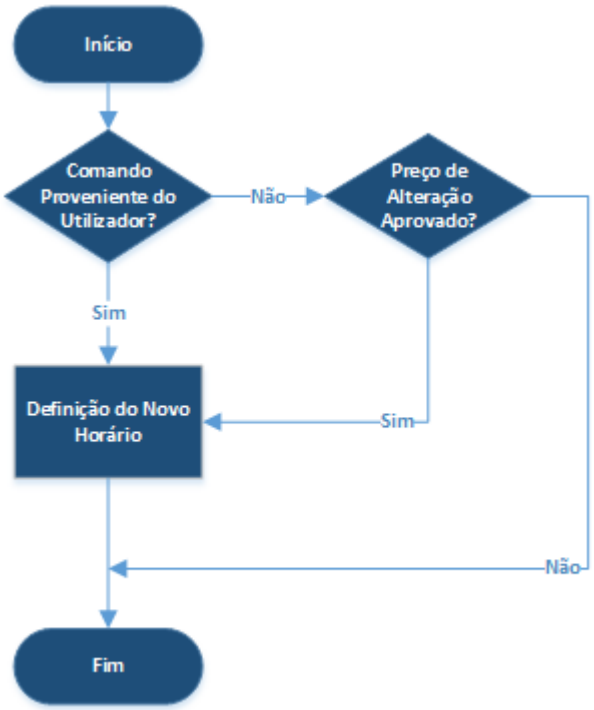


Figura 3.17 - Fluxograma do comando de alteração de horário.

O formato da instrução responsável pelo preço de alteração pode ser observado na Figura 3.18, para além dos campos para identificar o comando e o emissor deste existe, também um campo com o valor do preço.

O preço vai estar em cêntimos, ou seja, se neste campo estiver o valor um vai corresponder a um cêntimo. Como este campo tem oito bits de tamanho o valor pode ir até 255 cêntimos o que se traduz num valor de dois euros e cinquenta e cinco cêntimos. Este valor é considerado suficiente uma vez que dá muita margem de manobra quando comparado com o preço de um kilowatt/hora.

Alterar Preço																													
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Novo Preço								U/R																			

Figura 3.18 - Formato do comando para o preço de alteração.

O comando do preço de alteração vai apenas poder ser efetuado pelo utilizador, uma vez que diz respeito ao preço que este quer definir como mínimo, para aceitar que o funcionamento da sua carga seja alterado pela rede. Como tal o fluxograma deste vai ser parecido com o do comando de alterar prioridade tal como pode ser observado na Figura 3.19, sendo a única diferença o registo que este vai alterar.

Figura 3.19 - Fluxograma do comando de preço de alteração.

O comando para alterar o tipo de controlo tem o formato que pode ser observado na Figura 3.20, sendo o formato semelhante ao do alterar prioridade, com a exceção que este vai conter informação acerca do novo tipo de controlo em vez da prioridade.

Alterar Controlo																														
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Novo Tipo Controlo								U/R																		

Figura 3.20 - Formato do comando de alteração do tipo de controlo.

Figura 3.21 - Fluxograma do comando de alteração do tipo de controlo.

Como o comando de alteração do tipo de controlo também é um comando que apenas pode ser executado pelo utilizador, o seu fluxograma irá ser semelhante aos que já foram apresentados para comandos com esta característica, onde a única diferença

como se pode ver na Figura 3.21 é o registo que é alterado, neste caso o registo associado ao tipo de controlo.

O comando de informação foi dividido em dois, pois o tamanho de um comando não é suficiente para enviar todas as informações armazenadas nos registos internos, daí o processo de enviar informações para o exterior ter sido dividido, no entanto ambos funcionam de forma idêntica. Estes vão apenas conter as diversas informações para colocar no exterior e não vai existir nenhuma verificação extra para a sua execução, uma vez que mesmo sem ter existido uma verificação de preço que tenha sido aceite, a rede pode pedir o envio dos dados armazenados. Na Figura 3.22 pode ser observado o fluxograma destes comandos.

Figura 3.22 - Fluxograma dos comandos de informação.

O comando para a comparação de preço tem o formato que pode ser observado na Figura 3.23, sendo este um formato idêntico a outros comandos, com a diferença que contem o preço oferecido pela rede para que esta possa alterar o funcionamento da carga.

Comparar Preço																															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Preço a Comparar								U/R																			

Figura 3.23 - Formato do comando de comparação de preço.

O preço que foi definido previamente pelo utilizador é comparado com o preço que a rede oferece para que possa alterar o funcionamento da carga, se este for aceite vai ser acionado um sinal indicativo disso mesmo para que os futuros comandos provenientes da rede possam ser executados. Caso o preço não seja aceite este sinal vai ser posto a zero impossibilitando assim que os comandos provenientes da rede sejam executados. Isto pode ser observado na Figura 3.24 que representa o fluxograma deste comando.

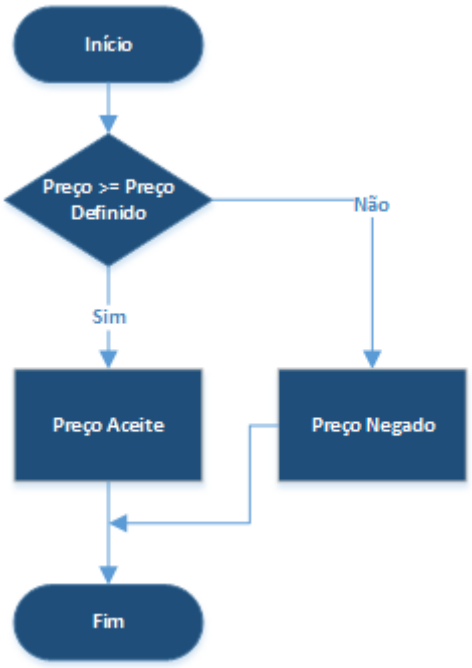


Figura 3.24 - Fluxograma do comando de comparação de preço.

O comando para a definição das horas e do dia do sistema vai ter o formato presente na Figura 3.25, este vai conter todos os elementos necessários para a definição do novo horário do sistema desde o dia até aos segundos.

Definir Horas																													
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Comando				Dia			Horas					Minutos						Segundos						U/R							

Figura 3.25 - Formato do comando para a definição do dia e da hora.

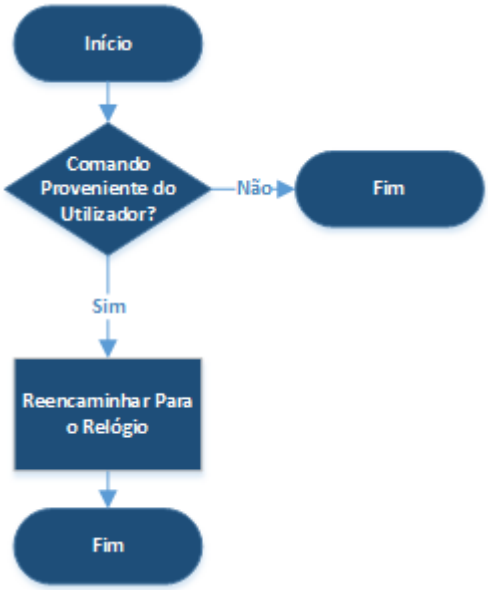


Figura 3.26 - Fluxograma do comando de definição de dia e hora.

A definição de dia e horas apenas pode ser executada pelo utilizador e para além de afetar este sub-módulo vai também afetar o sub-módulo do relógio, uma vez que o controlo do tempo é responsabilidade do sub-módulo do relógio e os registos que vão ser alterados por este comando também se encontram neste. Logo a ação que vai ser realizada por este comando é reencaminhar o comando para o relógio. Na Figura 3.26 pode ser observado o fluxograma para este comando.

Como foi mencionado anteriormente, o sub-módulo do relógio, que pode ser observado na Figura 3.27, é responsável por fazer o controlo do tempo e dos horários de funcionamento das cargas. Este faz o incremento das unidades de tempo recorrendo aos pulsos do oscilador como unidade base e aliado a isto, vai utilizar o dia atual para ir buscar ao sub-módulo de descodificação de instruções o horário de funcionamento para esse dia da semana. A hora atual vai ser usada para percorrer os vinte e quatro bits do horário de funcionamento e verificar o estado atual, que também é recebido através de uma entrada, com o estado que está no horário de funcionamento. Mediante as informações vai ser emitido um comando para ligar ou desligar a carga, que vai ser enviado por uma saída que vai estar ligada à entrada para o relógio no sub-módulo das prioridades.

As entradas para este módulo são o relógio, o *reset*, o comando de entrada para a definição das horas e a entrada de dados, que vai conter o horário de funcionamento para ser analisado, e o estado atual. Como saídas este módulo vai conter o comando de saída e o dia atual, este último vai ser utilizado para selecionar o horário referente a esse dia.

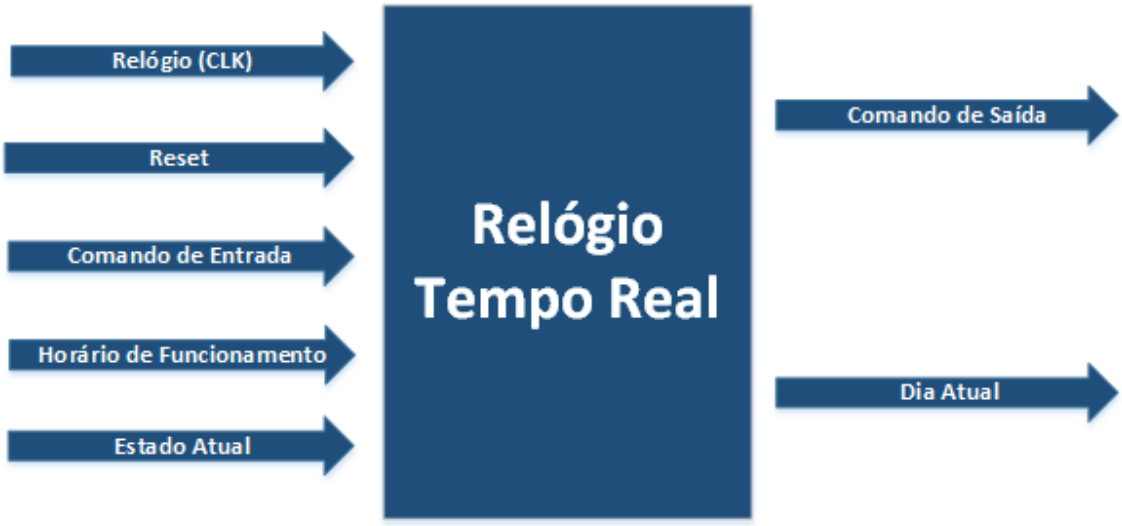


Figura 3.27 - Sub-Módulo do relógio.

Por último o sub-módulo das comunicações, que é visível na Figura 3.28, vai implementar o protocolo de comunicação via porta série RS232. Este foi escolhido por ser um protocolo simples de implementar e ainda é bastante utilizado; no entanto como o

módulo de comunicação é independente dos outros, o protocolo de comunicação pode ser alterado mudando apenas o módulo de comunicação.

Este módulo tem como entrada o sinal de relógio, de *reset*, a mensagem a ser enviada e o sinal de receção de dados (RxD). Como saída vai ter o comando recebido e o sinal de envio de dados (TxD).

Figura 3.28 - Sub-módulo das comunicações.

O módulo das comunicações e o da descodificação serão os únicos módulos que estarão sujeitos a mudanças, como já foi referido anteriormente o módulo das comunicações pode ser alterado conforme o protocolo de comunicação que se pretenda utilizar. O módulo da descodificação de instruções irá mudar consoante a carga que se pretenda utilizar, de acordo com a carga que se pretende controlar o número de comandos pode aumentar ou diminuir. As informações que precisam ser armazenadas também variam consoante a carga, no entanto os comandos descritos anteriormente são constantes para todos os tipos de cargas. Os possíveis estados que cada tipo de carga podem também variar de carga para carga. Na Tabela 3.5 é possível ver os comandos específicos para cada um dos tipos de carga.

Tabela 3.5 - Comandos específicos das cargas.

Cargas	**Banco de Condensadores**	**Filtro Ativo**	**Carro Elétrico**
Comandos Específicos das Cargas	Definir Número Máximo de Níveis	Não definido	Informação do estado de carga
	Definir Número de Níveis	Não definido	Definir estado de carga mínimo

As alterações à memória vão ser poucas, tomando como exemplo o carro elétrico e o banco de condensadores apesar de os comandos únicos destas cargas alterar elementos diferentes, estes são armazenados da mesma forma. Já no caso do filtro ativo que não

possuí nenhum comando específico a memória pode ser reduzida, uma vez que não precisa dos registos referentes aos comandos específicos.

Os diferentes estados das cargas podem ser visualizados na Tabela 3.6. Cada um destes estados é referente a atividades que estes tipos de cargas podem desempenhar, tendo o carro elétrico mais estados, uma vez que se prevê que este venha a desempenhar um papel muito importante no funcionamento das redes elétricas.

Tabela 3.6 - Estados das cargas.

Cargas	**Banco de Condensadores**	**Filtro Ativo**	**Carro Elétrico**
Estados	Parado	Parado	Parado
	Bloqueado	Bloqueado	Bloqueado
	Avariado	Avariado	Avariado
	Ativo Totalidade	Compensação	Em Carregamento
	Ativo Parcial	Fator de Potência	A Fornecer
	Não definido	Harmónicos	Carregamento Parcial
	Não definido	Tudo	A Fornecer Parcial
	Não definido	Não definido	Carregamento com Reativos

3.7. Conclusão

Neste capítulo foi apresentado o desenvolvimento do projeto do Sistema de Gestão de Cargas (SGC), começando com a descrição das funcionalidades pretendidas para este, com base nas informações recolhidas no capítulo anterior sobre o papel que estes sistemas vão desempenhar nas redes elétricas e explicando as escolhas das cargas a serem geridas neste projeto. Terminada a descrição das funcionalidades pretendidas foi dado início à descrição geral do sistema.

A descrição geral do sistema específica quais os elementos que constituem o sistema, sendo estes elementos: a aplicação para interface com o utilizador, o servidor e o SGC. As funcionalidades gerais de cada um destes elementos são também detalhadas, bem como as interações destes com outros elementos do sistema e com intervenientes externos, como é o caso da rede elétrica e do utilizador. Cada uma destas partes é então projetada para desempenhar as suas funções dentro do sistema.

A primeira das partes a ser projetada foi a aplicação de interface com o utilizador, sendo em primeiro lugar descritos os elementos e as ferramentas utilizadas para o desenvolvimento da aplicação, passando depois para a especificação desta onde é descrita

a forma como esta é implementada, a sua organização, bem como o seu comportamento mediante os estímulos do utilizador e do servidor. As ações desencadeadas por estes nos outros elementos do sistema são também especificados sob a forma de diagramas de sequência, para se poder especificar não só as ações desencadeadas pela aplicação, mas também a ordem em que elas ocorrem.

O projeto do servidor começou também pela descrição dos elementos e ferramentas utilizados no seu desenvolvimento, passando depois para a descrição do funcionamento pretendido para este e das funções necessárias para que desempenhe a sua função no sistema.

O SGC teve como fase inicial do seu projeto a descrição dos componentes utilizados no seu desenvolvimento, passando depois para as tarefas que este desempenha no sistema. Estas tarefas estão divididas em sub-módulos que as implementam de forma segmentada. Para além da descrição das funcionalidades e operações realizadas por cada sub-módulo, são também descritas as entradas e saídas destes para perceber o fluxo de entrada e saída de informação em cada um.

No sub-módulo para descodificação e processamento de instruções é também feita a descrição dos formatos dos comandos e são apresentados fluxogramas para especificar o seu comportamento.

CAPÍTULO 4

Simulação do Sistema de Gestão de Cargas Desenvolvido

4.1. Introdução

Neste capítulo é abordado o processo se simulação do módulo de gestão de cargas. Esta foi a única parte do trabalho que foi simulada sendo esta a única fase em que as ferramentas de simulação disponíveis trazem vantagens significativas em relação ao teste direto no hardware. O processo de simulação foi dividido pelos diferentes sub-módulos, sendo cada um destes simulado de forma individual antes de todos serem ligados entre si de forma a constituir o módulo de gestão de cargas.

A simulação de cada um destes sub-módulos foi também segmentada pelas funcionalidades que cada um deles tem de desempenhar, ou seja, em primeiro lugar foram simuladas as instruções individualmente, só depois o módulo foi testado na sua totalidade, ao realizar várias instruções em sequência para ver como este se comportava, sendo este o caso que se aproxima mais do funcionamento real.

A ordem em que os sub-módulos foram simulados foi baseada na ordem em que eles são chamados a atuar no processo de funcionamento do módulo geral. Por isso, foi primeiro simulado o módulo responsável por organizar os comandos recebidos, consoante as prioridades definidas, seguido do módulo responsável por fazer a descodificação e execução das instruções, e por fim, foi simulado o módulo do relógio. Terminada a simulação destes, e verificados os resultados obtidos, estes foram interligados entre si de forma a simular o funcionamento geral do Sistema de Gestão de Cargas (SGC).

4.2. Simulação do Módulo de Controlo de Cargas

O módulo de controlo de cargas foi desenvolvido e simulado utilizando as ferramentas da Xillinx. O ISim foi utilizado para simular este módulo de forma a poder verificar as suas funcionalidades antes da sua implementação em hardware. Uma vantagem da utilização desta ferramenta de simulação é que permite visualizar e analisar

todos os sinais internos dos módulos simulados, ao contrário de uma implementação direta no hardware, que apenas permite a visualização dos sinais de saída.

Uma vez que o módulo de gestão de cargas é constituído por vários módulos mais simples, cada um é responsável por uma função bem definida, ou seja, a simulação será feita numa primeira fase por partes, sendo simulados cada um dos módulos de forma individual para que seja mais fácil verificar o seu correto funcionamento. Após verificado o funcionamento de cada módulo, será feita a simulação da interação entre estes, de forma a validar que a comunicação entre módulos é feita de forma correta.

4.3. Simulação do Sub-Módulo Responsável pelas Prioridades

O primeiro módulo a ser simulado foi o módulo responsável pelas prioridades dos comandos provenientes dos diferentes intervenientes no controlo de cargas. Para verificar o correto funcionamento deste foram simuladas todas as ações que o mesmo pode realizar, para verificar se a sua saída é a pretendida. Como neste caso o módulo apenas compara as prioridades de cada interveniente e coloca à sua saída o comando do ator com mais prioridade, é apenas necessário averiguar se isto é feito de forma correta e, se na eventualidade de os atores com maior prioridade não possuírem nenhum comando ativo, os com menos prioridade não ficam bloqueados à espera destes.

4.3.1. Verificação da Entrada de comando DEV (Device)

A primeira entrada de comando a ser verificada foi a entrada "dev", que é a entrada proveniente do dispositivo. A verificação desta entrada passa pela introdução de um comando e verificar se com um *byte* de prioridade fixo e com todos os outros comandos nulos, ou seja, sem comandos válidos nas outras entradas, a saída do módulo corresponde ao que se encontra presente nela.

Na Figura 4.1 podemos observar o resultado de uma simulação obtida, onde foi introduzida na entrada "dev" um comando com o valor 00010000000010100011001011111000 e que o valor da saída "comm", que contém o comando de saída que é o mais prioritário, vai ser atualizado com este valor. Durante toda esta simulação o valor da entrada "prio", que contém o valor das prioridades atuais, vai ter o valor de 00111001, o que faz com que a ordem das prioridades seja a seguinte: "loc", "dev", "rt_clk", "net". Os comandos das outras entradas ("net", "loc", "rt_clk") vão se encontrar a zero, de forma a verificar que independentemente da prioridade quando só existe o comando da entrada "dev" este é posto na saída

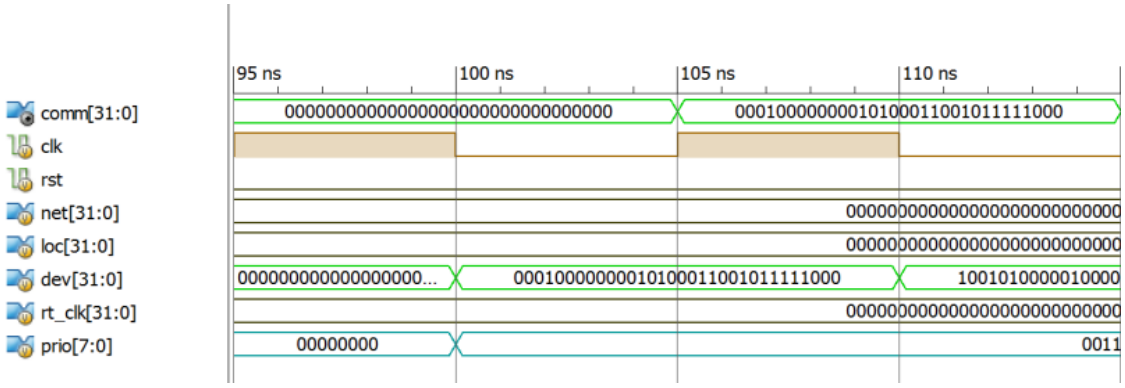


Figura 4.1 - Simulação da entrada dev fase um.

De forma a verificar que o módulo não fica preso após o primeiro comando e que este está pronto para receber outros comandos pela mesma entrada, foi introduzido um novo comando de forma a verificar que este também é colocado à saída do módulo, tal como o anterior. Isto pode ser observado na Figura 4.1 e na Figura 4.2 onde podemos apurar que mediante a introdução de um comando diferente na entrada "dev", desta vez o comando 10100100011110000000000000000000, este vai também ser posto na saída "comm".

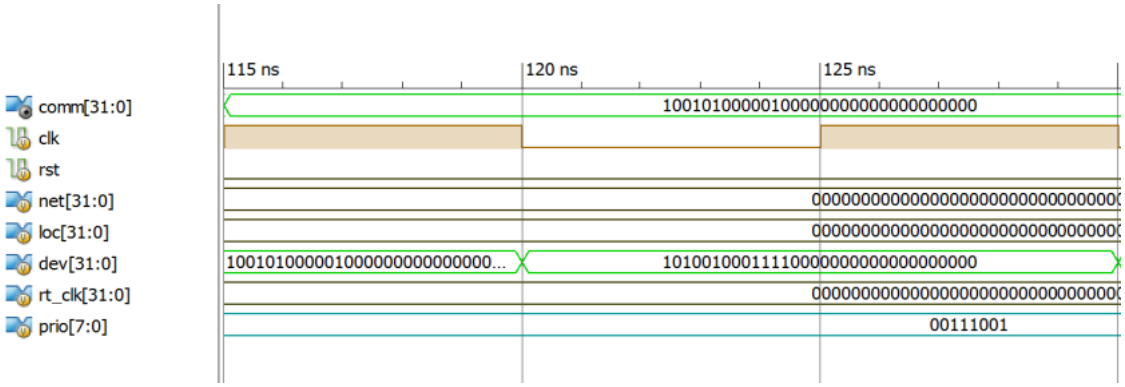


Figura 4.2 - Simulação da entrada dev fase dois.

Com base nesta simulação pode concluir-se que a entrada "dev" funciona corretamente de forma individual e que não é bloqueada pelas outras entradas quando estas são nulas, mesmo quando alguma delas tem uma prioridade maior, neste caso a entrada "loc".

4.3.2. Verificação da Entrada de comando LOC (Local)

A simulação seguinte foi a da entrada "loc", entrada proveniente do utilizador, foi utilizado o mesmo *byte* de prioridade que no caso anterior (00111001) e o método de simulação utilizado foi também o mesmo, ou seja, todos os outros comandos são colocados a zero de forma a não interferirem no processo de seleção do comando a ser

colocado à saída. No entanto, como neste caso a entrada "loc" é a que possui maior prioridade esta seria sempre a primeira a ser colocada na saída.

Através da Figura 4.3 pode ser observado que ao ser introduzido o comando 00110001111111111111111111111111 na entrada "loc" este vai ser passado para a saída "comm" tal como pretendido.

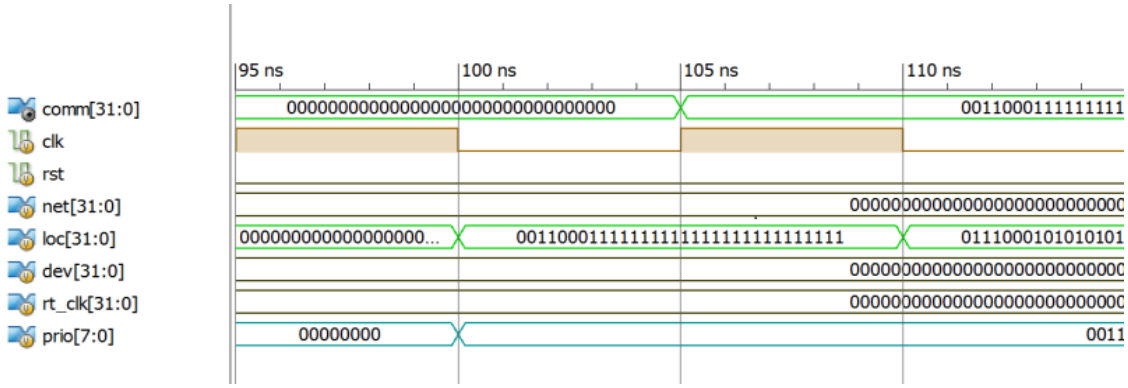


Figura 4.3 - Simulação da entrada loc fase um.

Tal como na simulação da entrada "dev" foi introduzido mais do que um comando de forma a verificar o funcionamento contínuo desta e também que não há problemas na transição entre diferentes comandos.

A continuação da simulação pode ser vista na Figura 4.4 onde podemos constatar que o segundo comando introduzido,00100000011101000110010111111000, também é posto à saída do módulo como pretendido.

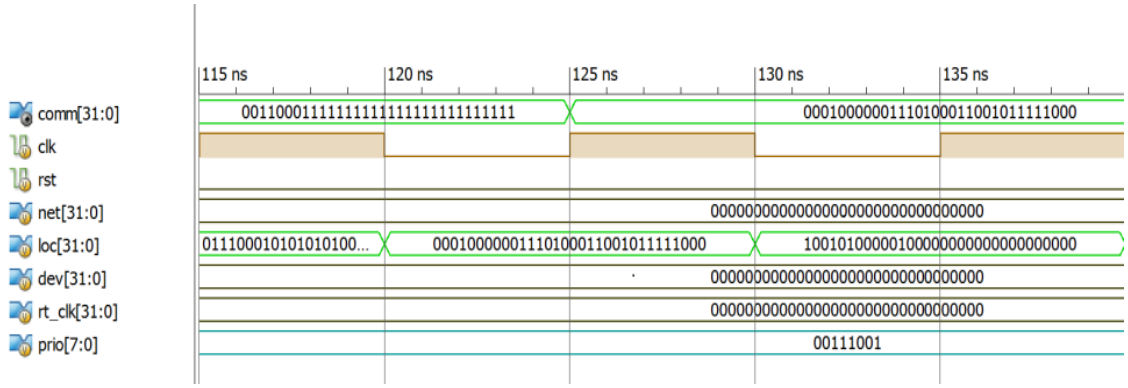


Figura 4.4 - Simulação da entrada loc fase dois.

Desta forma, pode ser verificado o funcionamento da saída "loc" de forma individual e que a saída à qual é atribuída a maior prioridade também funciona de forma correta, mesmo sozinha. Nenhuma das outras saídas foi simulada com a prioridade mais alta porque a atribuição de prioridades funciona por alocação de registos, logo, se funciona bem para este caso, os outros ficam automaticamente verificados.

4.3.3. Verificação da Entrada de comando NET (Network)

A entrada "net", entrada proveniente da rede, foi simulada de forma igual às anteriores com o mesmo *byte* de prioridade e o primeiro comando de teste foi o 01110001010101010000000000000000, que como podemos observar pela Figura 4.5 foi posto com sucesso na saída "comm", tal como desejado.

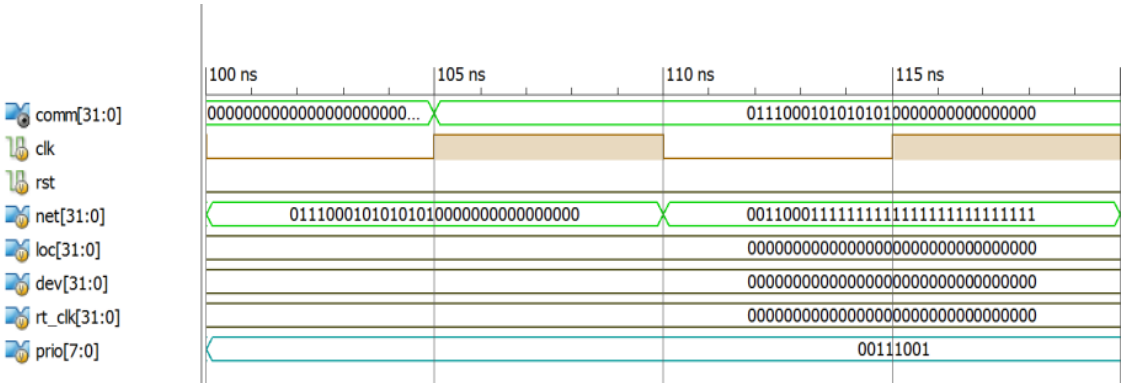


Figura 4.5 - Simulação da entrada net fase um.

O segundo comando utilizado para testar a sequência de comandos foi o 00110000111100000000111111110000 e pode ser observado na Figura 4.6. Figura esta que demostra a continuação da simulação em que este comando também é colocado com sucesso na saída "comm". Pode-se assim concluir com base nesta simulação que o funcionamento individual da entrada net está de acordo com o pretendido.

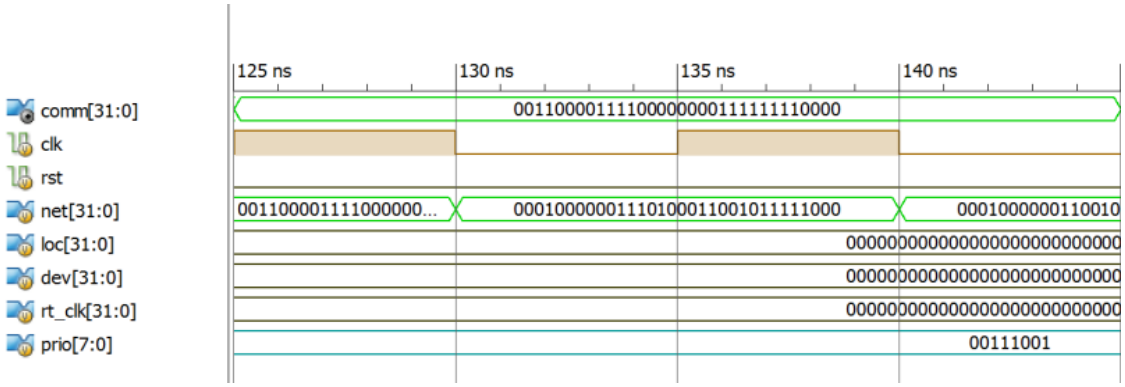


Figura 4.6 - Simulação da entrada net fase dois.

4.3.4. Verificação da Entrada de comando RT_CLOCK (Real Time Clock)

A última entrada de comando a ser testada foi a "rt_clk", entrada proveniente do relógio. Tal como nas simulações anteriores foi mantido o *byte* de prioridade. Para este caso foi apenas verificado um comando. O comando selecionado para ser simulado em primeiro lugar foi o 00010000000010100011001011111000 e como se pode observar na Figura 4.7 este comando é colocado na saída, verificando assim o correto

funcionamento desta entrada. Esta simulação é a última da fase de análise das entradas de comandos de forma individual, sendo o passo seguinte simular todas elas em conjunto de forma a confirmar que estas funcionam bem quando utilizadas em simultâneo.

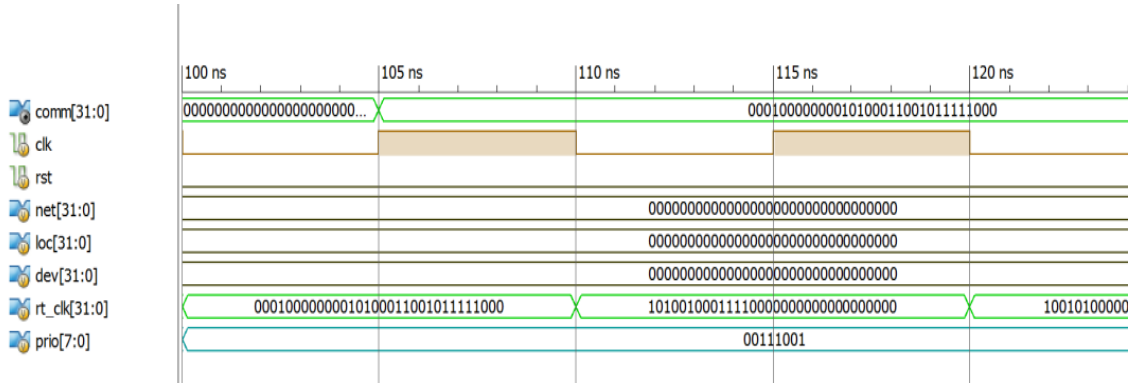


Figura 4.7 - Simulação da entrada rt_clk.

4.3.5. Verificação do funcionamento do sub-módulo das prioridades

Depois de todas as entradas terem sido verificadas de forma individual, foi simulado o funcionamento do módulo com comandos em todas as entradas simultaneamente, de forma a fazer a verificação num cenário mais aproximado ao funcionamento real pretendido para o módulo. Para fazer esta verificação foram feitas duas simulações diferentes, em que a diferença mais significativa é o *byte* de prioridade, de forma a verificar também o funcionamento deste, uma vez que nas simulações individuais este manteve-se sempre fixo visto que o objetivo era apenas testar as entradas.

O *byte* de prioridade usado foi o 11100100, que se traduz na seguinte ordem de prioridades, "net", "loc", "dev" e "rt_clk", o que faz com que os comandos a ser colocados na saída sigam também esta ordem. Os comandos introduzidos em cada uma das entradas são diferentes de forma a ser mais fácil verificar a sua origem para comprovar que as prioridades estão a ser cumpridas. Como tal, na entrada "net" foi introduzido o comando 01110001010101010000000000000000, na entrada "loc" o comando introduzido foi 00110001111111111111111111111111, na entrada "dev" foi o 10010100000100000000000000000000 e por fim na entrada "rt_clk" foi introduzido o comando 00010000000010100011001011111000.

O início deste processo pode ser observado na Figura 4.8, e é possível verificar que todos os comandos são introduzidos de forma simultânea em que o primeiro a ser posto na saída "comm" é o da entrada mais prioritária, neste caso a entrada "net". Após algum tempo esta entrada é colocada a zero para indicar que todos os comandos dessa

entrada foram transmitidos ao módulo seguinte e que os das outras entradas podem ser agora transmitidos.

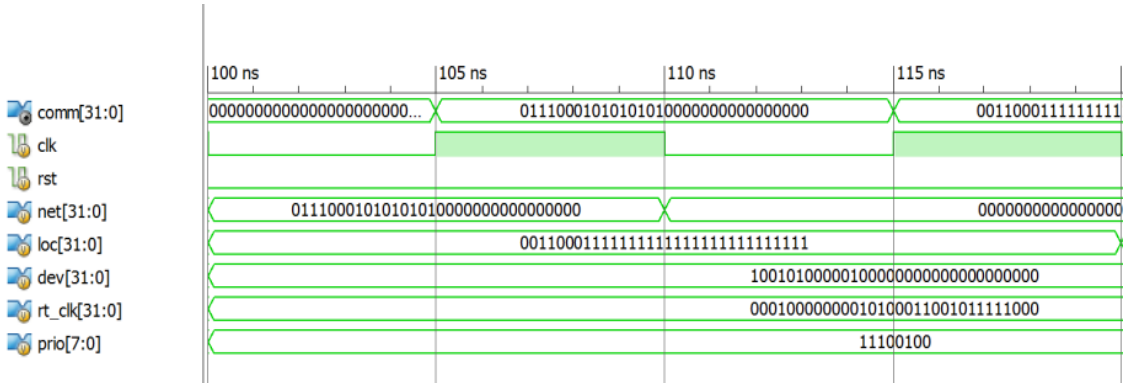


Figura 4.8 – Simulação do módulo completo fase um.

A continuação desta simulação pode ser vista na Figura 4.9, onde se constata que os comandos que vão sendo postos na saída seguem a ordem das prioridades, ou seja, a seguir do comando presente na entrada "net" vem o comando da entrada "loc" seguido do da entrada "dev" e por fim o que está na entrada "rt_clk". Estes resultados comprovam que o funcionamento do módulo é o pretendido mesmo com comandos presentes simultaneamente em todas as entradas.

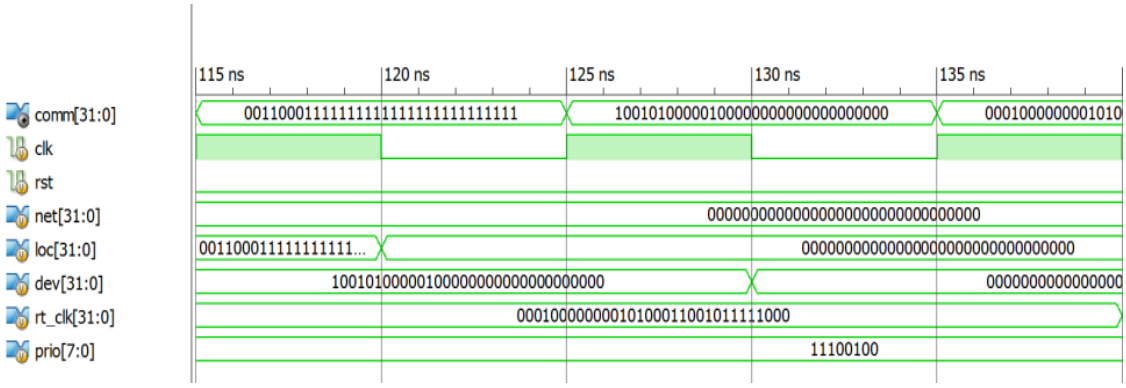


Figura 4.9 - Simulação do módulo completo fase dois.

A simulação seguinte seguiu o mesmo processo, com a única diferença que o *byte* de prioridades foi alterado para 00111001 o que muda a ordem das prioridades para "loc", "dev", "rt_clk" e "net". É possível observar na Figura 4.10 que esta alteração se traduz também numa modificação da ordem dos comandos que são colocados na saída.

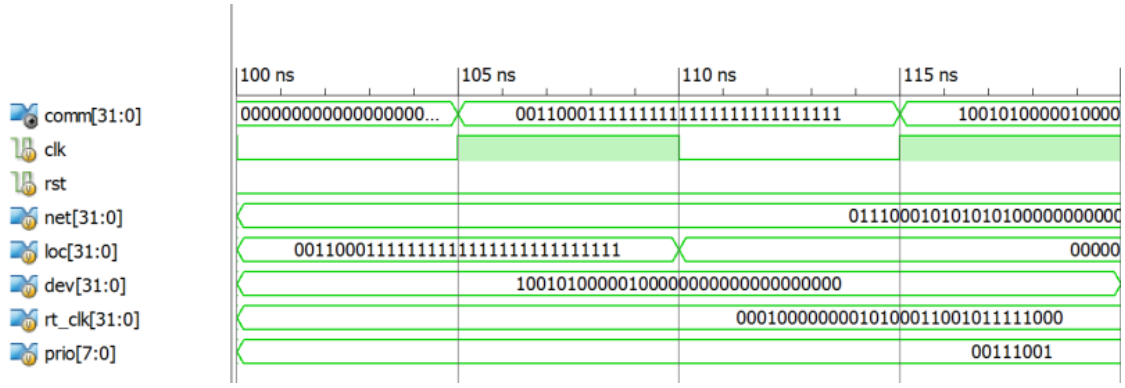


Figura 4.10 - Simulação do módulo completo fase três.

A continuação da simulação é possível ser visualizada na Figura 4.11 que também demostra que a ordem dos comandos colocados na saída é alterada com a alteração do *byte* de prioridades. Isto demonstra que a alteração afeta todas as entradas e não apenas a mais prioritária.

Com esta simulação termina-se a verificação do sub-módulo responsável pelas prioridades, podendo-se concluir através das simulações anteriores que o seu funcionamento vai de encontro ao que é pretendido e que este fornece as informações necessárias para que os outros módulos possam desempenhar as suas funções corretamente.

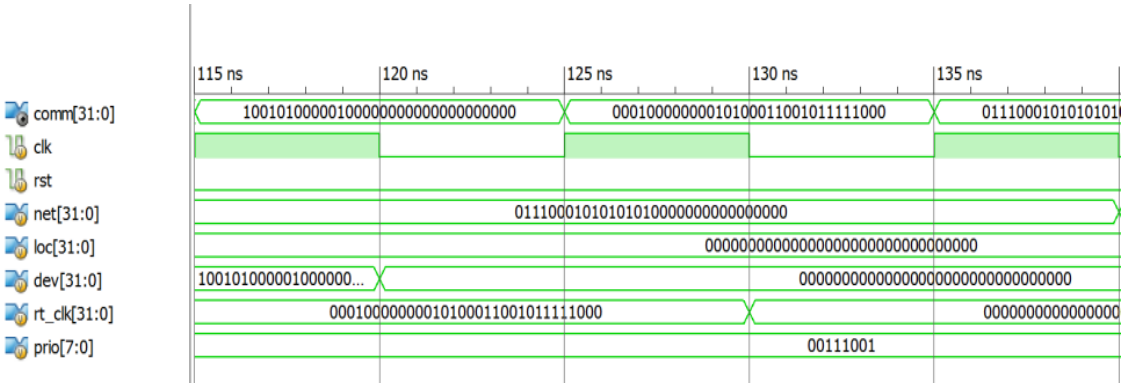


Figura 4.11 - Simulação do módulo completo fase quatro.

4.4. Simulação do Módulo Responsável pela Descodificação de Instruções

O módulo seguinte a ser testado foi o responsável pela descodificação das instruções recebidas e que previamente foram enviadas para este pelo módulo das prioridades, após estas terem sido organizadas de acordo com as prioridades atribuídas.

Para verificar este módulo são inseridos comandos neste e observa-se as alterações que este produz quer a nível de registos internos, quer a nível de saídas e se estão são as pretendidas. Para isso nas seguintes simulações serão visualizadas as saídas e os registos

internos relevantes para a execução de cada comando. Por exemplo, ao simular um comando de alteração de estado, para além de ser visualizado o resultado que este produz nas saídas, também é necessário confirmar que o registo que guarda o estado atual foi também corretamente alterado. Um exemplo pode ser observado na Figura 4.12. Foi introduzido na entrada "cmd" o comando 00010000001110100011001011111000, que corresponde a um comando de alteração de estado, em que o novo estado corresponde ao valor 00000011, que são os oitos bits que vêm a seguir aos quatro primeiros que indicam o estado. Como pode ser visualizado na figura a saída "state_out", que indica o valor do registo de estado, passa a ter este valor tal como é pretendido e a saída "action", saída com a ação a realizar pela carga, também é atualizada com o valor do comando de entrada, que neste caso significa que na presença deste comando o módulo está a funcionar como é pretendido.

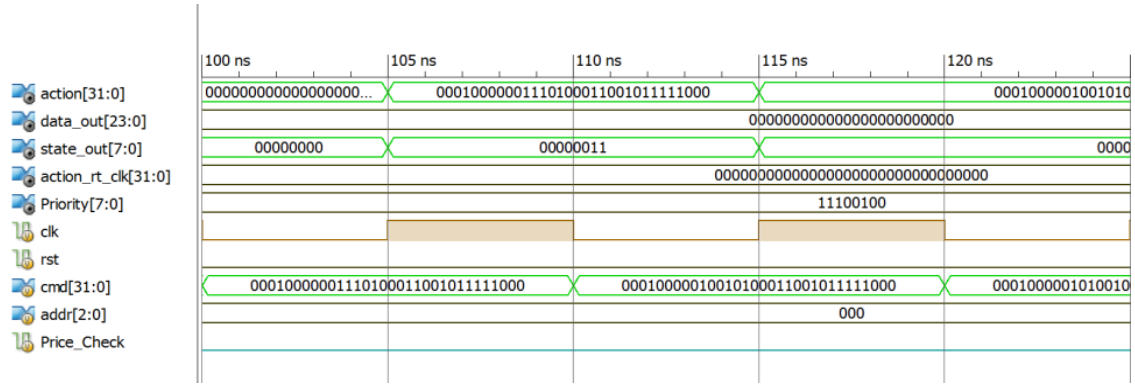


Figura 4.12 - Simulação do módulo de descodificação de instruções fase um.

Na Figura 4.12 e na Figura 4.13 podemos ver a continuação da simulação, onde é inserido também o comando 00010000010010100011001011111000, que corresponde a um comando de alteração de estado para fornecer, como pode ser verificado na Figura 4.13, na qual se pode ver o resultado deste comando, quer no registo interno quer na saída "action" comprovando que este também está a funcionar corretamente. Importa realçar que estes comandos de alteração de estado só funcionaram porque o bit 19 que indica se o comando é proveniente do utilizador está a um, o que significa que não é preciso fazer verificação de preços de alteração para o comando ser executado.

Uma vez mais, na Figura 4.13 pode-se ver um novo comando a ser inserido com o valor 00010000011000100011001011111000 e como é possível constatar o bit que indica que é um comando do utilizador não está a um, o que significa que o comando é proveniente de uma fonte que precisa de verificação de preço para alterar o funcionamento, como por exemplo a rede. Nesta mesma figura, pode observar-se que este comando não produziu nenhuma alteração, nem no registo nem na saída "action". Isto

acontece pois o sinal "Price_Check", que indica se houve uma comparação de preço que permita alteração por fontes externas, está a zero.

O comando a ser inserido a seguir é exatamente um comando de comparação de preço com o valor 01110001010101010000000000000000. A comparação é feita para o dia 000 (Segunda-Feira) e o valor para comparação é 10101010 (170) e uma vez que o preço de alteração é zero qualquer valor irá ser aceite, como se pode ver na figura com o sinal "Price_Check" a passar de zero para um.

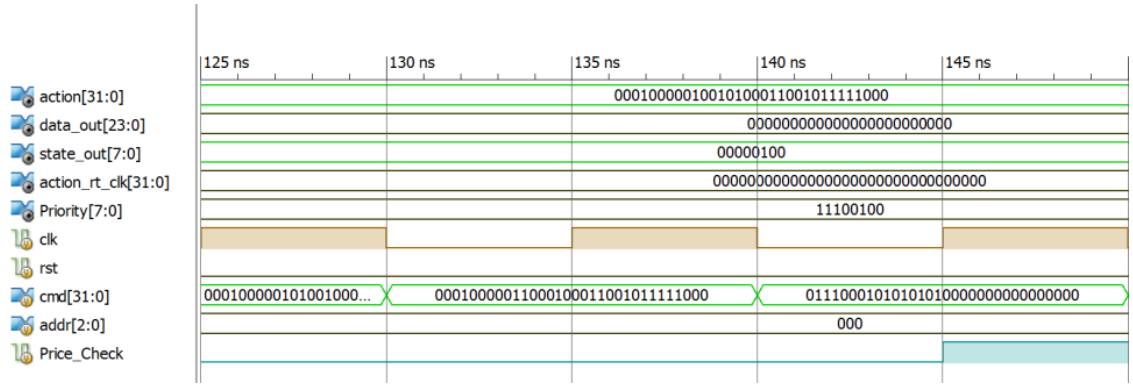


Figura 4.13 - Simulação do módulo de descodificação de instruções fase dois.

Já com o sinal de "Price_Check" a um, foi simulado o comando 00010000011100100011001011111000, que é um comando de alteração de estado vindo do utilizador. Na Figura 4.14 é possível confirmar que este comando é efetuado com sucesso e que o sinal de "Price_Check" não afeta os comando recebidos do utilizador. O próximo comando a ser simulado, 00010000100000100011001011111000, também é um comando de alteração de estado, mas este é proveniente de uma fonte externa (rede) já executado, uma vez que o sinal "Price_Check" está a um. O comando seguinte tem o valor de 00010000000010100011001011111000, sendo mais uma vez um comando proveniente do utilizador de forma a testar que os comandos provenientes desta fonte continuam a funcionar, mesmo depois da realização de um comando de fontes externas e que foi aprovado por uma comparação prévia de preço, sendo esta bem sucedida (Sinal de "Price_Check" a um).

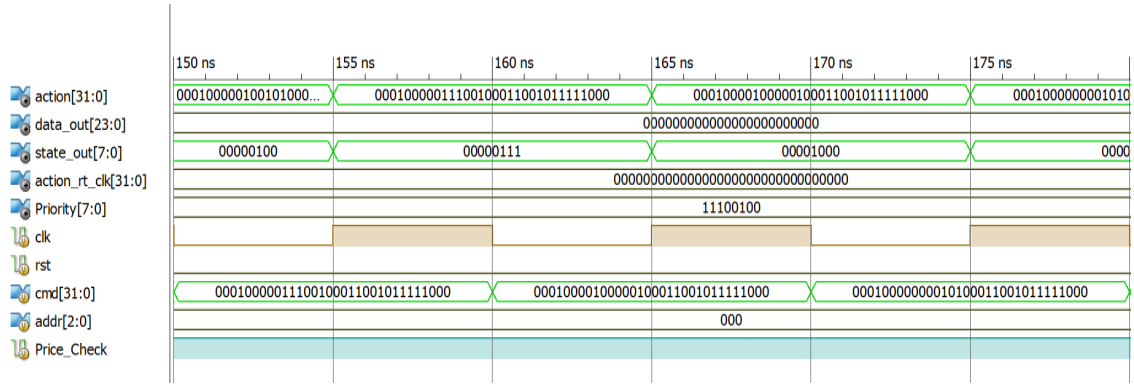


Figura 4.14 - Simulação do módulo de descodificação de instruções fase três.

O resultado produzido pelo comando 00010000000010100011001011111000 que pode ser observado na Figura 4.15 é uma alteração do registo de estado e da saída "action" como é pretendido. Também nesta figura é possível observar que este comando é seguido por um comando de alteração de preço com o valor de 01000001000000110000000000000000, em que o dia que o preço irá ser alterado corresponde ao valor 000 (Segunda-Feira) e o valor do preço 10000001 (129). Em seguida foi introduzido um comando de comparação de preço, desta vez com o valor 01110000010101010000000000000000. A comparação é feita para o dia 000 (Segunda-Feira) e o valor do preço a ser comparado é 00101010 (42). O resultado desta comparação é o esperado, isto é, como o valor a ser comparado é menor que o definido pelo utilizador, a comparação vai falhar e o sinal de "Price_Check" vai ser colocado a zero impossibilitando assim a execução de comandos de fontes externas até voltar a existir uma comparação de preço que seja aceite.

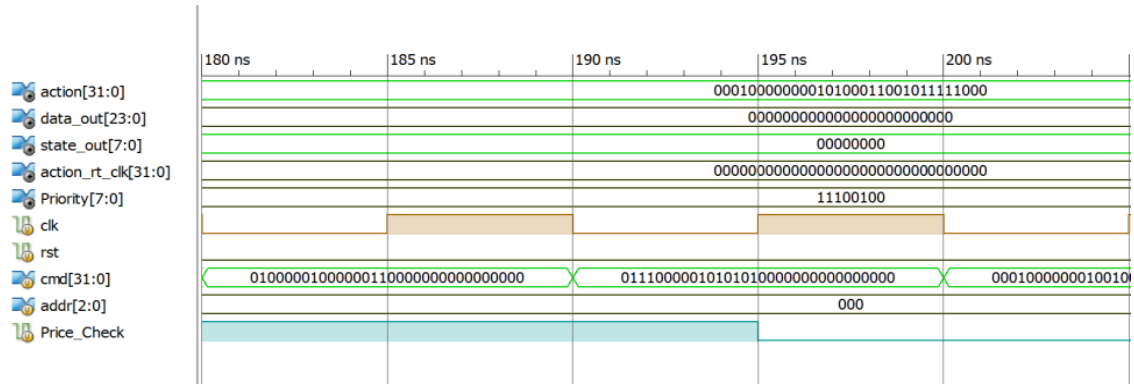


Figura 4.15 - Simulação do módulo de descodificação de instruções fase quatro.

Terminada a simulação do processo de execução de comandos com e sem verificação do preço de alteração, foi iniciado o processo de simulação dos restantes comandos que fazem parte da lista de instruções. Estes vão ser simulados de forma individual e em cada simulação vão ser mostrados os sinais, saídas e registos que o

respetivo comando vai afetar, por forma a poder concluir que o seu funcionamento é o pretendido e que ao executar cada comando as operações associadas a cada um são executadas de forma correta na sua totalidade.

O primeiro comando a ser testado desta forma foi o comando de alteração de prioridade, para isto foi inserido na entrada "cmd" um comando deste tipo com o valor 00100011011010000000000000000000, em que o novo valor do *byte* de prioridade é 00110110. Na Figura 4.16 pode ser observada a execução do comando e a alteração do registo "Priority", que contém o valor do *byte* das prioridades, para o valor que está no comando, o que verifica que a descodificação e execução do mesmo está correta.

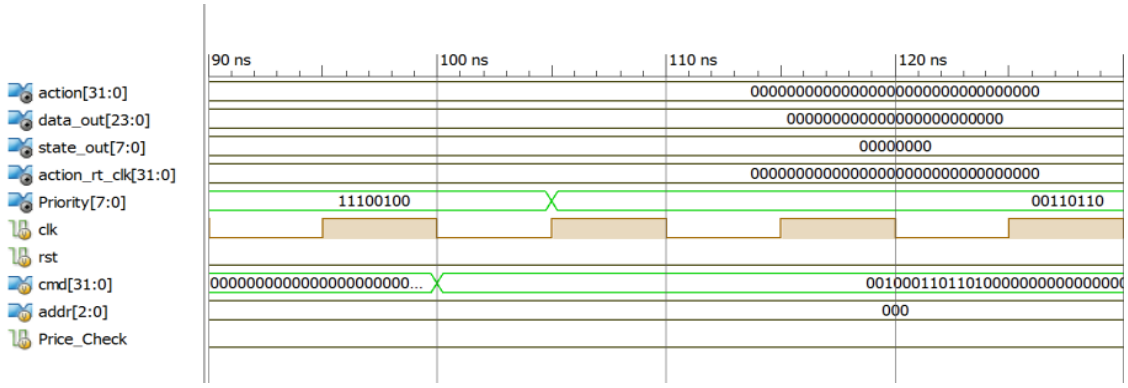


Figura 4.16 Simulação do módulo de descodificação de instruções fase cinco.

O comando seguinte a ser testado foi o de alteração do tipo de controlo. Na Figura 4.17 pode ser observada a simulação deste comando. Na entrada "cmd" foi inserido um comando deste tipo com o valor 01010000001110000000000000000000, onde o valor do novo tipo de controlo é 00000011. O valor do registo "Ctrl_Type", registo com o valor do tipo de controlo, é atualizado com este valor, o que demostra que a descodificação e execução do comando está correta.

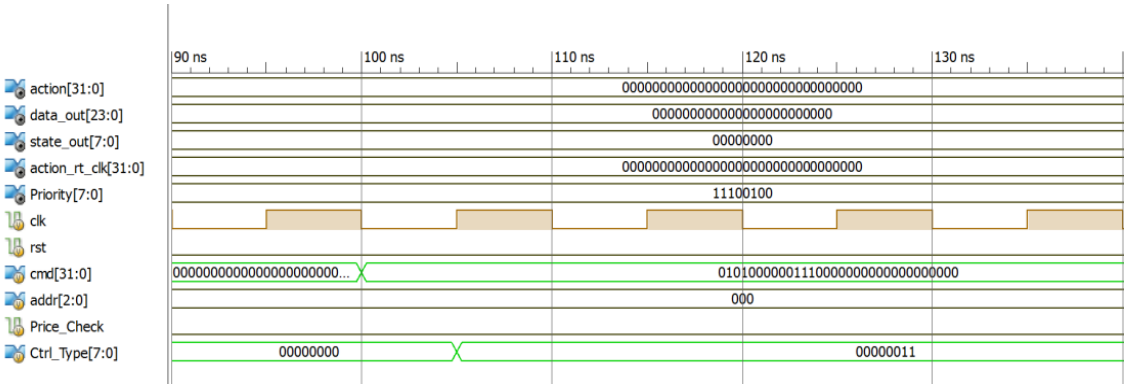


Figura 4.17 - Simulação do módulo de descodificação de instruções fase seis.

O próximo comando a ser simulado foi o de alteração de horário e na Figura 4.18 pode ser observado o resultado desta simulação. O valor do comando inserido é

00110001111110000000111111110000, em que o dia selecionado tem o valor de 000 (Segunda-Feira) e o valor da saída "data_out", que vai conter o horário de funcionamento para o dia selecionado, também é atualizado com o valor 11110000000011111110000, verificando assim o funcionamento deste ao constatar que coloca nas saídas as informações pretendidas.

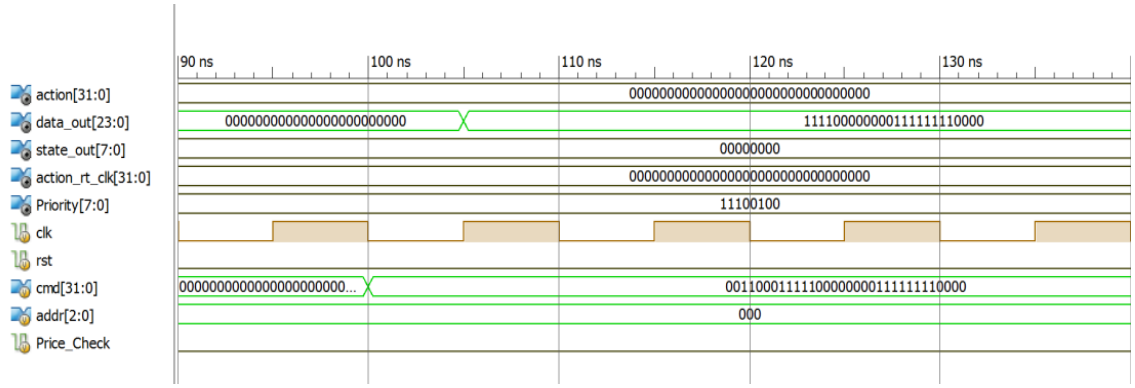


Figura 4.18 - Simulação do módulo de descodificação de instruções fase sete.

Seguidamente foi simulado o funcionamento do comando que define as horas, que tem como objetivo colocar na saída "action_rt_clk" este tipo de comando, de forma que este seja passado para o módulo do relógio. Na Figura 4.19 pode ver-se o comando deste tipo com o valor 10001110000100000110000000 a ser inserido na entrada "cmd", e é possível observar que na saída "action_rt_clk" está presente este mesmo comando como é pretendido, confirmando assim o correto funcionamento do mesmo.

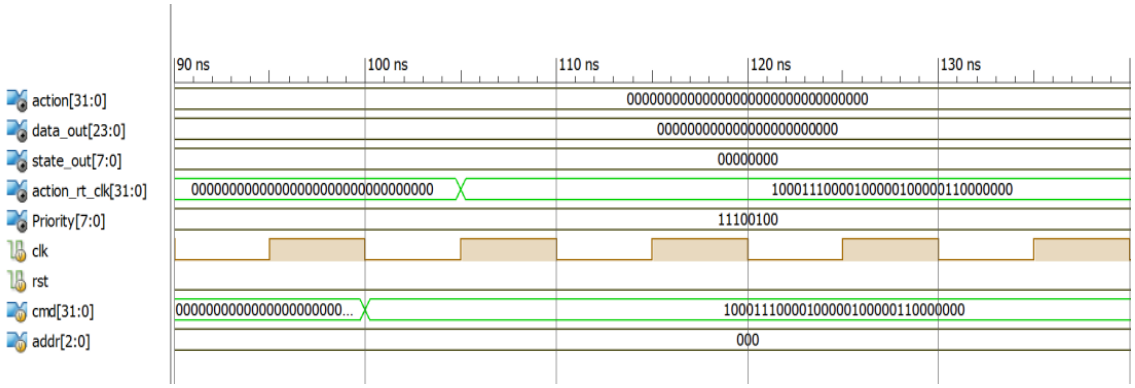


Figura 4.19 - Simulação do módulo de descodificação de instruções fase oito.

Na Figura 4.20 é possível ver a simulação do comando de alteração do estado de carga. O valor do comando inserido para esta simulação foi 10010100000110000000000000000000 e este possui como novo valor para o estado de carga 01000001, sendo possível observar nesta figura que é esse o valor que é colocado no registo "SoC", que guarda o valor do estado de carga, como é pretendido, verificando-se assim o correto funcionamento do mesmo.

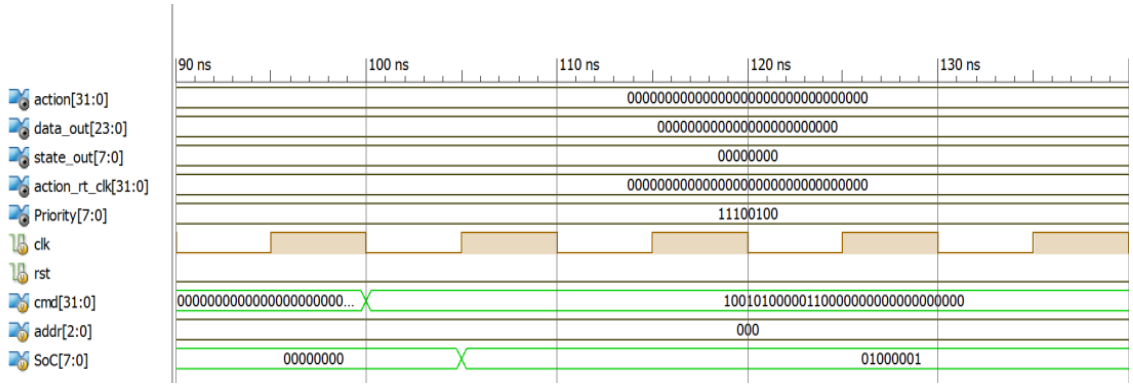


Figura 4.20 Simulação do módulo de descodificação de instruções fase nove.

Em seguida simulou-se o comando de alteração do estado de carga mínimo. Para isto, foi inserido o comando 10100100011110000000000000000000 que contém como novo valor para o estado de carga mínimo, 01000111. Na Figura 4.21 pode-se observar o resultado da simulação e conferir que é este o valor que se encontra no registo "SoC_min", registo que guarda o valor do estado de carga mínimo. Assim é possível verificar o seu correto funcionamento.

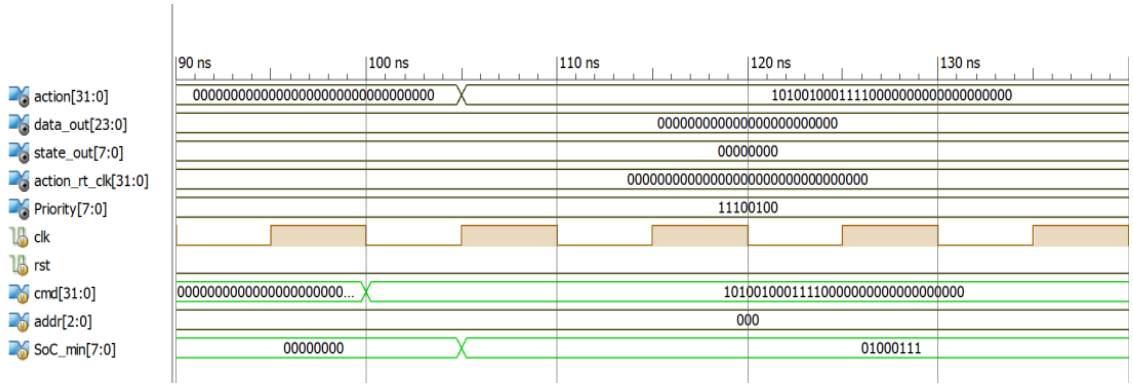


Figura 4.21 - Simulação do módulo de descodificação de instruções fase dez.

Os últimos comandos a ser simulados foram os dois comandos existentes para pedir informações a este módulo. Na Figura 4.22 é possível observar que são inseridos os comandos de pedido de informação um e dois. Pode-se observar também que as informações requeridas por estes comandos são colocadas na saída "action" de forma a serem enviados para quem requisitou esta informação.

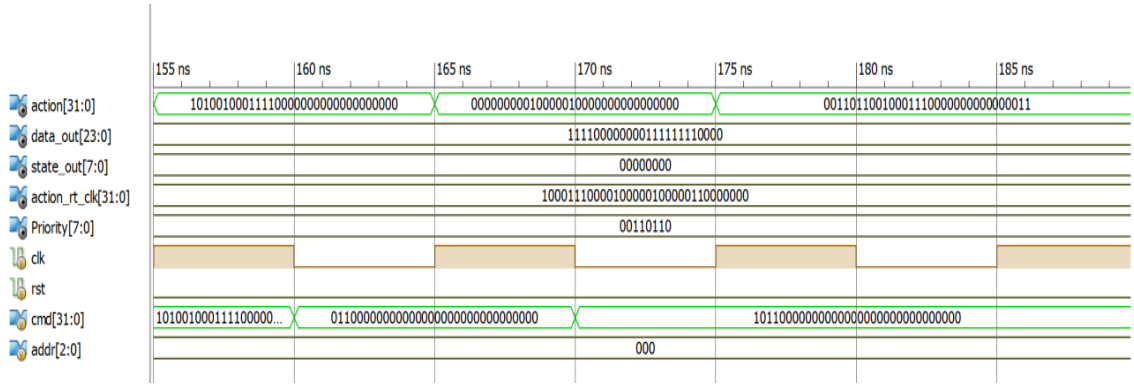


Figura 4.22 - Simulação do módulo de descodificação de instruções fase onze.

Com as simulações individuais de cada um dos comandos termina assim a simulação deste sub-módulo. Com os resultados obtidos em todas nas simulações anteriores, pode ser concluído que este está a funcionar de forma correta e por isso está pronto para ser integrado com o outro sub-módulo já simulado até ao momento.

4.5. Simulação do Sub-Módulo Responsável pelo Real Time Clock

O último sub-módulo a ser testado foi o responsável por fazer a contagem do tempo para o controlo das cargas. Como este módulo tem como objetivo contar o tempo real e como as unidades de tempo da simulação são de ordens muito inferiores a este, para realizar a simulação, o funcionamento do módulo foi alterado ligeiramente. A alteração está relacionada com o número de pulsos a contar para marcar a passagem de cada segundo, que foi substancialmente reduzido de forma a não prolongar demasiado o tempo de simulação. O número de pulsos utilizados nesta simulação para marcar a passagem dos segundos foi dez pulsos, para além disto o registo utilizado para armazenar o valor de contagem também foi reduzido para se enquadrar melhor com o valor de pulsos utilizados na simulação. No entanto, esta alteração não era a única solução, seria também possível manter o tamanho do registo e limitar apenas o número de contagens.

Na Figura 4.23 podemos verificar que ao contar dez pulsos de clock o registo com o valor atual dos segundos é incrementado. Após o incremento do valor dos segundos o registo pulse, responsável por armazenar o valor atual do número de pulsos decorridos, é posto a zero de forma que possa voltar a ser efetuada a contagem para o incremento dos segundos.

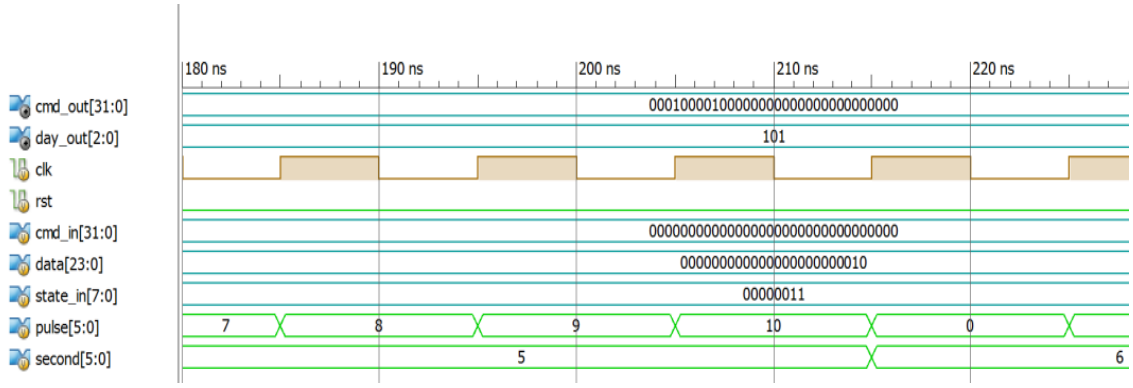


Figura 4.23 - Simulação do módulo RT_CLK fase um.

O mesmo procedimento pode ser observado na Figura 4.24, mas nesta situação o valor a ser incrementado é o do registo "second", referente aos segundos, como o próprio nome indica. Neste caso este é incrementado até ao valor 59 para corresponder ao número de segundos que um minuto tem. Quando isto acontece o registo "min", registo dos minutos, é incrementado e o registo "second" é posto a zero de forma a estar pronto para uma nova contagem.

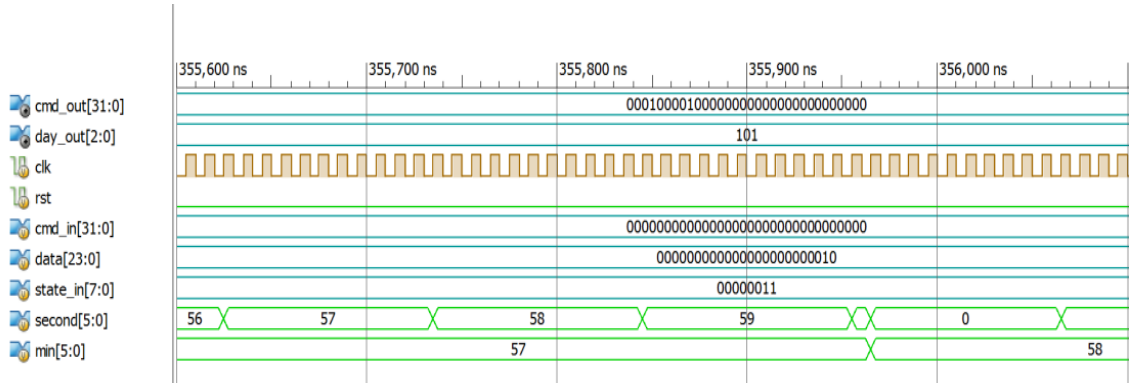


Figura 4.24 – Simulação do módulo RT_CLK fase dois.

O processo repete-se, mas desta vez com o registo dos minutos, como é possível observar na Figura 4.25, onde se pode ver o registo dos minutos a ser incrementado até ao valor 59. Uma vez alcançado este valor, o registo "hour", responsável por armazenar o valor da hora atual, é incrementado e o registo dos minutos é posto a zero para que este esteja pronto para uma nova contagem.

Nesta parte da simulação pode ser feita a verificação da funcionalidade deste módulo como controlador do horário de funcionamento das cargas. Na entrada "state_in" foi introduzido o valor do estado 00000011, que representa um dos estados em que a carga está em funcionamento, e na entrada data foi introduzido como horário de funcionamento da carga o seguinte valor 000000000000000000000010, que se traduz no funcionamento desta apenas da uma hora às duas da manhã. Neste ponto da simulação a hora atual é

cinco da manhã. Com base nestes valores é pretendido que o módulo ponha na saída "cmd_out" um comando para alterar o estado para um estado em que ela não esteja a funcionar, uma vez que na hora atual do sistema esta não deve estar operacional. Dado que o valor presente nesta saída é 00010000100000000000000000000000, verifica-se que o funcionamento está correto, visto que este comando é uma alteração de estado para parar a carga.

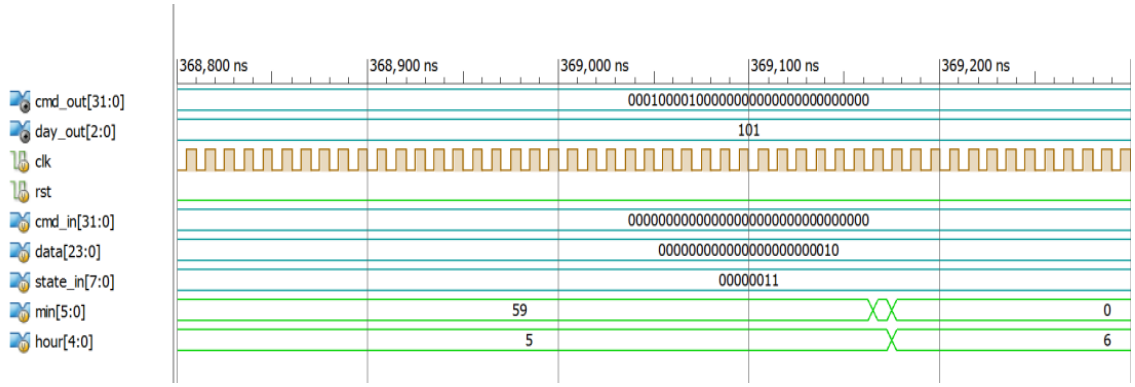


Figura 4.25 - Simulação do módulo RT_CLK fase três.

O incremento do registo das horas pode ser visto na Figura 4.26 e quando este valor é alcançado é feito o incremento do registo "day", registo com o valor dos dias que é colocado a zero de forma a estar pronto para uma nova contagem. Por sua vez, este registo é colocado a zero quando atinge o valor 7, o que também pode ser verificado nesta figura.

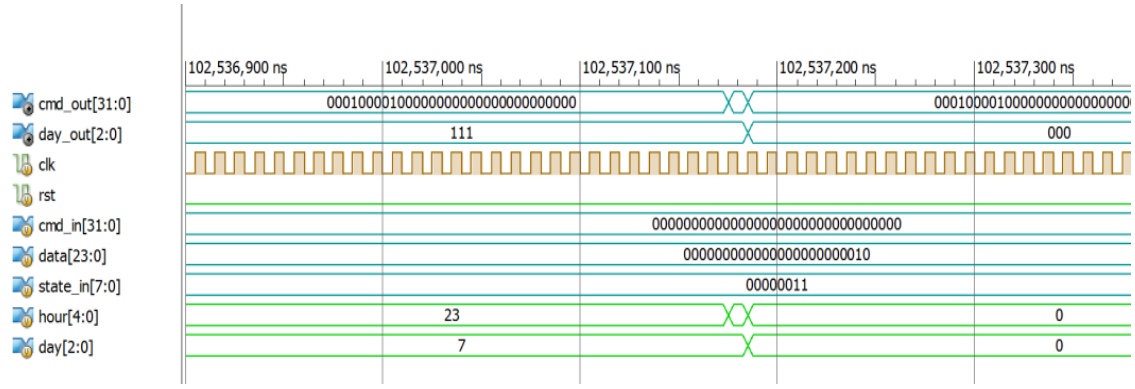


Figura 4.26 - Simulação do módulo RT_CLK fase quatro.

Terminada a verificação da contagem dos diferentes elementos associados à passagem do tempo e do controlo do horário de funcionamento das cargas, foi simulada a receção por parte deste módulo de um comando de alteração do horário atual, simulação esta que pode ser observada na Figura 4.27. Na entrada "cmd_in" foi introduzido o comando 10001010010100010000010110000000, que é o comando para a alteração do horário do módulo, que contém os novos valores a serem atribuídos aos diferentes

elementos. O valor para os segundos é 5, para os minutos 4, para as horas 5 e para os dias também 5. Através da figura constata-se que após a receção deste comando os valores dos respetivos registos são atualizados com novos valores, o que indica que esta funcionalidade está a implementada de forma correta.

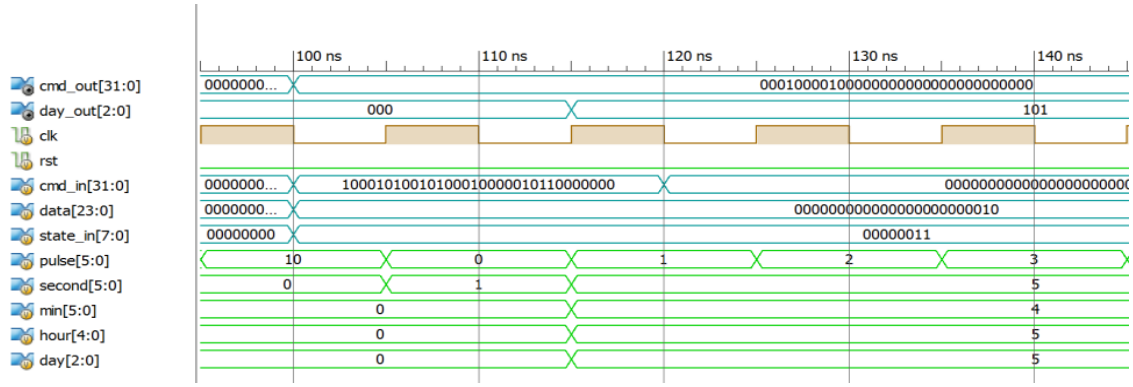


Figura 4.27 - Simulação do módulo RT_CLK fase cinco.

Assim é concluída a simulação do módulo do RT_Clk. Mediante os resultados obtidos é possível atestar o correto funcionamento deste e que disponibiliza de forma correta as informações que os outros módulos vão precisar para o seu funcionamento. Também foi verificado que este envia os comandos devidos para controlo do horário de funcionamento das cargas, podendo-se constatar que o módulo está pronto para ser ligado aos restantes numa fase mais avançada de simulação.

4.6. Simulação do módulo para controlo de cargas com todos os sub-módulos já integrados

Uma vez terminadas as simulações individuais de cada sub-módulo foi dado início à simulação do módulo completo. Para que fosse possível simular as interações destes e se o funcionamento geral do módulo é o pretendido. O processo de simulação do módulo na sua totalidade é semelhante ao realizado até agora com a introdução de comandos nas entradas que se destinam para esse efeito e são observados os sinais, saídas e registos internos de forma a verificar se estes têm sempre os valores corretos a cada momento, e assim validar o correto funcionamento do módulo em geral.

A primeira entrada a ser simulada foi a entrada “dev”. Nesta foi introduzido um comando de alteração do estado de carga com o valor 10010100000110000000000000000000, em que o novo valor do estado de carga é 01000001. Na Figura 4.28 pode se observar que este comando produz o efeito desejado ao alterar o registo “SoC”. Como este comando tem por objetivo apenas alterar o valor do estado de carga, que é uma variável interna do módulo, não se verifica nenhuma

alteração na saída "message", que contém a mensagem a ser enviada para o exterior. De seguida também nesta entrada foi introduzido um comando de alteração de estado com o valor de 00010000011010100011001011111000, em que o novo estado tem o valor de 01000001. Na figura também pode ser observado o resultado deste comando, o registo "State_c" passa a ter o novo valor e a saída "message" ou seja, já vez já é alterada porque o comando de alteração de estado vai produzir alterações fora do módulo que precisam de ser comunicadas, nomeadamente à carga.

Com estas simulações de integração e com as simulações individuais que foram feitas anteriormente, pode se concluir que esta entrada funciona bem para a totalidade do módulo.

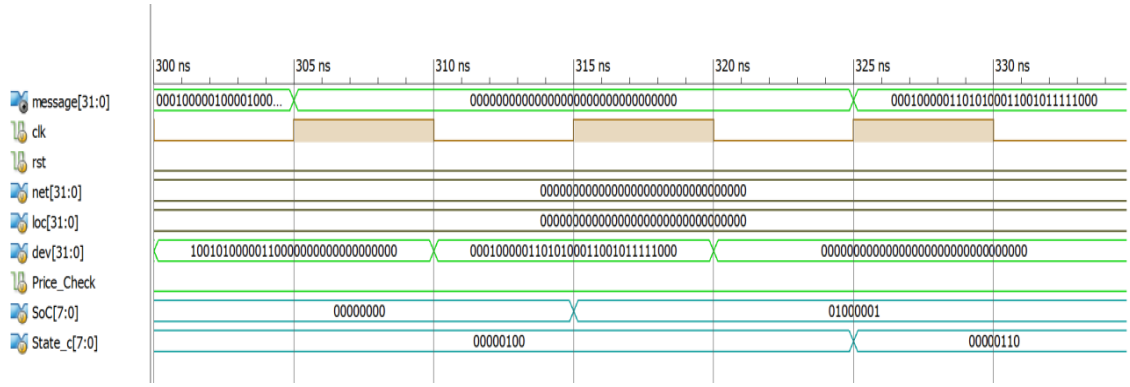


Figura 4.28 - Simulação do módulo completo - fase um.

Terminada a simulação da entrada "dev", foi dado início à simulação da entrada "loc". O primeiro comando a ser utilizado na simulação foi um comando de alteração de prioridade com o valor de 00100011011010000000000000000000, onde a nova prioridade tem o valor de 00110110. Na Figura 4.29 pode-se observar o resultado deste comando. Como esperado este produz a alteração desejada ao valor do registo de prioridade e como esta também é uma operação interna ao módulo a saída "message" não é alterada tal como é pretendido.

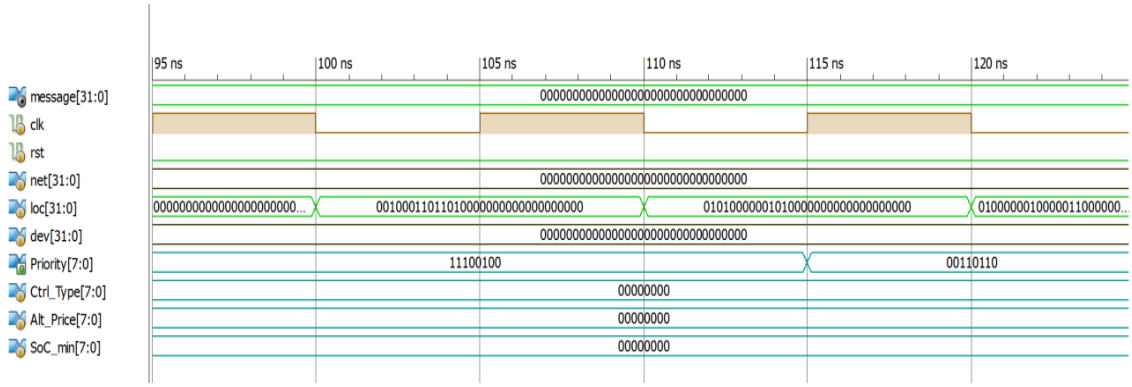


Figura 4.29 - Simulação do módulo completo - fase dois.

Na Figura 4.30 pode-se observar a continuação da simulação com a introdução de um comando e alteração do tipo de controlo com o valor de 01010000001010000000000000000000 que contém como valor para o novo tipo de controlo 00000010, que como é possível observar é passado para o registo "Ctrl_Type", tal como é pretendido. Depois deste comando foi introduzido um comando de modificação do preço de alteração com o valor de 01000000100000100000110000000000, em que o dia selecionado é o 000 e o novo valor para o preço é 01000001, sendo este valor passado corretamente para o registo "Alt_Price". Como ambos os comandos são operações internas ao módulo nenhum deles altera a saída "message", tal como pretendido.

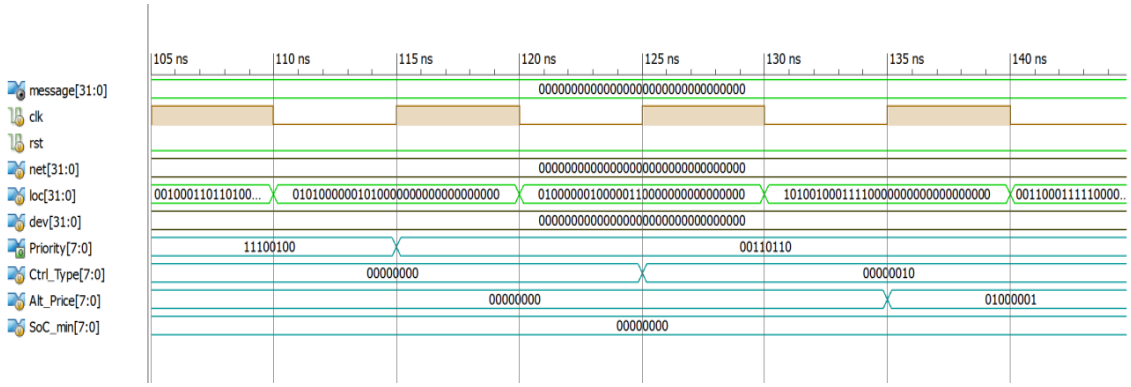


Figura 4.30 - Simulação do módulo completo - fase três.

A simulação continuou com a introdução de um comando de alteração do estado de carga mínimo, com o valor de 10100100011110000000000000000000, que possuí como novo valor para o estado de carga mínimo 01000111. Este valor pode ser observado como novo valor do registo "SoC_min" e o comando recebido é posto na saída "message", de forma a ser comunicado à carga que também é afetada por este tipo de comando. Estes resultados podem ser visualizados na Figura 4.31.

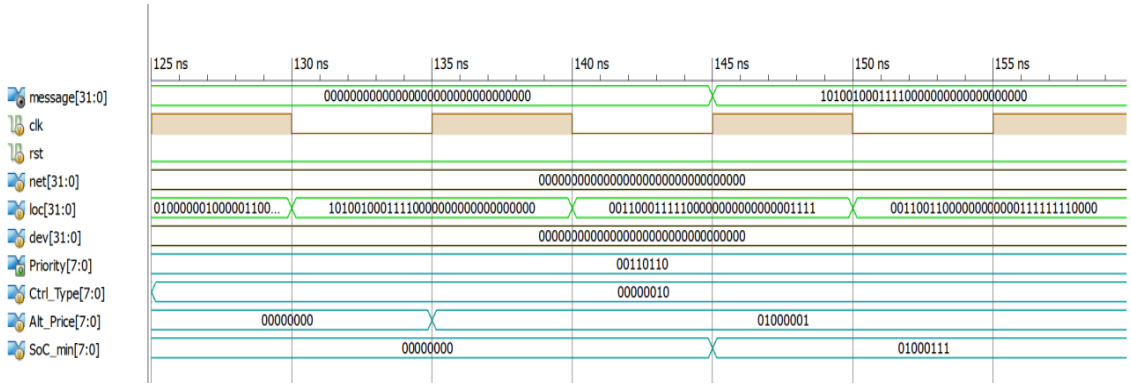


Figura 4.31 - Simulação do módulo completo - fase quatro.

O comando simulado a seguir foi o comando de alteração de horário. Na Figura 4.32 e Figura 4.33 pode-se observar uma sequência destes comandos, de forma a preencher por completo os horários de todos os dias. Como é possível constatar através da análise destas imagens, os valores vão sendo preenchidos consoante os dias que também são indicados nos comandos, o que verifica o funcionamento deste comando. Como acontece em outros comandos este também é um que apenas altera informação referente ao próprio módulo, logo não há alteração da saída "message" como é pretendido.

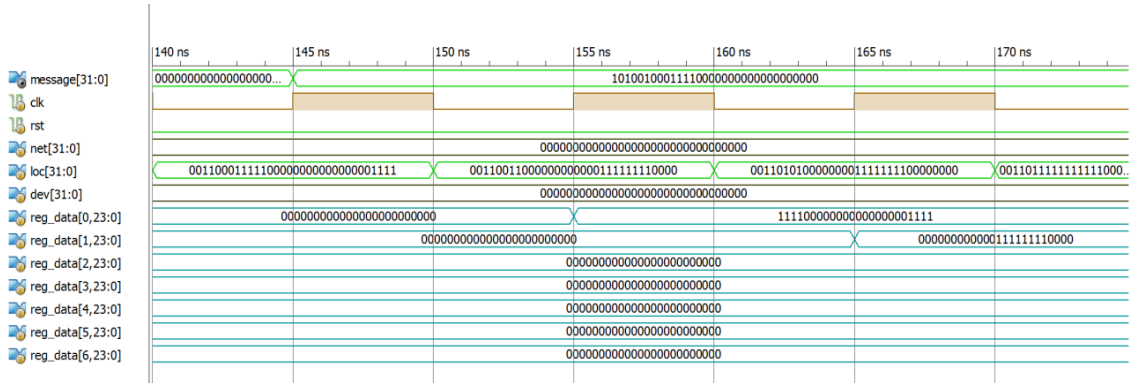


Figura 4.32 - Simulação do módulo completo - fase cinco.

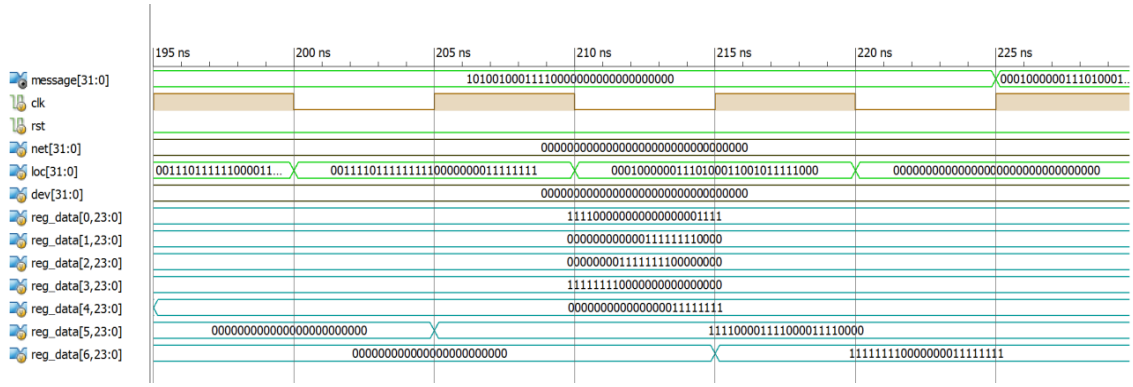


Figura 4.33 - Simulação do módulo completo - fase seis.

Por último na Figura 4.34 pode-se observar a simulação de um comando de alteração de estado, com o valor de 00010000001110100011001011111000, que contém como novo valor o estado 00000011. É possível observar a atualização, com este valor, do registo "State_c" e alteração da saída "message" para o valor do comando recebido. Esta execução de ambas as operações de forma correta, atesta o bom funcionamento deste comando.

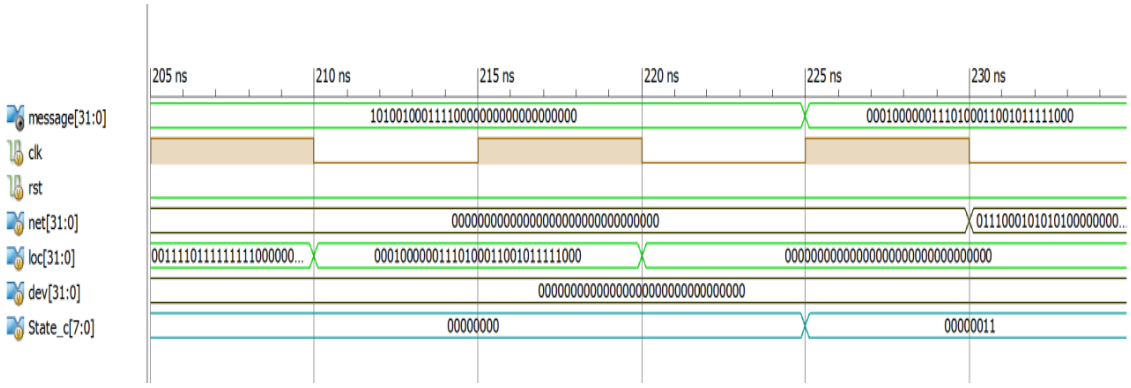


Figura 4.34 - Simulação do módulo completo - fase sete.

Com a simulação destas operações sobre a totalidade do módulo e as que foram feitas anteriormente nos sub-módulos, separadamente, é possível concluir que esta entrada está a funcionar corretamente. Portanto procedeu se à simulação da entrada "net", também esta sobre o módulo completo.

Para poder executar os comandos da entrada "net", o primeiro comando a ser simulado foi o de comparação de preço, com o valor 01110001010101000000000000000000 e tal como pode-se observar na Figura 4.35, isto resulta em uma aceitação do preço, pois o sinal "Price_Check" é colocado a um, o que permite verificar o funcionamento desta funcionalidade agora com o módulo completo e também que os próximos comandos provenientes desta entrada sejam executados. Ainda nesta figura é possível verificar a execução de um comando de alteração de horário com o valor 00111001111000011111111110000 e a alteração do registo correspondente ao dia indicado com o valor presente no comando, indo tudo isto de encontro ao que é pretendido.

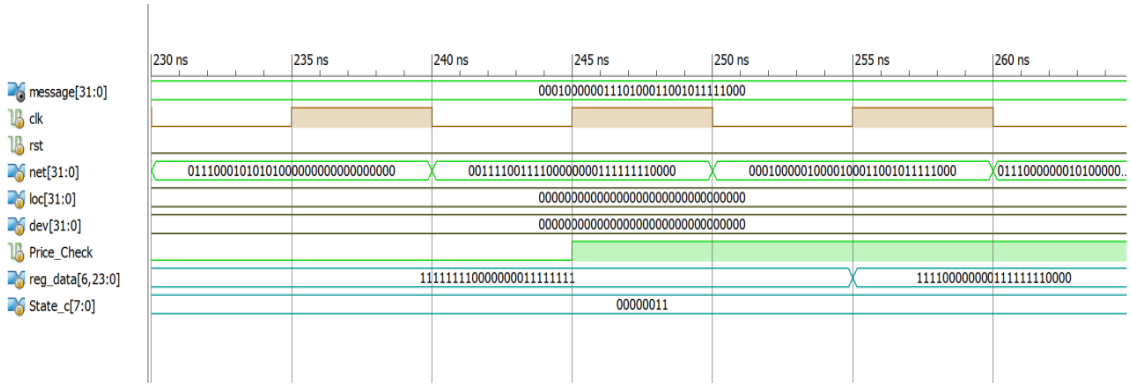


Figura 4.35 – Simulação do módulo completo - fase oito.

Na Figura 4.36 a simulação continua com a execução de um comando de alteração de estado, com o valor 00010000010000100010011011111000, que tem como novo estado o valor 00000100, valor este que pode ser visto a passar para o registo "State_c" como

pretendido. Mais uma vez como este comando também afeta a carga, este é colocado na saída "message" de forma a poder ser comunicado à mesma. O comando que se segue é outro comando de comparação de preço, com o valor de 01110000001010000000000000000, sendo o resultado deste comando o sinal de "Price_Check", que ao ser levado a zero diz-nos que esta comparação não foi aceite.

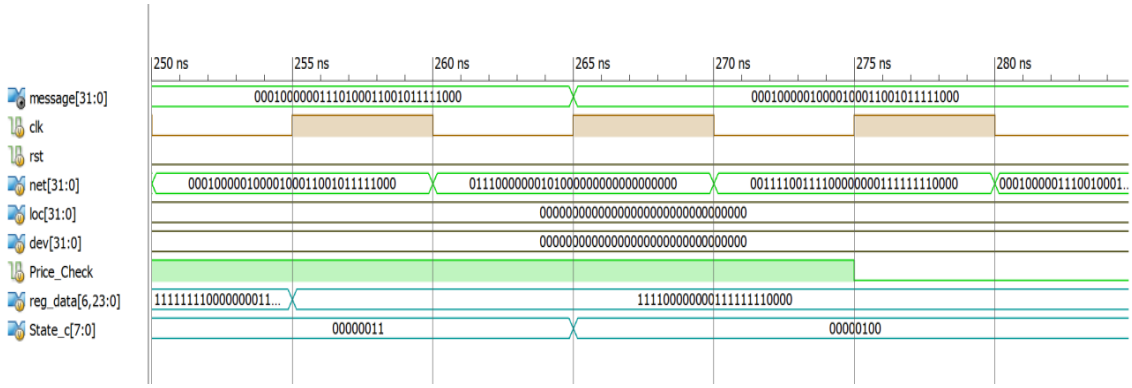


Figura 4.36 - Simulação do módulo completo - fase nove.

Como último comando a ser simulado nesta entrada foi escolhido um comando de alteração de estado com o valor 00010000011100100010010111111000. Novamente o sinal de "Price_Check" encontra-se a zero não obtendo o efeito pretendido. Estes resultados podem-se observar na Figura 4.37.

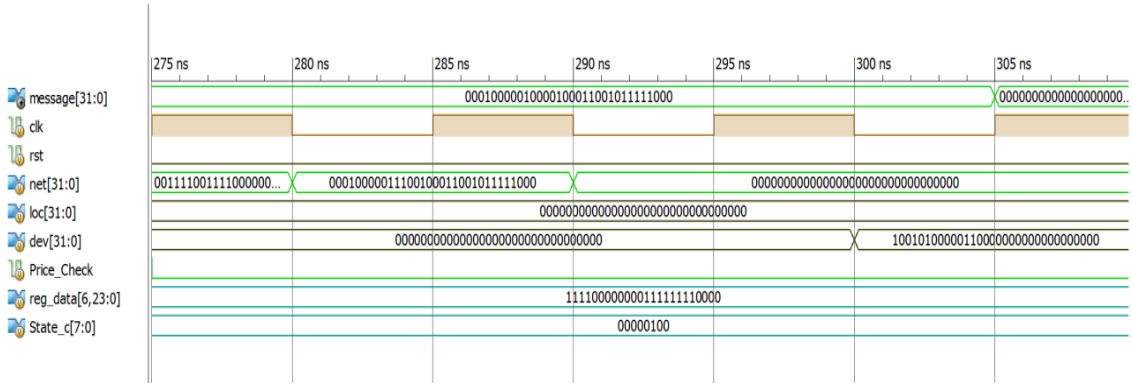


Figura 4.37 - Simulação do módulo completo - fase dez.

A correta execução destas instruções sob o módulo completo permite concluir que esta entrada encontra-se a funcionar de forma correta, e como tal procedeu-se à simulação da entrada em falta que é a entrada "rt_clk".

Na Figura 4.38 é possível observar a simulação de um comando de alteração de estado enviado pelo sub-módulo do relógio para esta entrada. O comando tem o valor de 00010000000011010010000110010000, sendo o valor do novo estado 00000001, sendo possível verificar que este valor é passado corretamente para o registo "State_c", tal como pretendido. Importante realçar que este comando é enviado automaticamente pelo

sub-módulo do relógio com base nas suas características de funcionamento e nas informações que lhe são fornecidas pelos outros sub-módulos. Assim sendo esta simulação é uma das que melhor demonstra o correto funcionamento do módulo como um todo.

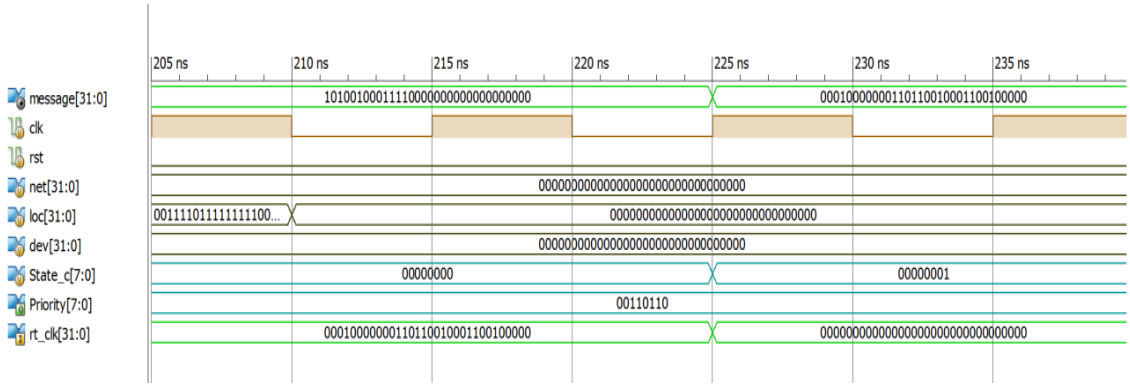


Figura 4.38 - Simulação do módulo completo - fase onze.

Com esta última simulação, pode-se dar por concluído o processo de simulação do módulo de gestão de cargas, através de simulações de funcionalidades individuais, passando pela simulação dos diferentes sub-módulos e culminando com a simulação do módulo completo. É possível concluir através dos resultados obtidos que o funcionamento deste vai de encontro ao pretendido.

4.7. Conclusão

Este capítulo abordou a simulação do Sistema de Gestão de Cargas (SGC) utilizando a ferramenta ISim, que faz parte do ISE da Xilinx tendo sido simulados todos os sub-módulos do SGC, com a exceção do módulo de comunicações uma vez que a simulação deste não traria vantagens devido ao facto que só é possivél verificar realmente a comunicação enviando e recebendo informação de forma assíncrona coisa que não é possível fazer na simulação.

O processo de simulação foi realizado desde as operações mais elementares até às mais elaboradas de forma a reduzir a complexidade das simulações e a isolar mais facilmente os erros que foram aparecendo durante o processo. Isto permitiu que a obtenção destes resultados fosse mais fácil e rápida.

Os resultados das simulações verificaram que o funcionamento dos módulos está de acordo com o que foi projetado estando estes pronto para serem implementados na FPGA.

Uma vez terminada esta fase de simulação foi dado início à fase de implementação propriamente dita., em que foram postas de lado as ferramentas de simulação e este

módulo do sistema e os restantes foram implementados devidamente nas plataformas adequadas.

CAPÍTULO 5

Implementação do Sistema de Gestão de Cargas para *Smart Grids*

5.1. Introdução

A implementação do Sistema de Gestão de Cargas para *Smart Grids* é abordada neste capítulo, sendo descritas todas as alterações e configurações feitas às ferramentas utilizadas para que fosse possível a realização deste trabalho. Estas alterações são divididas por cada uma das partes do sistema a que pertencem (aplicação, servidor e Sistema de Gestão de Cargas), uma vez que cada uma destas usa ferramentas bastante distintas.

Devido a esta diversidade e segmentação de ferramentas, as alterações necessárias para a implementação de uma parte do sistema não afetam nenhuma outra parte deste. No entanto, as configurações feitas para cada uma das partes são dependentes umas das outras, não sendo possível descartar nenhuma delas, havendo apenas uma única exceção, que consiste na instalação e configuração das ferramentas de acesso remoto ao servidor. Estas configurações não influenciam as restantes alterações efetuadas para a implementação do servidor e foram feitas por uma questão prática e de acessibilidade do servidor.

5.2. Implementação da Aplicação de Interface com o Utilizador

Para a implementação desta aplicação foi utilizado o IDE Android Studio, cujas características foram descritas no capítulo 3. A realização da aplicação foi efetuada para o *tablet* Asus 7 2013 com o Android Lollipop como sistema operativo instalado. Para tal foi preciso adaptar o IDE para esta configuração sendo por isso necessário selecionar como AVD a utilizar um que correspondesse a estas configurações. Para isto recorreu-se ao AVD manager que o Android Studio disponibiliza. Na Figura 5.1 é possível ver o ecrã de seleção do hardware que vai ser utilizado. Este disponibiliza uma lista predefinida de dispositivos para seleção e também informação sobre o tamanho do ecrã, a resolução e a

densidade de pixéis, sendo todas estas informações bastante relevantes uma vez que vão influenciar a organização e características dos componentes da aplicação.

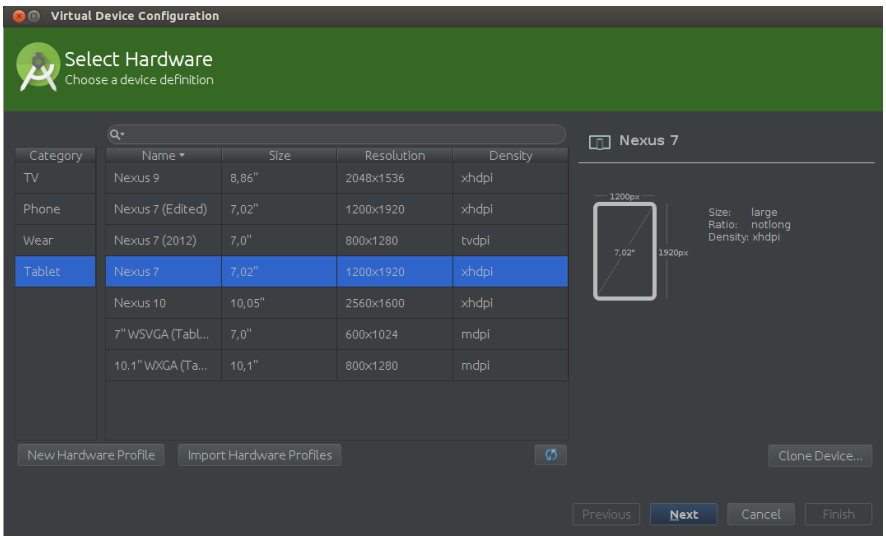


Figura 5.1 - Seleção do Hardware a utilizar.

Terminada a seleção do hardware, o passo seguinte é a seleção do sistema operativo a utilizar e a arquitetura do sistema a emular. Neste caso as informações fornecidas são relativas ao nome da versão do sistema operativo, o nível da API, a arquitetura a ser emulada e a versão do sistema operativo. Este ecrã de seleção pode ser observado na Figura 5.2.

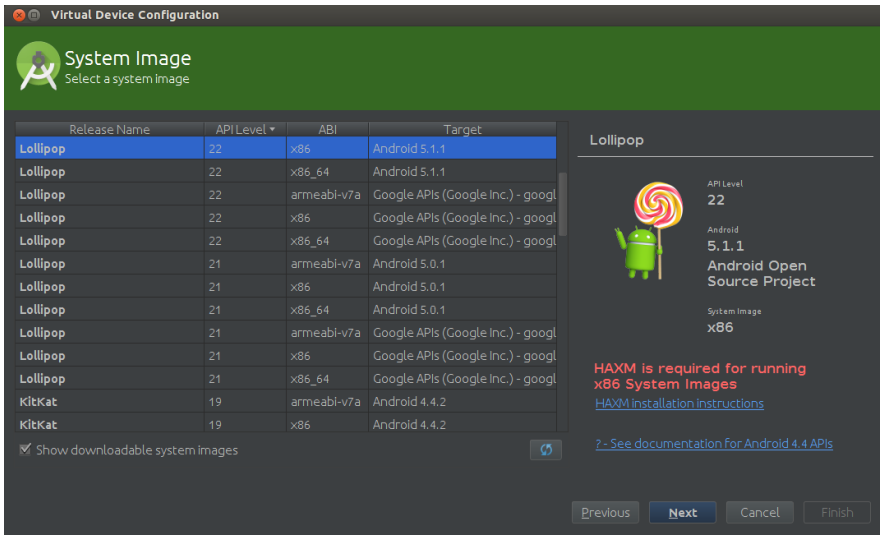


Figura 5.2 - Seleção da versão do sistema operativo.

Por fim, procedeu-se à verificação dos dados anteriormente selecionados. No entanto, este ecrã de seleção também permite a edição dos dados como a orientação em que se vai fazer a emulação da aplicação e se se pretende usar câmaras neste processo. No caso do *tablet* selecionado que possui duas câmaras, uma na frente do dispositivo e

outra atrás, aparece a opção para selecionar a câmara a utilizar na emulação. Uma opção será utilizar a *webcam* que vem integrada nos portáteis. No entanto como a aplicação a ser desenvolvida não vai utilizar captura de imagem nenhuma câmara foi selecionada.

Para além destas opções é também possível selecionar o tamanho da memória RAM, emular a existência de um cartão de memória e definir o seu tamanho e ajustar o tamanho da *heap* da máquina virtual. O ajuste destes parâmetros são importantes especialmente se os recursos do computador utilizado forem limitados.

É também possível a seleção para que a introdução dos dados seja feita pelo teclado, opção que foi selecionada para ser mais prática. No entanto continua a ser possível fazer a introdução de dados pelo teclado virtual que aparece na aplicação. Este ecrã de seleção pode ser observado na Figura 5.3.

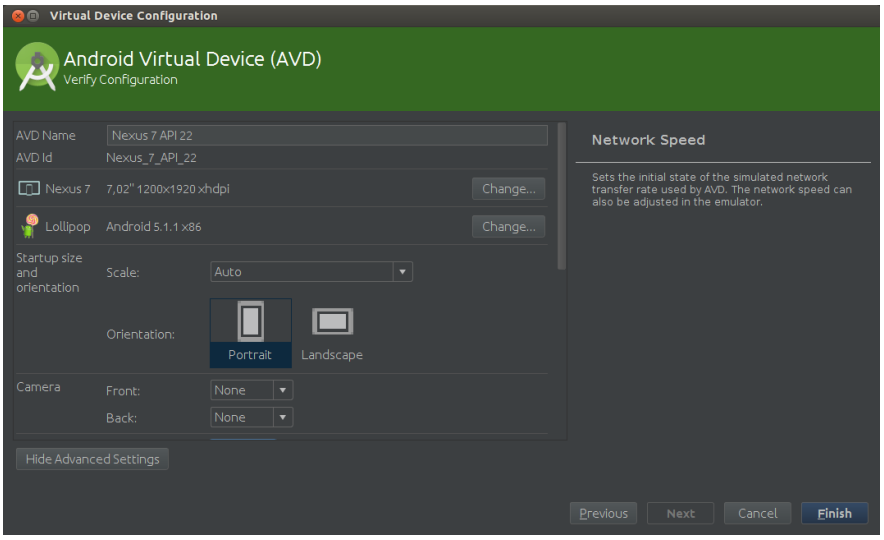


Figura 5.3 - Seleção das caraterísticas da máquina virtual.

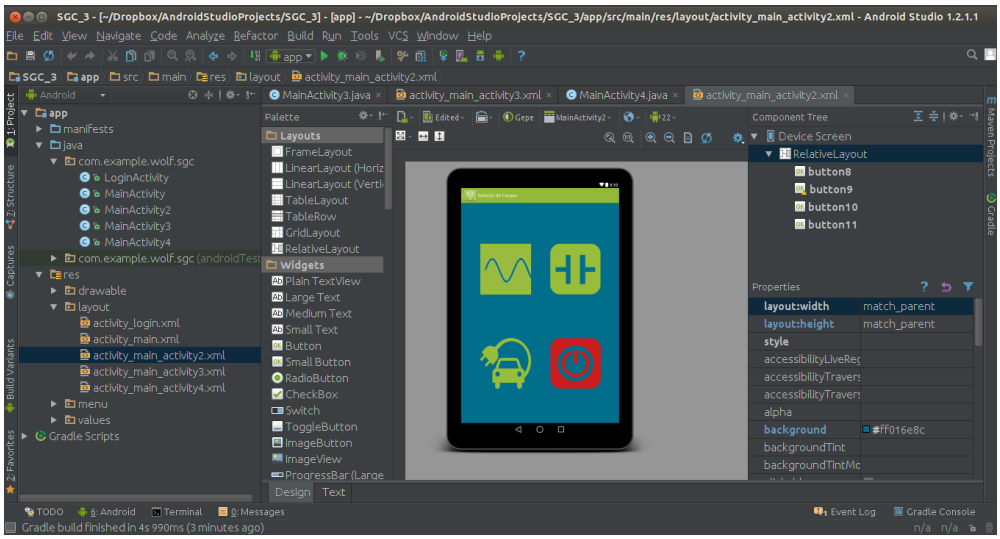


Figura 5.4 - Ecrã de desenho da aplicação.

Estas configurações são necessárias para que, durante o processo de implementação, se possa verificar em tempo real as alterações feitas ao aspeto da aplicação. Estas alterações podem ser vistas no modo de desenho da aplicação que está representado na Figura 5.4, onde é possível ver o aspeto da aplicação. Devido às configurações feitas anteriormente este traduz as dimensões e organização reais quando executado no *tablet*.

O aspeto da aplicação pode ser editado através do modo gráfico ou através da alteração do código XML (*eXtensible Markup Language*), utilizado para descrever a organização e dimensões dos elementos presentes em cada ecrã da aplicação. No modo de edição gráfico também é apresentada uma lista com os diversos elementos que podem ser adicionados à aplicação estando apenas disponíveis os que são válidos para o nível de API selecionado.

Foi também necessário fazer a atualização do SDK para que este esteja na sua versão mais recente de forma que se tenha acesso a todas as funcionalidades disponíveis. Isto é feito através do gestor do SDK que pode ser observado na Figura 5.5. Este separa os pacotes para instalação por nível de API para que se possa selecionar fazer o *download* dos mesmos para a API que se vai desenvolver.

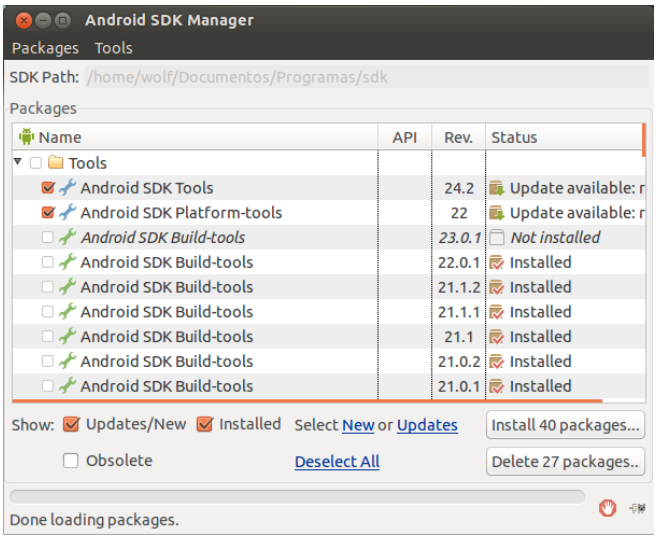


Figura 5.5 - Gestor do SDK.

5.3. Implementação do Servidor

O servidor foi implementado numa placa Raspberry Pi, com o sistema operativo Raspbian, tendo sido o primeiro passo da implementação desta parte do sistema o *download*, instalação e configuração do sistema operativo na placa.

A instalação do sistema operativo é feita no cartão de memória a ser utilizado. Existem várias formas de instalar o sistema operativo no cartão, no entanto a forma

selecionada foi a utilização do NOOBS (*New Out of the Box Software*), que é um gestor de instalação de sistemas operativos para a Raspberry Pi. Para se utilizar esta ferramenta basta fazer *download* dos ficheiros e copiar estes para o cartão de memória. Uma vez colocado o cartão na placa, quando esta é ligada, será lançado o gestor de instalação que vai solicitar a escolha do sistema operativo a instalar, a linguagem pretendida e a configuração do teclado a utilizar.

Os sistemas operativos para instalação podem estar disponíveis no próprio cartão ou através de *download* via Internet, sendo para isto necessário que no momento de instalação a placa já possua uma ligação à Internet. Na Figura 5.6 pode-se observar o aspeto deste assistente de instalação. Na frente de cada sistema operativo disponível é possível ver a indicação se este se encontra disponível localmente ou através da Internet. Uma vez selecionado o sistema operativo pretendido, neste caso o Raspbian, o gestor de instalação procede com a sua instalação. Terminada a instalação, a placa está pronta para arrancar com o sistema operativo.

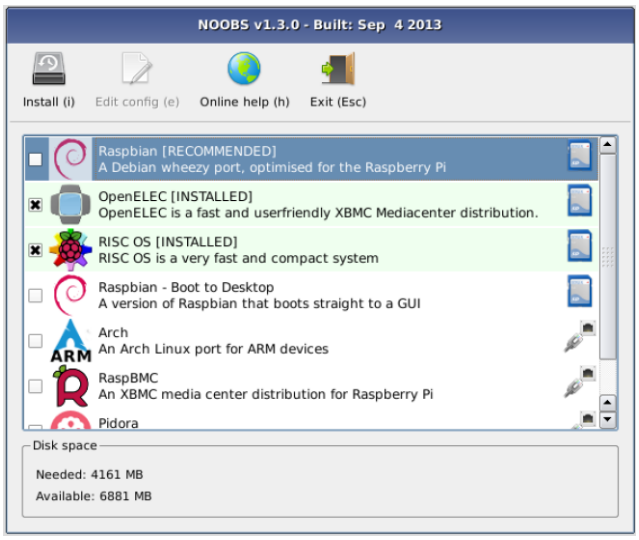


Figura 5.6 - Assistente de instalação NOOBS.

Seguidamente foi necessário fazer as configurações necessárias ao sistema operativo para obter as funcionalidades pretendidas para o servidor. Às configurações por defeito do sistema foram alterados em dois parâmetros. A primeira alteração, foi desativar a linha de comandos via porta série, uma vez que esta não vai ser utilizada e a placa só dispões de uma porta série que vai ser necessária na comunicação com o SGC. A outra alteração, foi a ativação do servidor SSH (*Secure Shell*) para se poder aceder à placa remotamente de forma a dispensar a necessidade de rato, teclado e monitor. Para isto foi utilizado o comando Raspi-config que lança a consola de configuração que é possível ver na Figura 5.7, sendo através desta que são feitas as alterações pretendidas.

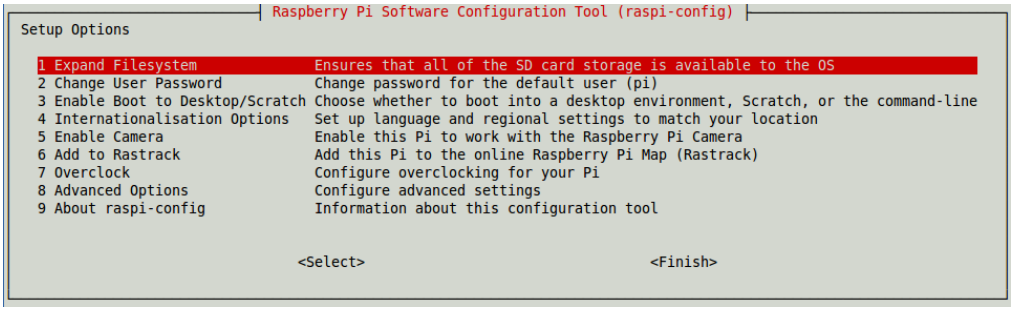


Figura 5.7 - Consola de configuração do Raspi-config.

No entanto o servidor SSH apenas permite interagir com a placa via linha de comandos, logo foi necessário também a instalação do TightVNC para se poder aceder remotamente ao ambiente gráfico. Para isto foi utilizado o comando "sudo apt-get install tightvncserver".

Depois de instalado o TightVNC, procedeu-se à sua configuração sendo utilizado para isto o comando "vncserver :0 –geometry 1366x768 –depth 24". Neste comando é possível observar que é definida a resolução pretendida para o ambiente de trabalho remoto, aqui com o valor de 1366x768, correspondente à resolução do PC utilizado para aceder a este. É também possível definir a profundidade de cores, neste caso com o valor de 24 bits.

Foi ainda necessária a instalação da aplicação VNC Viewer (Figura 5.8) para que se possa visualizar o ambiente gráfico através de qualquer computador, bastando para isso selecionar o endereço de IP (*Internet Protocol*) ao qual se pretende aceder e numa fase seguinte, introduzir a palavra-passe para aceder ao ambiente de trabalho remoto.

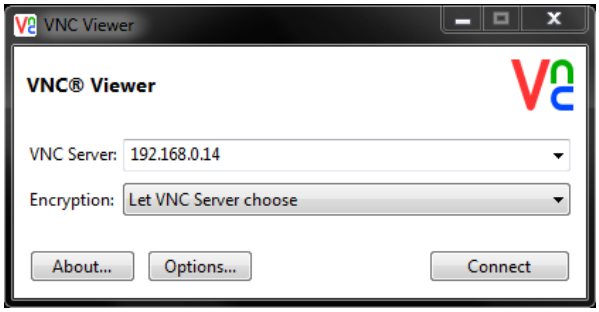


Figura 5.8 – *Login* no ambiente de trabalho remoto utilizando o VNC Viewer.

Terminada a instalação e configuração das ferramentas para acesso remoto à placa, foram instalados os componentes para o desenvolvimento da aplicação. Como o IDLE já vem instalado por defeito, apenas foi necessário instalar o módulo extra para a comunicação via porta série. Este módulo foi instalado utilizando o comando "sudo apt-get install python3-serial".

Foi também necessária a instalação e configuração do servidor de MySQL na placa, bem como a instalação da biblioteca para fazer a ligação com este através do Python. Para isto foi utilizado o comando "sudo apt-get install mysql-server python-mysqldb".

5.4. Implementação do Sistema de Gestão de Cargas

A implementação do SGC foi realizada através das ferramentas de desenvolvimento da Xillinx, mais especificamente o ISE Desing Suite. Na placa Basys, como já foi mencionado no capítulo três, esta utiliza a FPGA Spartan 3E, logo foi preciso configurar o IDE de desenvolvimento para esta FPGA. Isto pode ser efetuado no menu de definições de projeto, sendo também aqui que se define a linguagem de programação que se vai utilizar no mesmo, neste caso Verilog. Estas definições podem ser observadas na Figura 5.9.

Com esta definição de parâmetros também é possível ver através do IDE que percentagem dos recursos estão a ser utilizados, sendo esta uma funcionalidade útil para que ao longo da implementação se tenha noção dos recursos utilizados de forma a saber se a implementação do SGC não ocupa mais recursos dos que os que são disponibilizados pela FPGA escolhida, o que implicaria a utilização de outra FPGA ou um esforço adicional para otimizar a utilização de recursos desta. Na Figura 5.10 pode-se observar a informação disponibilizada pelo IDE sobre a utilização de recursos da FPGA, estando esta dividida por tipo de elementos lógicos existentes na própria.

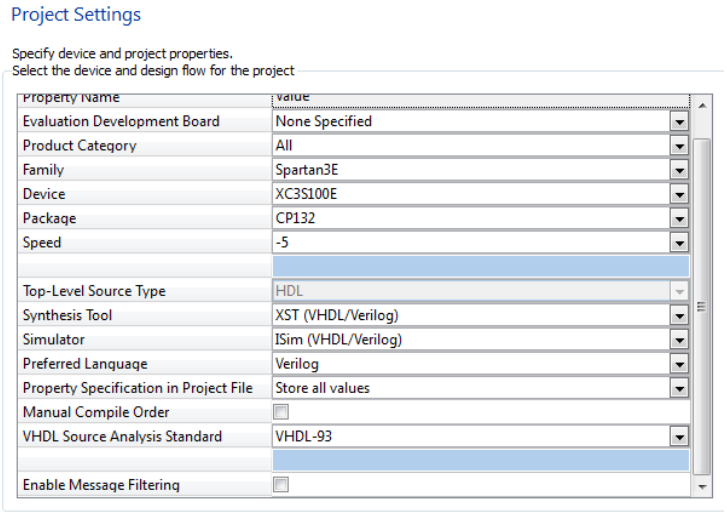


Figura 5.9 - Definições do projeto no ISE.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	548	960	57%
Number of Slice Flip Flops	621	1920	32%
Number of 4 input LUTs	869	1920	45%
Number of bonded IOBs	4	83	4%
Number of GCLKs	1	24	4%

Figura 5.10 - Utilização de recursos da FPGA.

Uma vez definidas as configurações do sistema e tendo os módulos sido implementados antes de os passar para a placa é necessário mapear as entradas e saídas na mesma. Para isto foi utilizada outra ferramenta disponibilizada por este IDE, com o nome de PlanAhead. Esta ferramenta permite associar as saídas e entradas dos módulos aos portos da FPGA correspondentes ao tipo da mesma, gerando depois o ficheiro UCF (*User Constraints File*) que vai ser utilizado na geração do *bitstream* que vai ser empregado na programação da placa. O aspeto desta ferramenta pode ser observado na Figura 5.11.

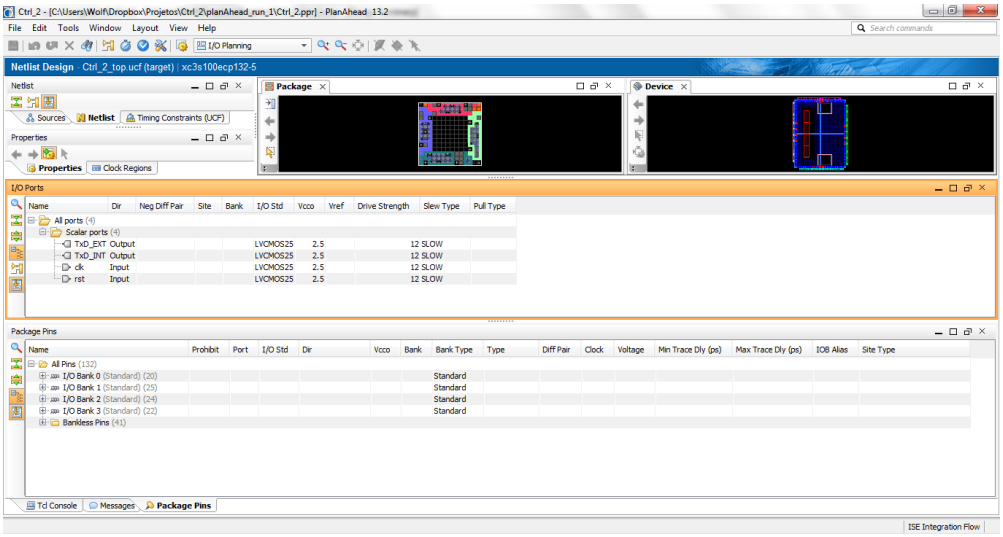


Figura 5.11 – Mapeamento das entradas e saídas utilizando o PlanAhead.

Depois de gerado o *bitstream* para ser programado na placa é necessário utilizar o software Adep da Digilent Inc (Figura 5.12), para poder passar este para a FPGA. Para realizar esta operação basta selecionar a FPGA que se pretende programar. De seguida, é necessário selecionar o ficheiro *bitstream* que se quer programar na FPGA ou na PROM (*Programmable Read-Only Memory*) que se encontra disponível na placa utilizada.

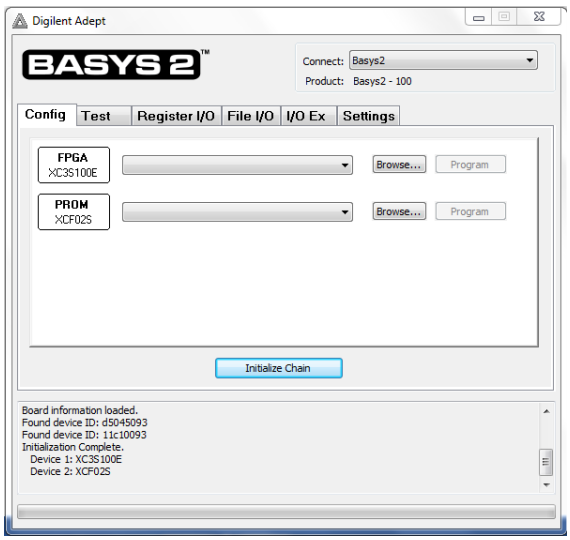


Figura 5.12 – Programação da FPGA utilizando o Diligent Adept.

5.5. Conclusão

O processo de implementação do Sistema de Gestão de Cargas para *Smart Grids* desenvolvido foi apresentado neste capítulo, sendo apresentadas todas as configurações e alterações necessárias para que a edificação do sistema com os parâmetros desejados seja possível.

O processo para a implementação da aplicação passou essencialmente pela configuração necessária do IDE, para que este utilizasse a versão do sistema operativo pretendida, tal como o hardware onde iria ser implementada a aplicação.

Para o servidor, o processo de implementação começou com a instalação e configuração do sistema operativo a utilizar na placa, seguindo-se a instalação e configuração de outros programas necessários para que esta pudesse funcionar como um servidor, destacando-se nesta área o servidor de base de dados. Foi também necessária a adição de novas bibliotecas às já existentes no Python instalado na placa.

A etapa de implementação do sistema de gestão de cargas passou essencialmente pela configuração do IDE utilizado na implementação e pela utilização e configuração de ferramentas adicionais, para que o código gerado possa ser programado corretamente na placa a utilizar.

CAPÍTULO 6

Teste do Sistema de Gestão de Cargas para *Smart Grids*

6.1. Introdução

Neste capítulo são apresentados os testes ao sistema desenvolvido, de forma a perceber se o seu funcionamento está realmente de acordo com o que foi projetado. Os testes numa primeira fase, foram realizados apenas com a aplicação e o servidor para ser mais fácil a deteção de possíveis erros e só terminados estes testes foram realizados os testes ao Sistema de Gestão de Cargas (SGC), sendo estes efetuados já com todos os elementos a funcionar em conjunto.

Os testes à aplicação incidiram sobre a verificação do aspeto desta, de forma a constatar que este está funcional e com o aspeto pretendido. Foram ao mesmo tempo, testadas as funcionalidades da aplicação, como a recolha e envio de informações tal como os testes às operações de *login* e *logout*. Para testar estas funcionalidades foi indispensável a utilização do servidor, no entanto nesta fase este não estará ligado ao SGC.

O servidor foi testado ao mesmo tempo que a aplicação de forma a verificar que este é capaz de responder aos pedidos que esta efetua, sendo também verificado nesta fase se este formata a informação corretamente para que esta seja enviada para o SGC.

Os testes ao SGC foram iniciados pelo teste ao módulo de comunicação de forma a filtrar possíveis erros na transmissão da informação para que estes não sejam confundidos com erros no funcionamento do SGC. De seguida foi testado o funcionamento do SGC no seu todo já com os comandos a serem enviados pela aplicação, funcionando isto como teste ao sistema completo.

6.2. Teste à Aplicação de Interface com o Utilizador

Os testes à implementação da aplicação foram iniciados com a verificação do aspeto da aplicação e a sua funcionalidade, se todos os elementos necessários estão a funcionar corretamente, se a navegação entre os diferentes ecrãs é feita da forma

pretendida e, por fim se a verificação dos dados introduzidos opera de forma a evitar a introdução de dados errados no sistema.

Estes testes foram realizados no *tablet* que foi escolhido para correr a aplicação, e os resultados destes são demonstrados através de capturas de ecrãs retiradas quando aplicação se encontra a correr no dispositivo. Para estes testes foi também necessário que o servidor estivesse ativo de forma a que fosse possível efetuar o *login* de um utilizador com o objetivo de ganhar acesso às funcionalidades da aplicação. Isto também permite que a aplicação estabeleça a ligação para o envio e receção de informação, sendo um dos efeitos visíveis a notificação ao utilizador se os comandos enviados são executados com sucesso ou não.

O primeiro passo nesta verificação foi o teste à aparência e funcionamento do ícone de lançamento da aplicação. Na Figura 6.1 pode ser observada a aparência deste no menu de aplicações do dispositivo em que é facilmente verificado que apresenta a imagem e o título da aplicação pretendidos. Após carregar no ícone de lançamento é lançado o ecrã inicial da aplicação, visível na Figura 6.2, este ecrã apresenta um tema personalizado que é constante em todos os ecrãs da aplicação.

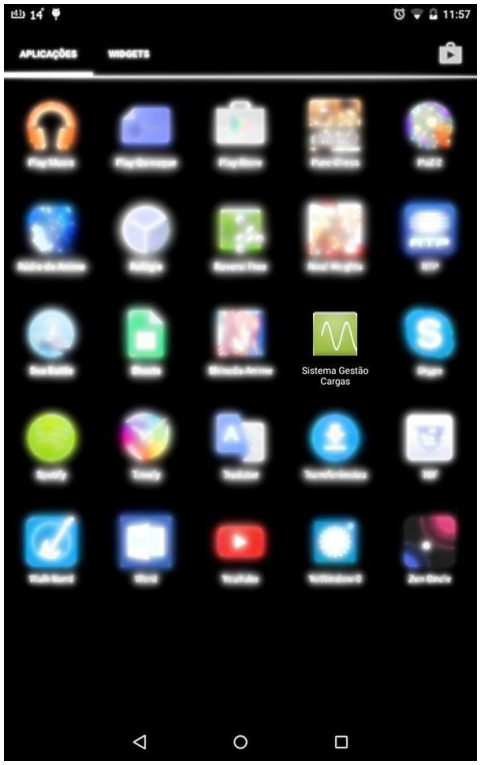


Figura 6.1 - Ícone da aplicação.

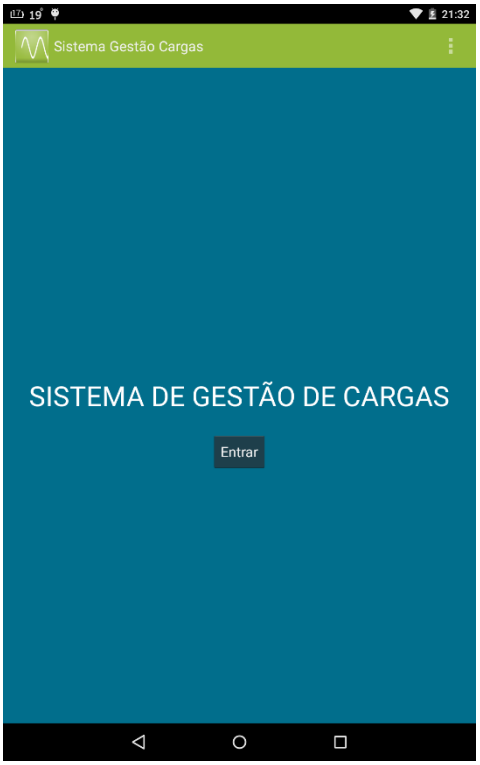


Figura 6.2 - Ecrã inicial.

Quando se pressiona o botão "Entrar", este faz com que seja lançado o ecrã seguinte, que é o ecrã de autenticação de utilizador, onde mediante a introdução dos dados de um utilizador válido, vai ser dado acesso às funcionalidades da aplicação.

O ecrã de autenticação possui duas caixas de texto, uma para o nome de utilizador e outra para a palavra-chave, e o botão "Entrar", cuja função é enviar os dados introduzidos para o servidor de forma que estes possam ser verificados com os dados existentes na base de dados dos utilizadores. O mesmo efeito pode ser atingido ao utilizar o botão "Entrar" existente no teclado e, todos estes elementos podem ser observados na Figura 6.3.

Caso as informações não estejam corretas, é dada a informação ao utilizador de que informação está errada. Como é possível ver na Figura 6.4, caso seja a informação da palavra-chave que esteja errada, este campo é selecionado e assinalado e uma mensagem aparece para informar o utilizador que o problema ocorrido com o processo de autenticação está relacionado com este campo. Caso o problema seja relacionado com o nome do utilizador inserido, o processo vai ser o mesmo, como é possível ver na Figura 6.5, onde este campo é selecionado e assinalado ao mesmo tempo que a mensagem de erro é mostrada ao utilizador. Desta forma o utilizador consegue estar a par dos erros que ocorrem durante este processo.

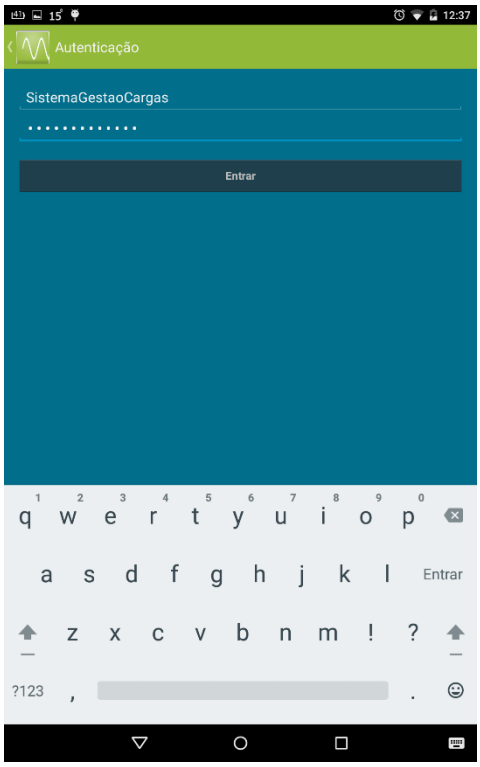


Figura 6.3 – Ecrã de autenticação.

Figura 6.4 – Mensagem de erro na palavra-chave.

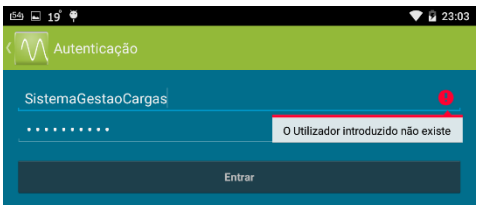


Figura 6.5 – Mensagem de erro no utilizador.

Caso de as informações se encontrem corretas, é dado acesso ao utilizador ao ecrã seguinte, que é o ecrã de seleção de cargas, onde o utilizador pode selecionar que tipo de carga pretende gerir ou então fazer *logout* da aplicação.

O ecrã de seleção de cargas é composto por quatro botões, três para selecionar as cargas a gerir e um para fazer o *logout* da aplicação, Figura 6.6. O botão no canto superior esquerdo representa os filtros ativos, o botão no canto superior direito os bancos de condensadores e o botão no canto inferior esquerdo representa os carros elétricos.

Na Figura 6.7, é possível visualizar a mensagem de confirmação de *logout* que aparece quando o botão para este efeito é pressionado. Caso o utilizador escolha a opção "não", a aplicação mantém-se no ecrã de seleção de cargas à espera que seja selecionada uma carga para gerir. Contudo se for selecionada a opção "sim", a aplicação volta ao ecrã de autenticação e a sessão do utilizador no servidor é encerrada, sendo necessário que este volte a introduzir os seus dados para poder aceder às funcionalidades da aplicação. Quando um dos botões das cargas é premido, o respetivo ecrã da carga é lançado enquanto o ecrã atual é encerrado.

Para demonstrar este funcionamento foi selecionado o ecrã da gestão do carro elétrico, uma vez que este é o ecrã mais elaborado e com mais elementos para a introdução de dados, sendo os ecrãs das restantes cargas baseados neste.

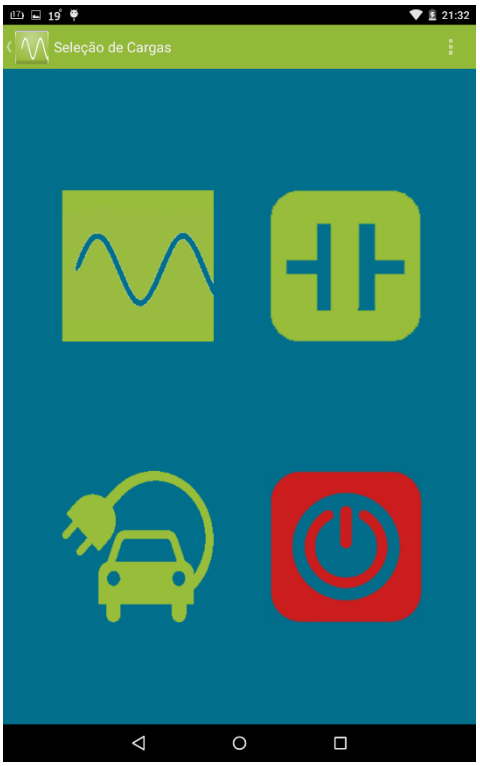


Figura 6.6 - Ecrã de seleção de cargas.

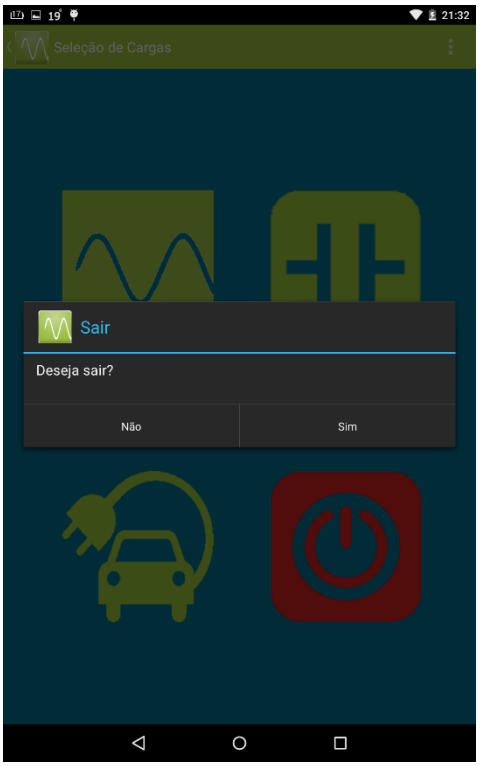


Figura 6.7 - Mensagem de confirmação de *logout*.

O ecrã da gestão do carro elétrico pode ser observado na Figura 6.10, e contem quatro *NumberPickers*, Figura 6.8, um para a capacidade de carga a utilizar, que vai estar limitado entre zero e cem, um para o estado de carga mínimo que também vai estar limitado entre zero e cem, um para o fator de potência que apresenta os mesmos limites

dos anteriores e um para o preço de alteração, este com limites entre zero e duzentos e cinquenta. Estes limites existem de forma a manter as informações introduzidas dentro do intervalo admissível para as respetivas variáveis, o que faz com que a validade esteja assegurada sem ser necessária nenhuma verificação extra. Os valores por defeitos destes elementos são os que se encontram a meio da sua gama de valores.

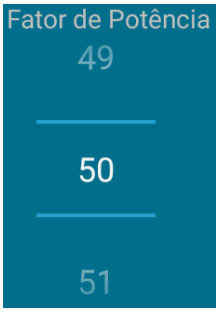


Figura 6.8 – Exemplo de *NumberPicker* utilizado para introdução de valores.

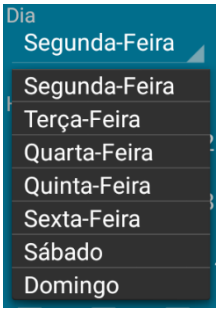


Figura 6.9 – Exemplo de *Spinner* utilizado para seleção de opções.

Existem também neste ecrã quatro *spinners*, Figura 6.9, com os valores referentes aos diferentes intervenientes que podem agir sobre o gestor de cargas. A ordem dos *spinners* determina a prioridade do interveniente selecionado, sendo o que se encontra mais acima o mais prioritário e o mais em baixo o menos prioritário. Esta informação está presente através de caixas de texto, juntas aos respetivos *spinners*.

Como os valores dos *spinners* já estão definidos, à partida é garantido que não são introduzidos valores não suportados pelo sistema, no entanto é necessário também garantir que não existe mais do que um *spinner* com o mesmo valor, uma vez que o mesmo interveniente não pode ter prioridades diferentes. Caso isto aconteça, é mostrado um diálogo ao utilizador assinalando esta ocorrência, de forma que esta seja possa ser corrigida, verificável na Figura 6.11.

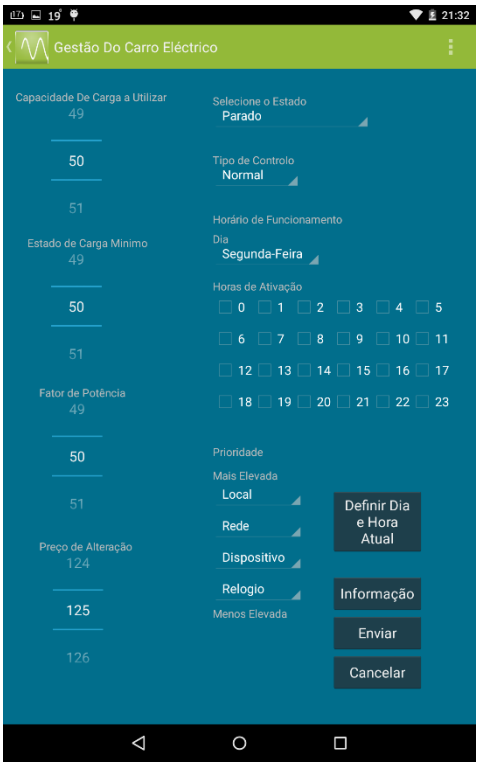


Figura 6.10 - Ecrã de gestão do carro elétrico.

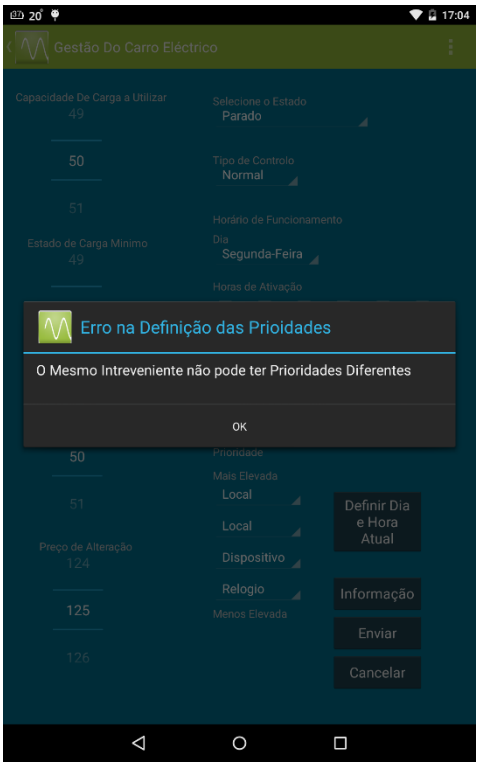


Figura 6.11 - Mensagem de erro na definição das prioridades.

Para além dos *spinners* para a definição de prioridades, existem também outros três, um com o valor a ser atribuído ao novo estado, um com o valor a ser atribuído ao novo tipo de controlo e, por fim, um para a seleção do dia da semana. Tal como no exemplo anterior os valores presentes nestes já estão previamente definidos, de forma a evitar a introdução de dados estranhos ao sistema. Para além destes elementos, existem vinte e quatro *CheckBoxes*, cada uma representando uma hora. Estes valores servem para indicar a que horas é que o utilizador pretende que a carga a ser controlada esteja ativa.

Em adição a estes elementos para a introdução de dados, existem quatro botões. O botão "Enviar", que vai fazer com que seja mostrado ao utilizador um diálogo contendo os diferentes tipos de comandos possíveis de ser executados após a seleção, o comando a ser enviado e irá ser formatado para conter as informações relevantes que foram previamente introduzidas pelo utilizador, Figura 6.12. Este comando lança outro diálogo, para notificar o utilizador se o seu comando foi efetuado com sucesso ou não, Figura 6.13. O sucesso da execução de cada comando é determinado com base na resposta do servidor a este mesmo comando. Por sua vez o botão "Cancelar" faz a aplicação regressar ao ecrã de seleção de cargas.

Figura 6.12 - Mensagem com a seleção do comando a enviar.

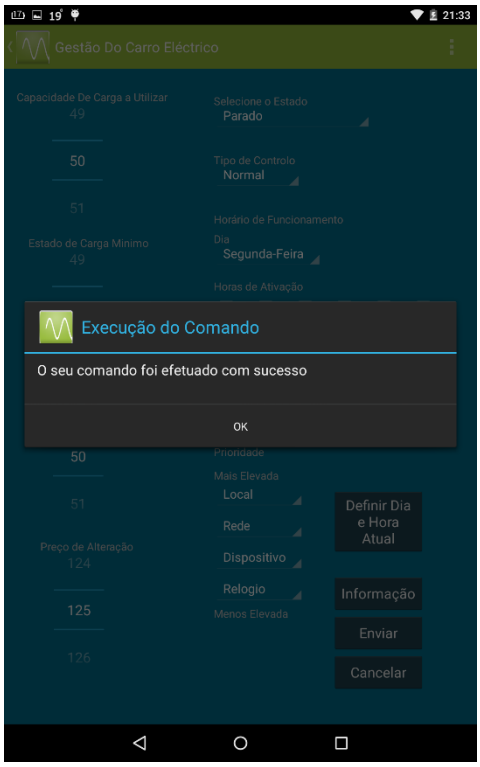


Figura 6.13 - Mensagem de notificação do estado da execução do comando.

Um botão "Informação" faz com que seja enviado um pedido de informação ao sistema de gestão de cargas. No retorno deste comando, os elementos acima referidos vão ser preenchidos com os valores mencionados de forma a mostrar ao utilizador a configuração atual do sistema de gestão de cargas, como é possível observar na Figura 6.14, onde os valores presentes nos elementos já são os recebidos através da execução deste comando.

O botão "Definir Dia e Hora Atual" faz com que a seja iniciado outro ecrã para a definição da hora e dia do sistema de gestão de cargas, o que pode ser observado na Figura 6.15. Este ecrã possui um *TimePiker* para que o utilizador possa escolher a hora que pretende atribuir ao sistema, de forma mais fácil e rápida e sem a possibilidade de haver informação a mais ou a menos a ser enviada. De forma a garantir a conformidade da informação enviada, este foi formatado para apresentar a seleção de horas no formato de vinte e quatro horas, tal como é pretendido para este caso. Para a seleção do dia, à semelhança do que é feito no ecrã anterior, foi utilizado um *spinner*. Existem também dois comandos, um para "Definir Dia e Horas" que faz com que seja enviado o comando para definir o dia e hora do sistema de gestão de cargas, esse já com os dados introduzidos pelo utilizador previamente e o botão "Cancelar", que faz com que a aplicação volte ao ecrã anterior (Gestão do Carro Elétrico).

Figura 6.14 – Novos valores dos elementos recebidos através do comando de informação.

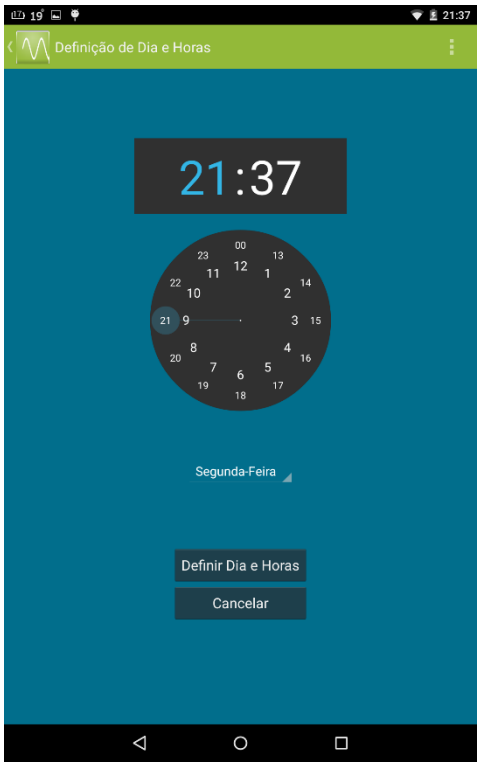


Figura 6.15 – Ecrã de definição de dia e hora.

6.3. Teste ao Servidor

Os testes à implementação do servidor que está alojado numa placa Raspberry Pi foram efetuados em simultâneo com o teste à aplicação de interface com o utilizador. Enquanto o ponto anterior documentou os resultados obtidos do teste à aplicação, agora serão documentados os resultados obtidos do lado do servidor.

Os testes foram iniciados com a verificação da operacionalidade do servidor MySQL instalado na placa. Para isto foi criada uma base de dados chamada USERS com apenas uma tabela, também ela chamada Users, para albergar a informação de acesso dos utilizadores da aplicação. Esta tabela terá apenas dois campos o nome do utilizador e a respetiva *password*, que pode ser observada na Figura 6.16.

O campo *password* encontra-se codificado com o método de codificação SHA1, sendo que os valores deste campo armazenados na base de dados são o resultado de encriptação das *passwords* de cada utilizador, utilizando este algoritmo de encriptação. Logo, os valores deste campo que se podem observar na figura não são as palavras-passe reais, mas sim a sua codificação segundo este algoritmo.

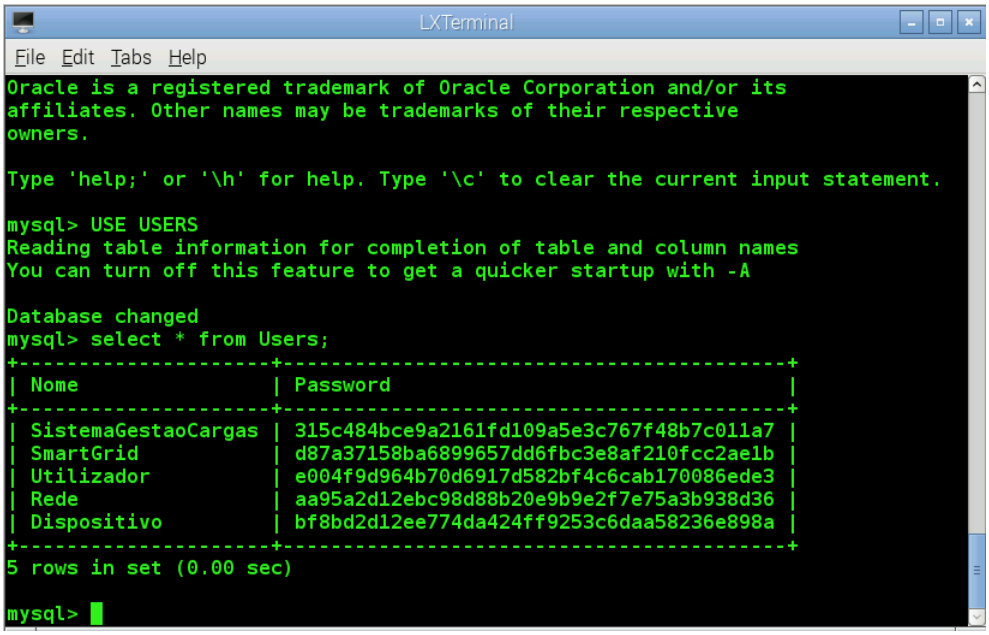


Figura 6.16 - Tabela dos utilizadores.

O processo de *login* pode ser observado na Figura 6.17. Foram efetuadas várias tentativas de *login* para ilustrar o que acontece no servidor nas diferentes situações que podem ocorrer. Para realizar o processo de *login* o servidor recebe uma trama com os dados necessários. A primeira trama pode ser observada na figura a seguir ao texto de iniciação do servidor e tem o valor "Login:Anonimo:12345", em que os diferentes valores estão separados por os dois pontos. O primeiro (Login), serve para indicar ao servidor que operação se pretende realizar, o segundo (Anonimo) indica ao servidor o nome do utilizador e por último (12345), indica o valor da palavra-chave introduzida para o processo de *login*.

Após recebida a trama, esta é separada nos seus diferentes elementos, para que em primeiro lugar possa ser analisado o tipo de instrução a realizar, sendo que no caso em questão trata-se de uma operação de *login*. O valor do nome recebido vai ser então utilizado para fazer uma pesquisa na base de dados. Neste primeiro caso, como se pode observar pela imagem, o retorno é feito quando não existe nenhum utilizador com o nome inserido, resultado que se encontra correto uma vez que, como é possível observar na Figura 6.16, não existe nenhum utilizador com o nome Anonimo na base de dados.

A segunda trama é semelhante à primeira, mas com a alteração do nome de utilizador inserido, que neste caso é SistemaGestãoCargas e, ao consultar a tabela com os utilizadores da Figura 6.16, pode observar-se que este valor já é um utilizador válido, no entanto a palavra passe recebida é a mesma.

No caso da verificação da palavra passe, esta antes de poder ser comparada com o valor da base de dados, tem de ser encriptada e só depois é que é feita a comparação. O

resultado da encriptação realizada sobre a palavra-chave recebida através da aplicação é 8cb2237d0679ca88db6464eac60da96345513964, logo é diferente do valor que se encontra na base de dados, o que resulta em uma falha no processo de *login* devido ao facto de que a palavra-chave introduzida encontrar-se errada, o que é possível observar na Figura 6.17. No terceiro caso mantiveram-se os dados do caso anterior, com a exceção da palavra-chave, desta vez resultando num *login* bem sucedido. Quando isto acontece o nome de utilizador é guardado numa lista para utilizadores ativos, para que seja possível saber que utilizadores se encontram ativos em cada determinado momento.

O processo de *logout* é feito de forma semelhante, mas neste caso a parte da trama que indica a operação a realizar tem o valor de Logout, de forma a indicar que é esta a operação a realizar e o efeito que este comando produz é a remoção do utilizador indicado da lista de utilizadores ativos. Como é possível verificar na Figura 6.17, o comando é reconhecido e efetuado corretamente.

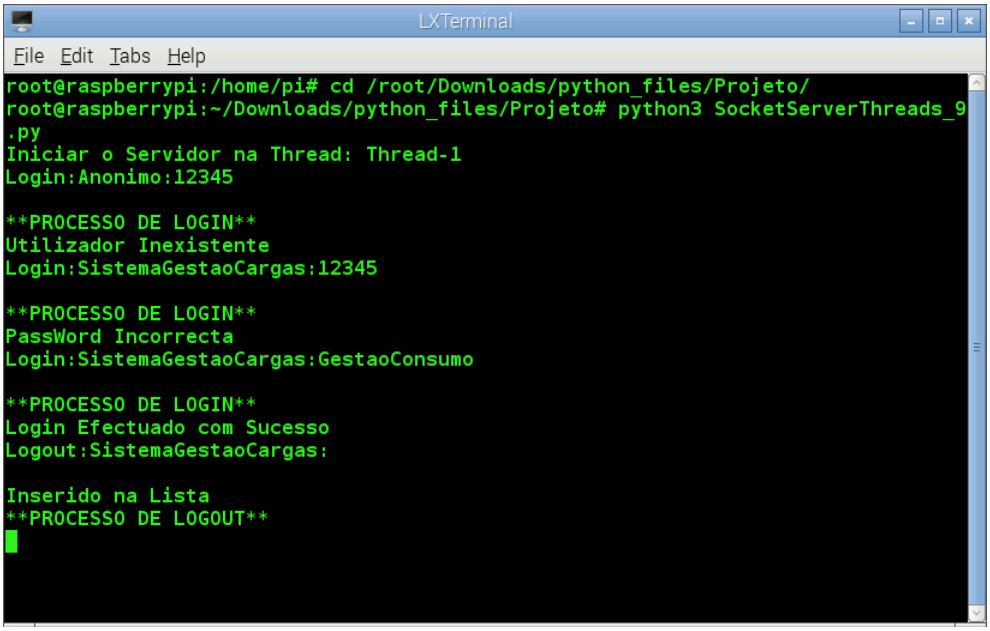


Figura 6.17 – Processo de *login* e *logout*.

Depois de testada a implementação do processo de *login* e *logout* do sistema foi dado início aos testes da receção dos comandos e o posterior envio destes por porta série. Na Figura 6.18, é possível ver a execução do comando de alteração de estado e do comando de alteração de prioridades.

O primeiro passo deste teste foi efetuar o *login* de um utilizador válido de modo a ter acesso ao envio de comandos. Uma vez efetuado o *login*, foi enviada através da aplicação, a trama que pode ser visualizada na figura logo a seguir à indicação que o *login* foi efetuado com sucesso.

A trama recebida contém o valor Send, que é o valor que vai indicar ao servidor que é para realizar uma operação de envio, seguido do nome de utilizador para que seja verificado se este tem uma sessão ativa ou não. O valor seguinte é o que vai indicar ao sistema de gestão de cargas o comando a executar. Este valor é indicado pelo utilizador, ao selecionar o tipo de comando a enviar na mensagem que existe para este efeito. Para os restantes valores foram usados os valores selecionados por defeito nos elementos existentes na aplicação para a inserção de dados para este comando.

Através da figura mencionada anteriormente, é possível verificar que o processo de envio é iniciado de forma correta e, uma vez iniciado, é dado lugar à formatação das informações recebidas de forma a estas serem enviadas ao SGC num formato que este seja capaz de reconhecer.

A formatação é feita consoante o comando selecionado para enviar, neste caso, observável na Figura 6.18, é o comando de alteração de estado. Os dados são formatados num inteiro de trinta e dois bits que, como se pode visualizar, tem o valor de 269062544, que é a representação inteira da formatação dos valores recebidos para um comando de alteração de estado válido para o SGC. Este valor é então, enviado por porta série, assinalado nesta figura pelo "A Escrever…", sendo esta mensagem seguida do valor a escrever e dos *bytes* a serem enviados.

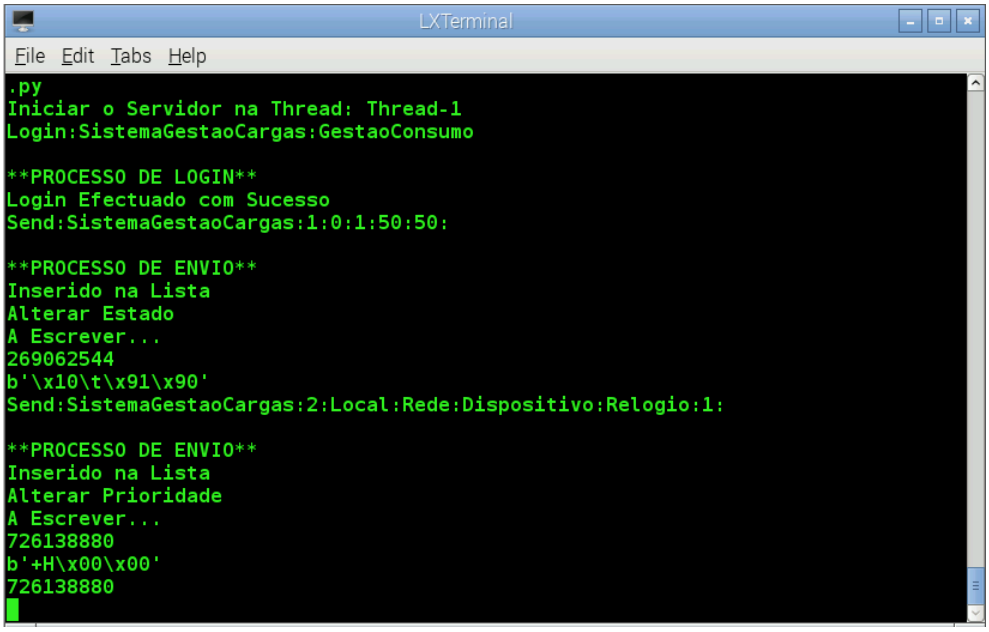


Figura 6.18 - Execução dos comandos alterar estado e alterar prioridade.

Na Figura 6.18 é também possível verificar a resposta do servidor mas, desta vez, a um pedido de envio de um comando de alteração de prioridade, sendo que os dois

primeiros elementos da trama recebida são os mesmos, uma vez que continua a ser uma operação de envio de comando e o utilizador não foi alterado.

O valor do comando a enviar ao SGC é diferente e, depois na trama, encontram-se os valores associados aos diferentes intervenientes do mais prioritário para o menos prioritário. Esta ordem de prioridades vai ser também formatada num inteiro de 32 bits que representa um comando válido para o SGC.

Os comandos que foram testados a seguir foram os comandos de alterar horário e alterar preço, sendo que os resultados destes testes podem ser observados na Figura 6.19. No caso do comando de alteração de horário, para além da informação referente à operação a realizar pelo servidor e ao utilizador que a solicitou, encontra-se na trama a indicação do comando a realizar pelo SGC, sendo estas informações genéricas para todos os comandos.

As informações presentes nesta trama referentes especificamente a este comando, são o valor zero, que indica o dia da semana cujo horário vai ser alterado e também os 24 bits que indicam em que horas se pretende que a carga esteja em funcionamento. Esta informação é também formatada para um inteiro de 32 bits de forma a ser enviada para o SGC. Este processo pode ser visualizado na primeira parte da Figura 6.19.

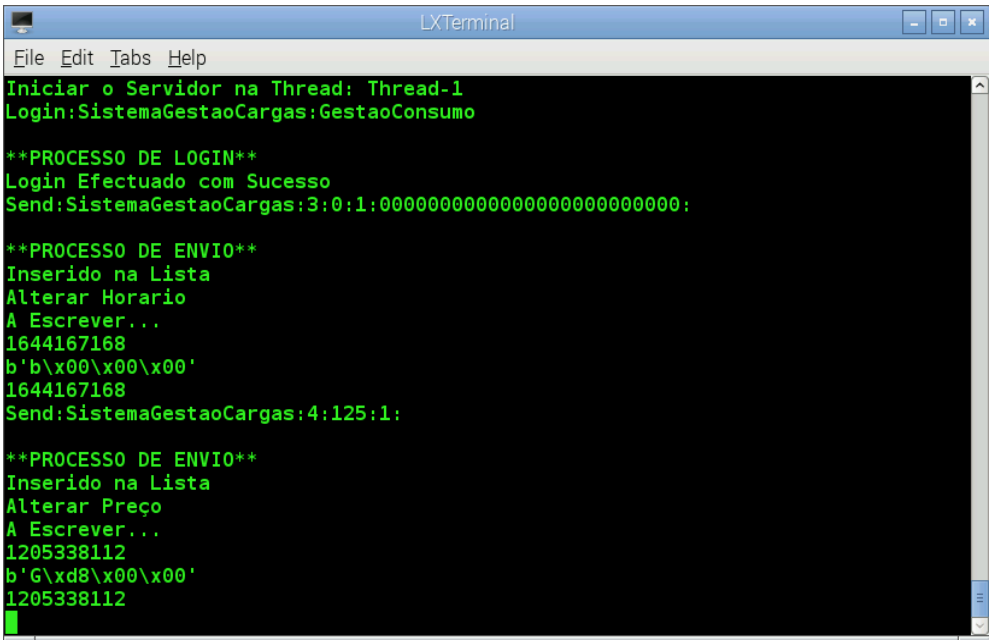


Figura 6.19 - Execução dos comandos alterar horário e alterar preço.

Na segunda metade desta figura, é possível visualizar a trama recebida para o envio do comando de alteração de preço, para além das informações genéricas que são comuns a todos os comandos e da indicação do comando a realizar pelo SGC. A informação presente na trama que é referente apenas a este comando, é o valor do novo

preço de alteração que neste caso é o valor 125, sendo então todas as informações relevantes formatadas para um valor inteiro que vai ser enviado para o SGC.

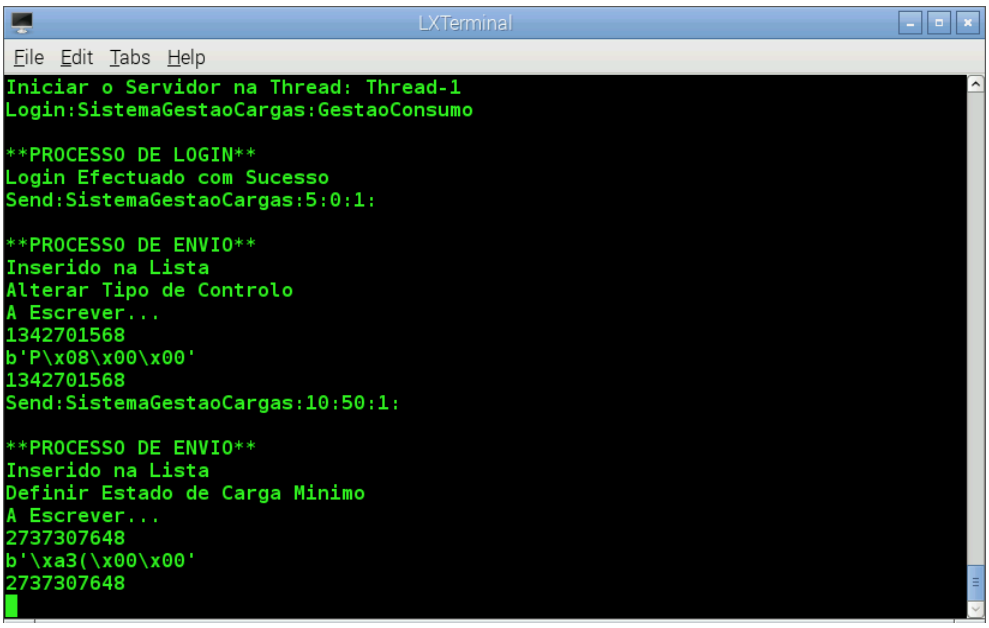


Figura 6.20 - Execução dos comandos alterar tipo de controlo e alterar estado de carga mínimo.

Seguiu-se o teste aos comandos de alteração do tipo de controlo e de definição do estado de carga mínimo. Os resultados da execução destes testes sobre o servidor podem ser observados na Figura 6.20. A trama recebida pelo comando de alteração de estado é semelhante à que foi recebida pelos comandos que foram testados anteriormente, sendo o único valor específico deste comando o valor zero, que é indicativo do novo tipo de controlo a aplicar ao SGC. É possível observar na figura que os dados a enviar são corretamente formatados para um inteiro e que este é posteriormente enviado pela porta série.

Através da segunda metade da Figura 6.20, é possível visualizar a execução do comando de definição do estado de carga mínimo, execução que segue os mesmos passos dos comandos anteriores. A trama recebida é, também, parecida à dos outros comandos, sendo que neste caso o valor específico é de cinquenta, o que representa o estado de carga mínimo a ser definido. Tal como nos comandos anteriores este é formatado e enviado corretamente.

Para encerrar o teste à operação de envio do servidor, foi efetuado um comando de definição de horas para que, tal como foi feito anteriormente, se possa verificar que todas as operações associadas a este são efetuadas de forma correta. Os resultados obtidos podem ser observados na Figura 6.21. Nesta figura é possível observar que, à semelhança do que aconteceu nos comandos prévios, a única diferença na trama recebida é o valor do comando a efetuar pelo SGC, assim como os dados referentes ao comando, que neste caso

são o zero, o quinze, o dezasseis e o um. Estes valores representam o dia, a hora, os minutos e os segundos, respetivamente. Todos estes dados são formatados num inteiro e enviados via porta série à semelhança do que acontece com os outros comandos.

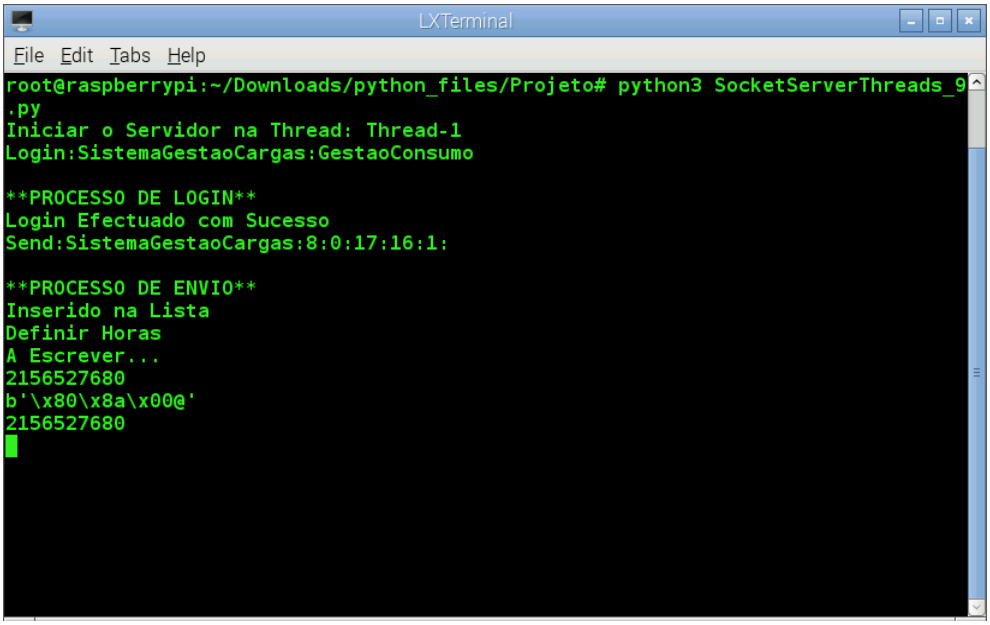


Figura 6.21 - Execução dos comando de definição de dia e hora.

Terminados os testes aos diferentes comandos da função de envio do servidor foi dado início ao teste da função de "Request" do servidor. Nesta operação o servidor recebe um comando de pedido informação da aplicação. No entanto, devido às características do SGC, este tem de ser dividido em dois antes de lhe ser enviado e, uma vez que o tamanho do barramento de dados é de trinta e dois bits e toda a informação que é pedida não cabe num único comando, este processo foi dividido em dois.

O servidor fica encarregue de enviar dois comandos diferentes para SGC, em que cada um vai requerer deste o envio de informações diferentes, sendo a informação proveniente destes dois comandos será unida e formatada para ser enviada para a aplicação, onde esta vai ser usada para preencher os elementos associados a cada informação.

Na Figura 6.22 é possível observar os resultados deste processo, sendo que ao contrário do que é feito nos comandos anteriores, a formatação e envio de cada comando é seguida da receção da informação que a execução de cada comando no SGC retorna. A trama associada a esta operação é a mais simples de todas, uma vez que apenas contém o valor a indicar a operação e o utilizador que requisitou essa informação. O passo que se segue à receção da trama é igual ao do comando de envio, que se trata da formatação e envio do comando para o SGC. A fase seguinte é única a esta operação, que é a fase de leitura marcada pela indicação "A Ler…". É possível observar que a seguir é apresentada

a informação lida formatada já nos seus diferentes valores, de forma a ser enviada para a aplicação, para que esta possa ser usada no preenchimento dos diferentes campos que são apresentados ao utilizador.

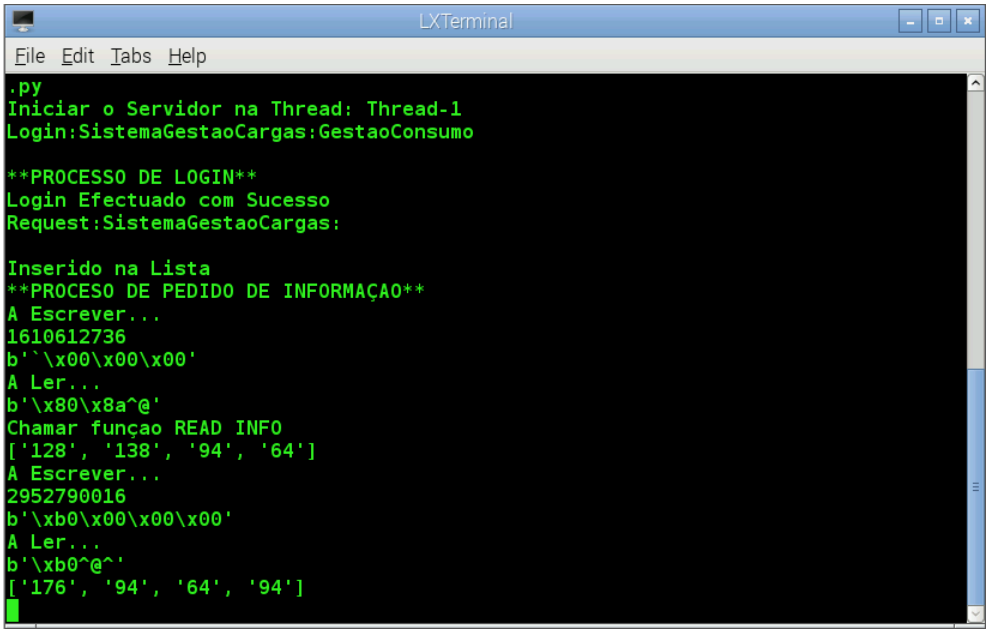


Figura 6.22 - Execução de um pedido de informação ao SGC.

6.4. Teste ao Sistema Gestão de Cargas

Na última fase de testes à implementação foram realizados os testes ao sistema de gestão de cargas. Como este sistema já foi simulado exaustivamente no capítulo três, referente à exceção do módulo de comunicações que, devido às suas características não seria vantajoso passar pela fase de simulação. Devido a este facto, este módulo será o primeiro a ser testado individualmente, para verificar que o seu funcionamento é o pretendido. Uma vez verificado o funcionamento deste, o sistema de gestão de cargas será testado na sua totalidade já integrado com todas as outras partes, nomeadamente aplicação e servidor, o que verificará o sistema como um todo.

6.4.1. Teste ao Módulo de Comunicação

Para testar o módulo de comunicação foram usadas duas placas Basys2, cada uma com uma versão do módulo. Como uma das características da comunicação via porta série é esta ser assíncrona, as placas foram configuradas para usarem frequências de relógio diferentes, de forma que os testes fossem realizados numa situação mais próxima do real.

Foram utilizadas como frequências de relógio de 100MHz e 50MHz, tirando partido assim dos diferentes valores de relógio que podem ser gerados pelo oscilador

presente na placa, sendo que para a produção do sinal de 100MHz foi necessário a adição de um *jumper* na placa para fazer a seleção deste valor de relógio.

A introdução dos dados foi realizada através dos oito interruptores existentes nas placas, SW0 até ao SW7. Para visualização da informação recebida foram utilizados oito LED presentes nas placas, LD0 até ao LD7. Sendo assim, o teste consistiu em colocar os valores a enviar nos interruptores da placa e pressionar o botão para assinalar que a informação está pronta a ser enviada, BTN0, o que dá início a este processo. Na Figura 6.23 e na Figura 6.24 podem ser observados os resultados obtidos. Na Figura 6.23 pode ser observado nos LED que o valor recebido foi 00000111 e através dos interruptores verifica-se que foi enviado o valor 11100000.

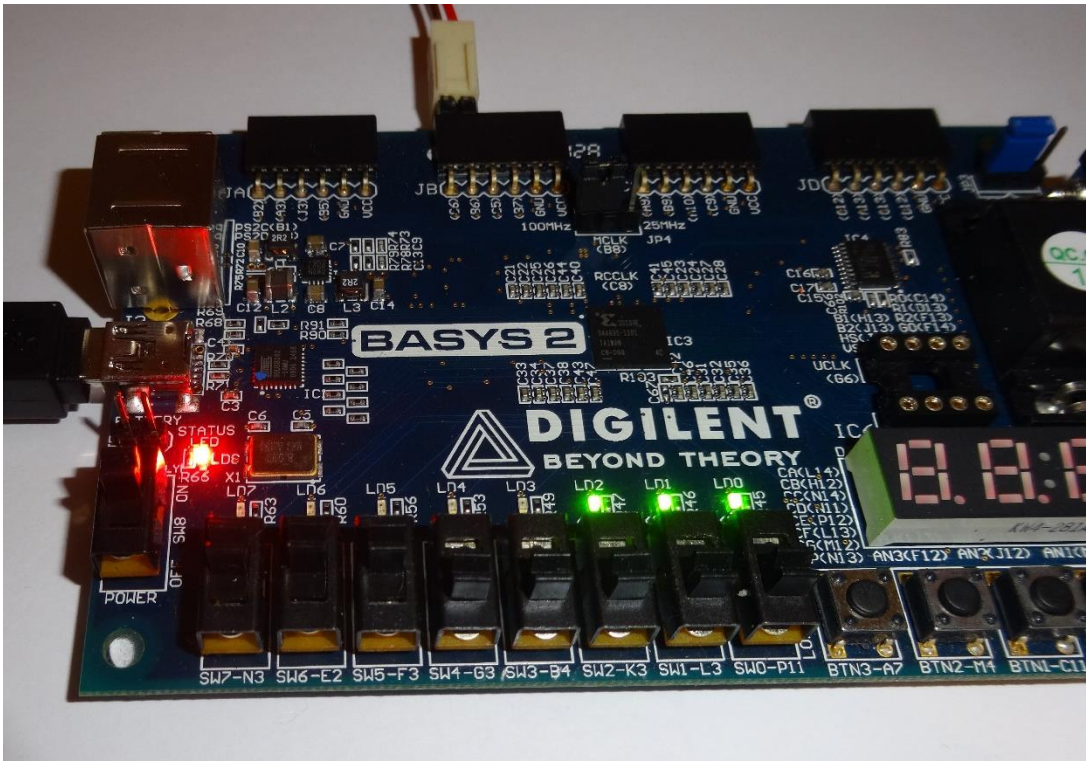


Figura 6.23 - Resultados da comunicação na placa com 100MHz de relógio.

Na Figura 6.24 verifica-se através dos LED que se encontram ligados, que o valor recebido está correto, uma vez que corresponde à disposição dos interruptores na outra placa. Nesta imagem também é possível verificar que a receção da informação na primeira placa está correta pois, a informação recebida coincide com a disposição dos interruptores nesta segunda placa.

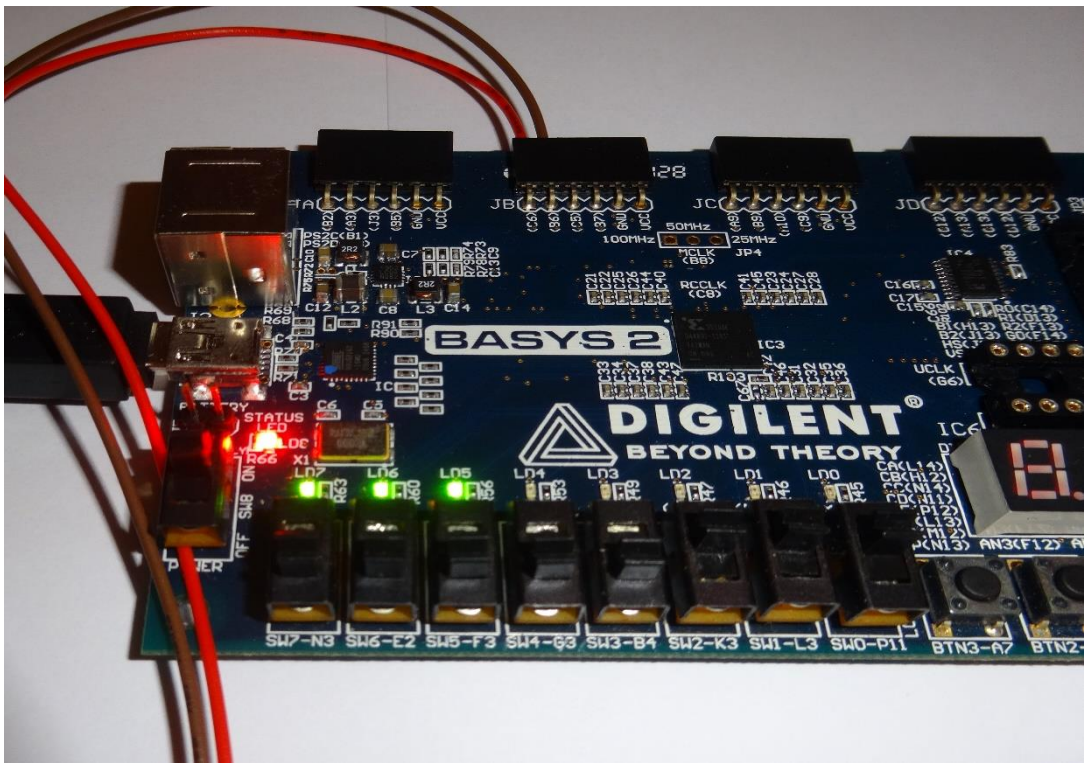


Figura 6.24 - Resultados da comunicação na placa com 50MHz de relógio.

Este teste verificou o correto funcionamento do envio e receção de um único *byte*, no entanto, como no funcionamento real do sistema gestão de cargas necessita do envio e receção de tramas de trinta e dois bits foi realizado um outro teste, desta vez para verificar se esta trama é recebida e enviada corretamente e se o funcionamento geral do gestor de cargas está correto. Este teste será realizado já com todos os outros intervenientes na gestão de cargas, neste caso a aplicação e o servidor.

6.4.2. Teste ao Módulo Completo

Neste teste todas a partes do sistema foram utilizadas em conjunto de forma a testar o funcionamento do gestor de cargas, assim como a comunicação entre todas as partes intervenientes no processo de gestão de cargas.

Antes de se poder realizar este teste foi necessário desenvolver um módulo adicional, uma vez que apenas existem oito LED para mostras de informação e os comandos que se pretendem testar têm um tamanho de dados de trinta e dois bits. Assim sendo, o módulo desenvolvido para este teste recebe a informação no tamanho original e divide esta em segmentos de oito bits de forma a estes poderem ser exibidos nos LED.

O *byte* a ser exibido é escolhido através dos interruptores SW0 e SW1, que representam o valor do *byte* a ser exibido. Quando o valor destes dois é 00, ou seja, ambos estão desligados, o *byte* que é apresentado é o zero. Quando ambos estão ligados,

apresentando então o valor 11 o *byte* que é apresentado é o terceiro. O processo repete-se para os restantes valores que estes interruptores podem assumir.

Neste teste o que vai ser passado a este módulo para ser exibido é o valor de saída do SGC, ou seja, o comando que vai ser enviado por este aos diferentes intervenientes possíveis. Por isso, para a verificação de resultados serão enviados comandos a partir da aplicação e verificar se o resultado que se encontra à saída do SGC é o esperado.

Para realizar este teste foi enviado, através da aplicação, um comando de alteração de estado, uma vez que este altera os registos internos do SGC e a saída deste. O valor escolhido para este comando foi 00010000001110100101101011111000, que se traduz num comando de alteração de estado, em que o valor do novo estado é a carregar (00000011), o valor do fator de potência a aplicar ao carregamento é noventa e cinco (01011111) e a percentagem de carregamento é setenta e cinco por cento (01001011).

Na Figura 6.25, é possível observar o *byte* mais significativo da saída do SGC, que tal como no comando recebido apresenta o valor 00010000, verificando assim, que o valor do primeiro *byte* se encontra em conformidade com o que foi enviado através da aplicação. É possível observar também, através desta figura que o SW1 e SW0 estão ambos ligados, que é o que faz com que seja exibido o valor do terceiro *byte*.

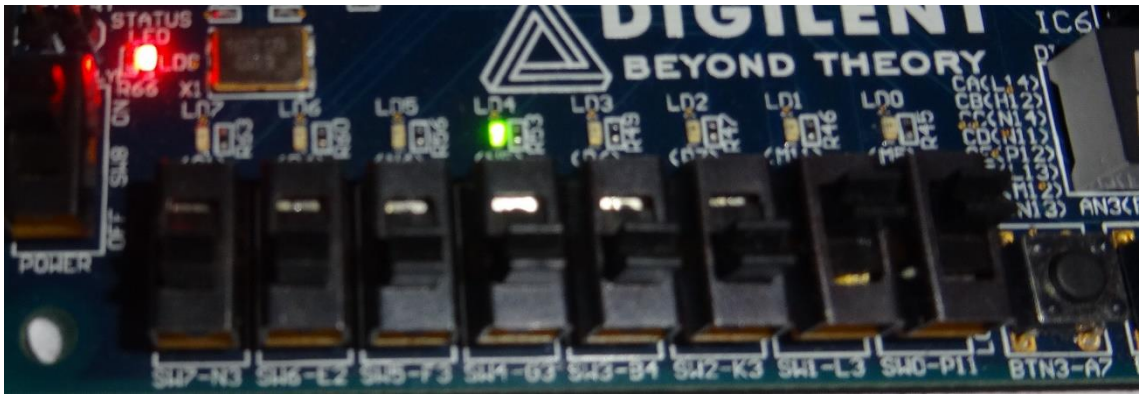


Figura 6.25 - *Byte* 3 (mais significativo) do comando de alteração de estado que se encontra à saída do SGC.

O valor do segundo *byte* da saída do SGC pode ser observado na Figura 6.26, sendo que este apresenta o valor de 00111010 tal como é pretendido. Também é possível observar nesta figura a configuração presente no SW1 e SW0, SW1 ligado e SW0 desligado, que faz a seleção deste *byte* para ser exibido.

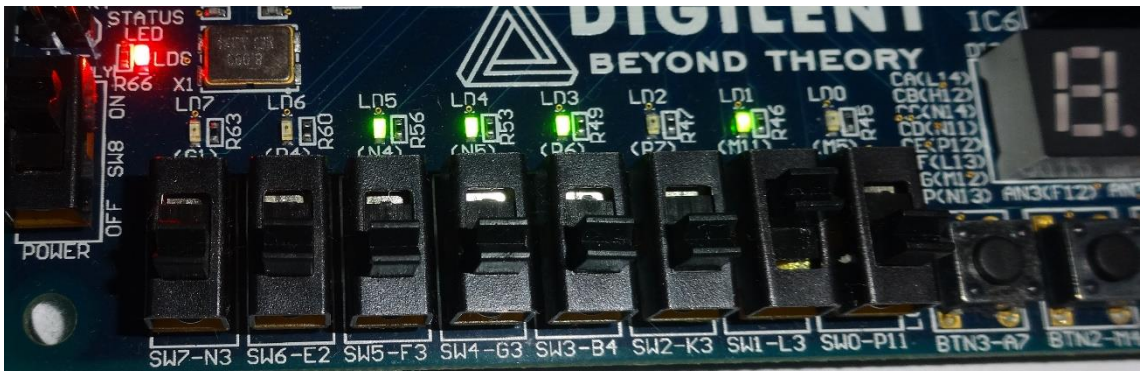


Figura 6.26 - *Byte* 2 do comando de alteração de estado que se encontra à saída do SGC.

Através da Figura 6.27 é possível ver que a nova configuração do SW1 e do SW0 apresenta agora o valor um em binário, o que faz com que seja selecionado para exibição nos LED o respetivo *byte* da saída, apresentando este o valor de 01011010, tal como no comando recebido da aplicação.

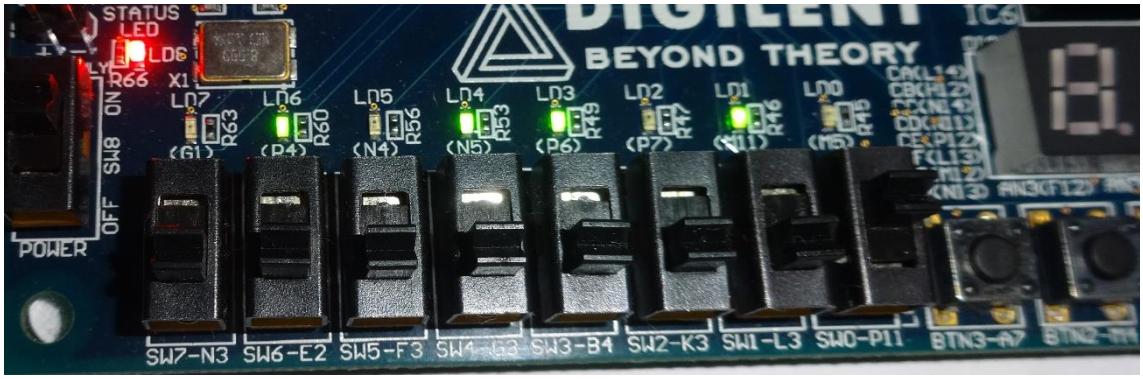


Figura 6.27 - *Byte* 1 do comando de alteração de estado que se encontra à saída do SCG.

O *byte* menos significativo da saída, bem como a configuração do SW1 e SW0 que o seleciona para exibição, podem ser observados na Figura 6.28. Ambos os interruptores encontram-se deligados, o que faz com que o valor combinado destes seja zero selecionando assim este *byte* para ser mostrado através dos LED. O valor que estes apresentam é de 11111000, tal como no comando enviado pela aplicação. Com este último *byte* verifica-se que a receção e execução deste se encontra correta.

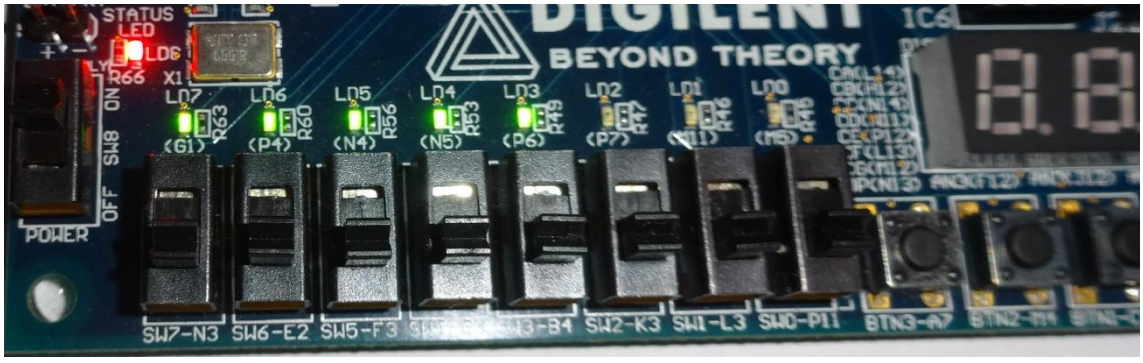


Figura 6.28 - *Byte* 0 do comando de alteração de estado que se encontra à saída do SGC.

Este teste verificou que a saída do módulo é alterada corretamente consoante o comando que lhe é enviado, no entanto falta ainda verificar se os registos internos são alterados corretamente. Para testar isto foi enviado para o SGC um comando de pedido de informação, uma vez que este coloca à saída os valores dos registos internos. O comando de informação escolhido para este teste foi o comando de informação um, visto que este coloca à saída o valor do registo de estado, do estado de carga atual, do registo de capacidade de carga e do registo do fator de potência formatados em uma trama de trinta e dois bits pela ordem apresentada. Como este comando foi efetuado a seguir ao comando de alteração de estado referido anteriormente, é possível verificar através dele se os registos foram alterados corretamente.

Na Figura 6.29, é possível verificar através do SW1 e SW0 que o *byte* selecionado é o terceiro, referente ao estado. O estado apresentado tem o valor 00000011, que indica que o estado definido é a carregar tal como tinha sido definido no comando de alteração de estado executado anteriormente.

Figura 6.29 - *Byte* referente ao Estado obtido através do comando de informação.

O valor do *byte* dois pode ser observado na Figura 6.30. O valor deste *byte* é zero, uma vez que este representa o estado de carga atual, que é um parâmetro que não é definido pelo comando de alteração de estado, mantendo assim o valor pré-definido.

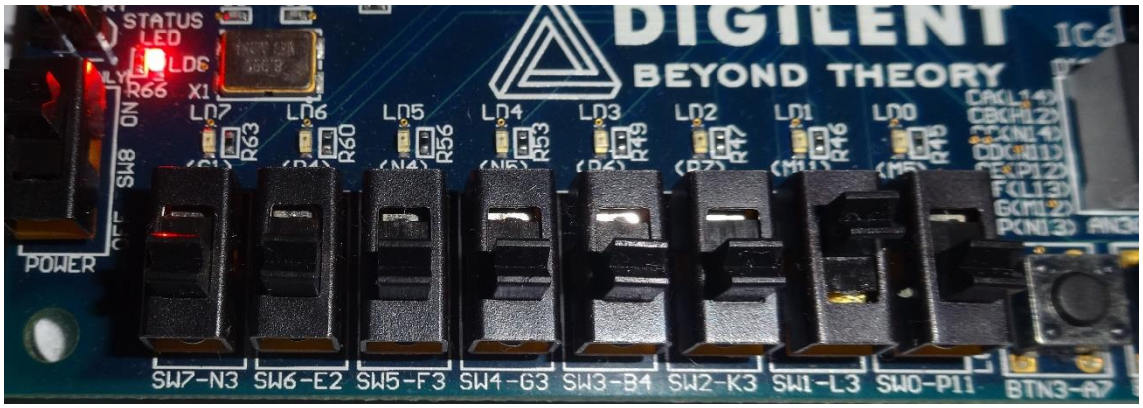


Figura 6.30 - *Byte* referente ao Estado de Carga Atual obtido através do comando de informação.

O byte um, referente à capacidade de carga utilizada, pode ser observado na Figura 6.31, onde se constata que o seu valor é 01001011, tal como foi indicado pelo comando de alteração de estado.

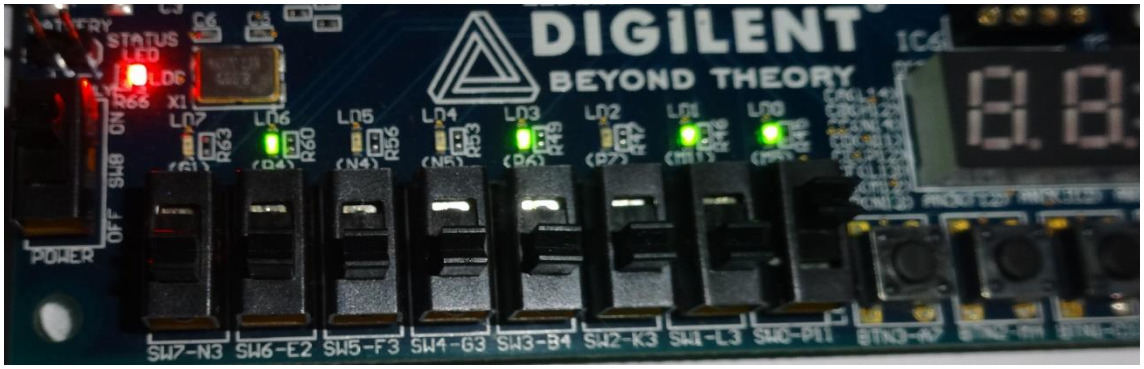


Figura 6.31 - *Byte* referente ao Capacidade de Carga Utilizada obtido através do comando de informação.

Por fim é selecionado para exibição o *byte* referente ao fator de potência, que ocupa a posição zero na trama de resposta ao comando de pedido de informação um, Figura 6.32. O valor deste é de 01011111, que mais uma vez vai de encontro ao que foi definido pelo comando de alteração de estado.

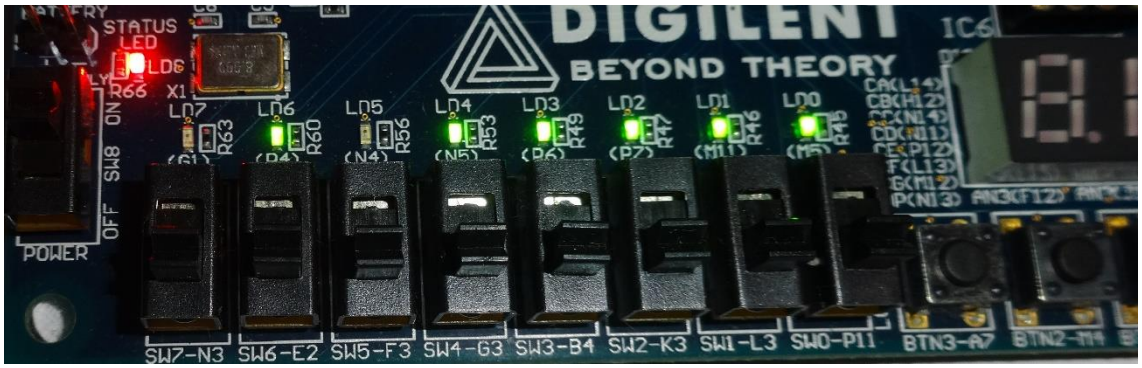


Figura 6.32 - *Byte* referente ao Fator de Potência obtido através do comando de informação.

Através destes dois testes foi possível verificar que o SGC recebe, interpreta e executa os comandos de forma correta. Ao visualizar os resultados produzidos pelos comandos na saída do módulo bem como nos seus registos internos, constata-se que este sistema funciona como pretendido, o que também era indicado pelos resultados obtidos previamente na fase de simulação do sistema.

6.5. Conclusão

Neste capítulo foram abordados os testes efetuados ao sistema desenvolvido, com a finalidade de verificar que todas as funcionalidades deste estão operacionais e que este é capaz de realizar a gestão de cargas, tal como é pretendido.

Os testes realizados à aplicação demostram que a sua aparência e composição estão de acordo com o pretendido. As funcionalidades desta também foram verificadas, ao executar as diversas operações que esta disponibiliza ao utilizador, obtendo em todas elas o resultado esperado. É, por isso, possível concluir que esta está apta a realizar a sua função principal que é de providenciar ao utilizador um meio para interagir com as restantes partes do sistema.

O servidor foi testado simultaneamente com a aplicação e as funcionalidades também foram verificadas. Foi constatado que este consegue fazer a receção e envio de todas as informações corretamente formatadas, para que seja possível a cada um dos intervenientes interpretar estas mesmas informações.

Numa fase final dos testes foram testadas as comunicações do Sistema de Gestão de Cargas (SGC). Os resultados obtidos com estes testes foram positivos, pelo que se passou para o teste da totalidade do SGC, que serviu também como teste de integração do sistema.

Uma vez que foram utilizadas todas as partes do sistema no teste ao SGC. Foi possível verificar que os comandos enviados pela aplicação são corretamente executados e recebidos pelo SGC. O que permite concluir que o sistema está operacional na sua totalidade.

CAPÍTULO 7

Conclusão

7.1. Conclusões

Esta dissertação trata do desenvolvimento de um Sistema de Gestão de Cargas (SGC) para uma *Smart Grid*, para ser utilizado no processo de gestão de consumo de energia, que irá ser um aspeto fundamental nas futuras *Smart Grids*. Este projeto teve como objetivo o desenvolvimento de um SGC capaz de desempenhar as funções essenciais para satisfazer as necessidades do processo de controlo de consumo.

No segundo capítulo foi realizada uma análise ao estado da arte dos sistemas de gestão de cargas, fazendo primeiro uma avaliação do que se entende por gestão de cargas, quais são as suas vantagens e os requisitos necessários para que este tipo de gestão possa ser implementada. Dentro destes requisitos encontram-se os SGC, que sendo estes o foco do trabalho receberam maior atenção neste estudo, sendo também analisados alguns exemplos já existentes no mercado. Neste capítulo foram analisadas, ainda, as *Smart Grids* uma vez que é nelas que se irão inserir os SGC. Esta análise focou-se nas vantagens, requisitos e desafios à implementação das *Smart Grids* de forma a perceber melhor em que estas consistem.

Terminado o estudo do estado da arte sobre os SGC e tendo-se obtido os requisitos que estes devem cumprir para poderem contribuir para o esforço de gestão de consumo, foi efetuado no terceiro capitulo o projeto do sistema a ser implementado. Este iniciou-se por uma avaliação das funcionalidades que o sistema deve implementar, uma descrição geral do sistema assim comodo seu funcionamento. Após efetuada a avaliação foram descritas as diversas partes do sistema, começando por uma descrição dos componentes e ferramentas utilizadas, passando depois para a descrição do funcionamento de cada uma das partes, aplicação, servidor e SGC, sendo também descritos os diversos sub-módulos que implementam as diversas funcionalidades internas deste.

Uma vez terminado o projeto e a definição das funcionalidades que cada parte do sistema teria de desempenhar, foi dado início à simulação do SGC através da ferramenta de simulação ISim, sendo este o único módulo cuja simulação traria vantagens em relação

a uma passagem direta para a implementação, assim como a realização de testes durante o processo de implementação. As simulações foram executadas por partes e de baixo para cima, isto é, dos elementos mais básicos para os mais complexos. Os resultados das simulações verificaram que o funcionamento dos diversos componentes cumpria os objetivos definidos no capítulo anterior, para o SGC.

Uma vez concluídas as simulações iniciou-se a implementação do sistema, sendo nesta parte descritas as alterações e configurações necessárias às diferentes ferramentas utilizadas no processo de implementação.

Para a implementação da aplicação foi necessário configurar o Android Studio, para que este estivesse preparado para utilizar a versão escolhida do Android e para que o aspeto do desenho da aplicação correspondesse ao dispositivo onde esta ia ser implementada.

A implementação do servidor necessitou da instalação e configuração do sistema operativo a correr na placa. Foram instalados os conteúdos adicionais necessários para que o servidor fosse capaz de desempenhar todas as funções pretendidas, sendo estes conteúdos o servidor de MySQL, a biblioteca necessária para fazer a interação com este através do Python e, por fim, a biblioteca para a comunicação via porta série, também para esta linguagem.

Para a concretização do SGC foi necessário a configuração do IDE utilizado para a FPGA escolhida de forma que este pudesse fornecer informação sobre a utilização dos recursos desta, algo importante durante o processo de implementação, mas principalmente para que este pudesse gerar o *bitstream* de forma correta. Durante o processo de implementação desta parte do sistema foi ainda necessário a utilização de ferramentas adicionais para mapear as saídas e entradas dos módulos, para as saídas e entradas da placa utilizada (PlanAhead). Foi ainda essencial a utilização do programa Adept da Diligent Inc para que código gerado pudesse ser programado na FPGA.

Terminado o processo de implementação de todas as partes do sistema, foram concretizados os testes ao sistema, que foram realizados de forma segmentada para facilitar a deteção e correção de erros.

A primeira fase dos testes foi realizada apenas com a aplicação e com o servidor ligados entre si, para verificar as funcionalidades de ambos. Esta fase visou também a verificação do aspeto e a organização dos elementos presentes na aplicação. Com estes testes foi possível concluir que ambas as partes do sistema estavam a funcionar como pretendido.

A fase seguinte foi o teste, de forma isolada, do sub-módulo de comunicação do SGC para verificar o bom funcionamento deste. Este sub-módulo foi testado de forma individual, para que possíveis erros na comunicação não fossem confundidos com erros funcionais do SGC. Os resultados obtidos ao teste deste sub-módulo foram positivos, pelo que foi dado início ao teste da totalidade do SGC.

O teste ao SGC funcionou também como teste de integração do sistema, uma vez que neste já foram utilizados todos os elementos do sistema, funcionando de forma conjunta. É possível observar através dos resultados, que todos os elementos estão a funcionar corretamente e que a interação entre eles opera como pretendido, demostrando assim que o SGC está operacional.

7.2. Sugestões para Trabalho Futuro

Os testes realizados ao sistema demonstram que este é capaz de implementar as funções necessárias para que as cargas possam ser usadas de forma a contribuir para a gestão de consumo, assim como a sua capacidade de interação com os utilizadores, rede e cargas de forma eficaz de forma a satisfazer as necessidades de todas estas partes.

No entanto, como este tipo de sistemas terão sempre como objetivo alcançar o maior número possível de utilizadores, interessa migrar a aplicação móvel para outras plataformas móveis, apesar de as plataformas Android serem as com maior número de utilizadores, existem outras que possuem também uma grande quota de mercado, como por exemplo as plataformas iOS, logo seria importante criar uma versão da aplicação que fosse capaz de correr neste sistema, de forma a aumentar o número de utilizadores da aplicação. O passo seguinte, seria a aplicação do sistema a plataformas Windows, assim como outras plataformas, de forma a expandir a sua abrangência.

Na aplicação será também interessante expandir as funcionalidades oferecidas ao utilizador, embora este assunto não esteja diretamente relacionado com as funcionalidades do SGC, isto é, será interessante que a aplicação seja capaz de mostrar ao utilizador todas as ações que foram realizadas e por quem. Seria ainda interessante utilizar a informação operacional armazenada, para mostrar ao utilizador informações sobre os ganhos provenientes da participação nos programas de controlo de consumo.

Em relação ao servidor, este teria de ser capaz de armazenar as informações referentes aos acessos e operações efetuadas, assim como armazenar as informações referentes aos custos e ganhos provenientes do controlo de consumo efetuado na carga,

sendo para isso necessária a implementação de um esquema de base de dados mais complexo.

Será também importante dotar o servidor de mais mecanismos de segurança, apesar de já ter sido implementado um algoritmo de codificação para armazenamento de palavras passe. Uma solução para este facto, seria também efetuar a encriptação de todas as comunicações realizadas, de forma a garantir que mesmo que estas sejam intercetadas, não possam ser analisadas ou alteradas por intervenientes não autorizados.

Por fim, as melhorias que podem ser feitas no SGC são a incorporação de mecanismos de codificação e descodificação de comunicações, tal como no servidor, e a integração neste de algoritmos de previsão e otimização de forma a facilitar e oferecer mais possibilidades de gestão de consumo ao utilizador, tirando para isto partido das capacidades de processamento paralelo das FPGAs.

Referências

[1] D. Van Hertem, J. Rimez, and R. Belmans, "Power flow controlling devices as a smart and independent grid investment for flexible grid operations: Belgian case study," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1656–1664, 2013.

[2] M. C. Vlot, J. D. Knigge, and J. G. Slootweg, "Economical regulation power through load shifting with smart energy appliances," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1705–1712, 2013.

[3] G. Koutitas and L. Tassiulas, "Periodic flexible demand: Optimization and phase management in the smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1305–1313, 2013.

[4] Z. Chen and L. Wu, "Residential appliance DR energy management with electric privacy protection by online stochastic optimization," *IEEE Trans. Smart Grid*, vol. 4, pp. 1861–1869, 2013.

[5] M. Amin and J. Stringer, "The Electric Power Grid : Today and Tomorrow," *MRS Bulletin*, vol. 33, no. April, pp. 399–407, 2008.

[6] O. Ma, N. Alkadi, P. Cappers, P. Denholm, J. Dudley, S. Goli, M. Hummon, S. Kiliccote, J. Macdonald, N. Matson, D. Olsen, C. Rose, M. D. Sohn, M. Starke, B. Kirby, and M. O. Malley, "Demand Response for Ancillary Services," vol. 4, no. 4, pp. 1988–1995, 2013.

[7] S. Mohagheghi, F. Yang, and B. Falahati, "Impact of demand response on distribution system reliability," *IEEE Power Energy Soc. Gen. Meet.*, pp. 1–7, 2011.

[8] G. Strbac, "Demand side management: Benefits and challenges," *Energy Policy*, vol. 36, no. 12, pp. 4419–4426, 2008.

[9] D. Jay and K. S. Swarup, "Frequency restoration using Dynamic Demand Control under Smart Grid Environment," *ISGT2011-India*, pp. 311–315, 2011.

[10] J. Aghaei and M. I. Alizadeh, "Demand response in smart electricity grids equipped with renewable energy sources: A review," *Renew. Sustain. Energy Rev.*, vol. 18, pp. 64–72, 2013.

[11] Y. Zong, D. Kullmann, A. Thavlov, O. Gehrke, and H. W. Bindner, "Application of model predictive control for active load management in a distributed power system with high wind penetration," *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 1055–1062, 2012.

[12] D. P. Chassin and K. Kalsi, "Effects of demand response on retail and wholesale power markets," *IEEE Power Energy Soc. Gen. Meet.*, pp. 1–7, 2012.

[13] M. Erol-Kantarci and H. T. Mouftah, "Supply and load management for the smart distribution grid using wireless networks," *2012 Japan-Egypt Conf. Electron. Commun. Comput.*, vol. 4, pp. 145–150, 2012.

[14] W. Zhang, S. Zhou, and Y. Lu, "Distributed Intelligent Load Management and Control System," *IEEE PES Gen. Meet. 2012*, pp. 1–8, 2012.

[15] Q. Hu and F. Li, "Hardware design of smart home energy management system with dynamic price response," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 1878–1887, 2013.

[16] EFACEC, "G Smart." [Online]. Available: http://www.efacec.pt/PresentationLayer/ResourcesUser/Catalogos 2012/Automa%C3%A7%C3%A3o/as74i0903e1_GSmart_EN.pdf. [Accessed: 06-Oct-2015].

[17] ABB, "Load Management with Ekip Power Controller for SACE Emax 2." [Online]. Available: https://library.e.abb.com/public/cbd96b4c9390350bc1257b4800498fdd/1SDC007410G0201.pdf. [Accessed: 06-Oct-2015].

[18] S. Electric, "Wiser Home Management." [Online]. Available: http://oreo.schneider-electric.com/flipFlop/387996237/index.htm?p_EnDocType=Brochure&p_Reference=0100BR1004&p_File_Name=998-4758_US-Web.pdf&flipflop=1#/1. [Accessed: 06-Oct-2015].

[19] J. M. Lujano-Rojas, C. Monteiro, R. Dufo-López, and J. L. Bernal-Agustín, "Optimum residential load management strategy for real time pricing (RTP) demand response programs," *Energy Policy*, vol. 45, pp. 671–679, 2012.

[20] J. M. Lujano-Rojas, C. Monteiro, R. Dufo-López, and J. L. Bernal-Agustín, "Optimum load management strategy for wind/diesel/battery hybrid power systems," *Renew. Energy*, vol. 44, pp. 288–295, 2012.

[21] S. Mei and L. Chen, "Research focuses and advance technologies of smart grid in recent years," *Chinese Sci. Bull.*, vol. 57, no. 22, pp. 2879–2886, 2012.

[22] P. M. Hart, "Continuous asset monitoring on the smart grid," *2011 IEEE PES Innov. Smart Grid Technol. ISGT Asia 2011 Conf. Smarter Grid Sustain. Afford. Energy Futur.*, 2011.

[23] C. H. Hauser, D. E. Bakken, and a. Bose, "A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid," *IEEE Power Energy Mag.*, vol. 3, no. 2, pp. 47–55, 2005.

[24] H. Tekiner-Mogulkoc, D. W. Coit, and F. a. Felder, "Electric power system generation expansion plans considering the impact of Smart Grid technologies," *Int. J. Electr. Power Energy Syst.*, vol. 42, no. 1, pp. 229–239, 2012.

[25] W. H. E. Liu, K. Liu, and D. Pearson, "Consumer-centric smart grid," *IEEE PES Innov. Smart Grid Technol. Conf. Eur. ISGT Eur.*, 2011.

[26] F. Blaabjerg and J. M. Guerrero, "Smart grid and renewable energy systems," *2011 Int. Conf. Electr. Mach. Syst.*, pp. 1–10, 2011.

[27] M. Kolhe, "Smart Grid: Charting a New Energy Future: Research, Development and Demonstration," *Electr. J.*, vol. 25, pp. 88–93, 2012.

[28] Mohibullah and S. H. Laskar, "Power quality issues and need of intelligent PQ monitoring in the smart grid environment," *Proc. Univ. Power Eng. Conf.*, 2012.

[29] J. Jackson, "Improving energy efficiency and smart grid program analysis with agent-based end-use forecasting models," *Energy Policy*, vol. 38, no. 7, pp. 3771–3780, 2010.

[30] W. Ketter, J. Collins, and P. Reddy, "Power TAC: A competitive economic simulation of the smart grid," *Energy Econ.*, vol. 39, pp. 262–270, 2013.

[31] C. K. Park, H. J. Kim, and Y. S. Kim, "A study of factors enhancing smart grid consumer engagement," *Energy Policy*, vol. 72, pp. 211–218, 2014.

[32] F. R. M. Leijten, J. W. Bolderdijk, K. Keizer, M. Gorsira, E. van der Werff, and L. Steg, "Factors that influence consumers' acceptance of future energy systems: the effects of adjustment type, production level, and price," *Energy Effic.*, pp. 973–985, 2014.

[33] T. Flick and J. Morehouse, "Securing the Smart Grid," *Secur. Smart Grid*, pp. 19–33, 2011.

[34] M. Gargenta, *Leaning Android*, First Edit. CA: Sebastopol: O'Reilly Media, Inc., 2011.

[35] A. Developers, "Eclipse ADT." [Online]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html. [Accessed: 22-Sep-2015].

[36] "Android Studio Overview." [Online]. Available: http://developer.android.com/tools/studio/index.html. [Accessed: 21-Jul-2015].

[37] B. C. Zapata, *Android Studio Application Development*, First Edit. Birmingham: Packt Publishing, 2013.

[38] J. Friesen, *Learn Java for Android Development*, Third Edit. NY: New York: Apress Media, LLC, 2014.

[39] A. ; Backiel, S. Brouke, and B. Baesens, *Beginning Java Programming*, First Edit. Indianapolis, Indiana: John Wiley & Sons, Inc., 2015.

[40] R. P. Foundation, "WHAT IS A RASPBERRY PI?" [Online]. Available: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/. [Accessed: 20-Jul-2015].

[41] R. P. Foundation, "THE MAKING OF PI." [Online]. Available: https://www.raspberrypi.org/about/. [Accessed: 20-Jul-2015].

[42] Raspbian, "Welcome to Raspbian." [Online]. Available: https://www.raspbian.org/. [Accessed: 20-Jul-2015].

[43] M. Lutz, *Python Pocket Reference*, Fifth Edit. CA: Sebastopol: O'Reilly Media, Inc., 2014.

[44] M. Lutz, *Learning Python*, Fifth Edit. CA: Sebastopol: O'Reilly Media, Inc., 2013.

[45] P. S. Foundation, “IDLE.” [Online]. Available: https://docs.python.org/3/library/idle.html. [Accessed: 22-Jul-2015].

[46] D. Inc., “Basys2 Reference Manual.” [Online]. Available: https://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf. [Accessed: 22-Jul-2015].