UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA


# Realização de uma ponte *Ethernet*-SPI para comunicação e controlo de placas de alta tensão do detetor *TileCal* da experiência ATLAS do LHC

João Maria Almendra Sabino

**Mestrado Integrado em Engenharia Física**

Dissertação orientada por:
Prof. Doutor José António Soares Augusto

2017

# Agradecimentos

À minha família pelo apoio incondicional durante esta jornada da minha vida que foi o meu percurso académico.

Aos meus colegas e amigos pelos momentos que me proporcionaram durante o tempo passado na FCUL.

Ao Professor José Augusto pela orientação e apoio proporcionados durante a realização deste trabalho e no meu percurso na FCUL. À Professora Guiomar, ao Professor Agostinho Gomes e à Professora Patrícia Conde pela disponibilidade demonstrada ao longo deste trabalho.

Ao Filipe Martins, Luís Gurriana e Luís Seabra pela paciência e apoio durante a minha estadia no CERN e durante o desenvolvimento deste trabalho.

Ao Departamento de Física da FCUL por me terem disponibilizado o espaço e o equipamento necessário à realização deste trabalho.

Ao LIP pela bolsa de investigação que me foi atribuída no âmbito deste projeto e da colaboração portuguesa na experiência *TileCal* do ATLAS e que me permitiu realizar este trabalho.

# Resumo

O trabalho apresentado nesta tese foi realizado no âmbito do projeto de atualização do calorímetro *TileCal* do detetor ATLAS, mais concretamente, na substituição das placas *HV-Opto* por uma versão melhorada (*HV-Remote*) que controlam a alta tensão fornecida aos fotomultiplicadores (PMTs) do detetor. Atualmente, a placa *HV-Opto* encontra-se em operação há mais de 10 anos, e uma vez que o equipamento se encontra no interior do detetor, está sujeita a elevadas doses de radiação. Outro problema prende-se com a necessidade de substituir equipamento avariado, o que, devido à radiação, só é possível quando se realizam paragens técnicas do LHC, algo que acontece esporadicamente. Assim, uma das propostas de atualização daquele sistema consiste em mover a eletrónica de controlo do *TileCal* atualmente no detetor, para um local na caverna USA15, num ambiente virtualmente sem radiação pois encontra-se a 100 m do detetor.

A atualização proposta pelo grupo de investigação do LIP (Laboratório de Instrumentação e Física Experimental de Partículas) e de outras instituições participantes no projeto está dividida em três partes: conceção e produção de uma carta-protótipo (*HV-Remote*), realização de testes com o protótipo e criação do sistema de controlo da placa *HV-Remote*. O trabalho realizado nesta tese incide no sistema de controlo da placa, e teve como objetivo a criação de uma interface que permite ao utilizador gerir os valores de tensão que são aplicados aos PMTs do detetor.

Adquiriu-se uma placa de teste que contém um módulo programável em linguagem C e/ou BASIC. Este módulo (da TIBBO), caso o seu desempenho cumpra os requisitos, será colocado na versão final da *HV-Remote* e terá como função direcionar a informação enviada pela interface de utilizador para os subsistemas, ou componentes, da placa, para que a respetiva parametrização seja efetuada. A interface de utilizador foi desenvolvida em linguagem C++, com recurso ao programa *WinCC* atualmente utilizado no controlo do *TileCal*. Esta interface permitirá ao utilizador escrever e ler valores de tensão aplicados aos PMTs, bem como controlar ou monitorizar outras variáveis, tais como a temperatura, que são cruciais para o normal funcionamento de todo o sistema. A ligação entre a interface e a placa *HV-Remote* será feita com *Sockets*, através de uma ligação via *Ethernet*, utilizando o protocolo de comunicação SPI para o envio de dados.

No trabalho, foram alcançadas com sucesso as seguintes metas:

- Teste do módulo programável que se encontra anexado a uma placa de testes.
- Teste do canal SSI do módulo (em configuração de *Master)* com *Arduinos Uno* (em configuração de *Slave*) e comunicação por *Sockets* sobre *Ethernet.*
- Teste do módulo TIBBO com os vários componentes que constituem a placa *HV-Remote*.
- Planificação e conceção do painel (interface de utilizador) para o DCS.
- Programação dos vários botões do painel.
- Testes finais com o módulo TIBBO, o painel DCS e os componentes da placa *HV-Remote*.

Palavras Chave: *TileCal*, *LHC upgrade*, *HV-Remote*, *WinCC*, DCS.

# Abstract

The work presented in this thesis was carried out as part of the project to upgrade the *TileCal* detector of the ATLAS experiment, more specifically, the replacement of the *HV-Opto* boards by an improved version (*HV-Remote*) that controls the high voltage supplied to the PMTs of the detector. The *HV-Opto* board has been in operation for more than 10 years now, and since the equipment is inside the detector, it is subject to high doses of radiation. Another problem may arise when some equipment needs to be replaced, a task that, due to radiation is only possible of being done when performing technical stops of the LHC, something that happens sporadically. Thus, one of the upgrade proposals consists in moving the detector's *TileCal* control electronics to a location in the USA15 cave, which has a much lower radioactive environment, located 100 m apart from the detector, thus allowing the system a longer expected operation time.

The upgrade proposed by the group is divided into three parts: design and production of a prototype (*HV-Remote*), testing of the prototype and creation of the control system for the new version of the *HV-Remote* board. The work carried out in this thesis focuses on the control system of the board, and aims to create an interface that allows the user to manage the voltage values that are applied to the PMTs of the detector.

A test board, which contains a module (from TIBBO) programmable in C and/or BASIC language, was acquired for the project. This module, if its performance is satisfactory, will be placed in the final version of the *HV-Remote* and will have the function of directing the information sent by the user interface to the correct sub-system in the board to carry out the instructions. The user interface will be developed in the C ++ language, using the *WinCC* program already used in the *TileCal* control. This interface will allow the user to write and read voltage values applied to the PMTs, as well as control or monitor other factors, such as temperature, that are crucial to the normal operation of the entire system. The connection between the interface and the *HV-Remote* board will be made via *Sockets*, through an *Ethernet* connection, using the SPI communication protocol for sending and receiving data.

The following tasks have been acomplished:

- Test the TIBBO module that is attached to a test board.
- Test the module's SSI channel (in master configuration) with *Arduinos Uno* (in slave configuration) and Sockets' communication over *Ethernet*.
- Test the module linked to several components that are in the *HV-Remote* board.
- DCS panel planning and design.
- Coding of the various buttons in the panel.
- Final tests with the TIBBO module, the DCS panel and the components acquired.

Keywords: *TileCal, LHC Upgrade, HV-Remote, WinCC,* DCS

# Índice

# Índice de figuras

# Índice de tabelas

# Siglas

| | |
|---|---|
| **ADC** | Analog to Digital Converter |
| **ASCII** | American Standard Code for Information Interchange |
| **ATLAS** | A Toroidal LHC ApparatuS |
| **BE** | Back End |
| **CAN** | Controller Area Network |
| **CERN** | Conseil Européen pour la Recherche Nucléaire |
| **CPHA** | Clock PHAse |
| **CPOL** | Clock POLarity |
| **CU** | Control Unit |
| **DAC** | Digital to Analog Converter |
| **DCS** | Detector Control System |
| **DIP** | Dual In-Line Package |
| **DU** | Device Unit |
| **EB** | Extended Barrel |
| **ELMB** | Embedded Local Monitor Board |
| **FE** | Front End |
| **fLVPS** | finger Low Voltage Power Supply |
| **FSM** | Finite State Machine |
| **GCS** | Global Control Station |
| **GPIO** | General Purpose Input Output |
| **HV** | High Voltage |
| **HVPS** | High Voltage Power Supply |
| **JCOP** | Java Card OpenPlatform |
| **LB** | Long Barrel |
| **LCS** | Local Control Station |
| **LED** | Light Emission Diode |
| **LHC** | Large Hadron Collider |
| **LSB** | Least Significant Bit |
| **LV** | Low Voltage |
| **MAC** | Media Access Control |
| **MISO** | Master Input Slave Output |
| **MOSI** | Master Output Slave Input |
| **MSB** | Most Significant Bit |
| **PLC** | Programmable Logic Controller |
| **PMT** | PhotoMultiplier Tube |

| | |
|---|---|
| **RAM** | Random Access Memory |
| **RTU** | Remote Terminal Unit |
| **SCS** | Subdetector Control Station |
| **SCT** | SemiConductor Tracker |
| **SLCK** | SeriaL ClocK |
| **SMI** | State Management Interface |
| **SPI** | Serial Peripheral Interface |
| **SS** | Slave Select |
| **SSI** | Synchronous Serial Interface |
| **TCP/IP** | Transmission Control Protocol/Intetnet Protocol |
| **TDAQ** | Trigger and Data AcQuisition |
| **TRT** | Transition Radiation Tracker |
| **USB** | Universal Serial Bus |
| **WLS** | WaveLength Shifter |

# 1. Introdução

O trabalho nesta tese foi desenvolvido no contexto da experiência ATLAS (A Toroidal LHC ApparatuS) do LHC (Large Hadron Collider) do CERN, que se localiza na fronteira entre a França e a Suíça, perto de Genebra, e é uma organização internacional fundada em 1954 que tem como principal objetivo o estudo das propriedades da matéria e das forças a ela associadas. O acrónimo "CERN" significa, em inglês, "*European Organization for Nuclear Research*", e o laboratório, além de ser o berço da internet, foi também palco de várias experiências importantes que resultaram na descoberta de correntes neutras através das quais ocorre a interação fraca, na descoberta dos bosões W e Z, principais intervenientes na interação fraca, e mais recentemente na descoberta do bosão de *Higgs*.

Para que novas descobertas sejam alcançadas, e haverá sempre algo mais a descobrir, é crucial que os detetores sejam constantemente melhorados, permitindo ir mais além nas descobertas da Física. É com base nesta premissa que o trabalho desenvolvido nesta tese encontra o seu ponto de partida, uma vez que poderá fazer parte da próxima atualização do ATLAS, focando-se na placa de alta tensão *HV-Remote*. Sendo do âmbito da participação portuguesa na experiência ATLAS, enquadra-se nos melhoramentos a efetuar nos sistemas eletrónicos da experiência *TileCal*-ATLAS do LHC com o objetivo de, num futuro próximo, suportar o aumento da taxa de aquisição de dados do sistema atualmente implementado.

## 1.1. Objetivos

O trabalho apresentado nesta tese faz parte de um projeto de desenvolvimento de uma versão mais moderna e mais fiável da atual placa *HV-Opto*, utilizada na experiência ATLAS, no calorímetro *TileCal*, para fornecer alta tensão a cerca de $10^4$ PMTs. Atualmente, o controlo da alta tensão fornecida aos PMTs é feito com as placas *HV-Opto* e *HV-Micro*. Em caso de ser necessária a substituição ou reparação de alguma destas placas, tal só poderá ocorrer quando houver uma paragem técnica devido à localização destas placas ser junto ao detetor. Assim, o projeto pretende mudar a localização atual (caverna do ATLAS) das placas que fazem o controlo da alta tensão do detetor para uma outra localização (caverna USA15) a cerca de 100 m de distância da localização atual, o que vai prevenir que o equipamento seja sujeito a doses de radiação elevadas.

O projeto em si, envolve o desenvolvimento de uma nova versão da placa *HV-Opto* que passará a designar-se de *HV-Remote*, o desenvolvimento de uma placa protótipo para testes para os vários componentes que constituem a *HV-Remote* e a programação de um sistema de controlo para esta placa, com base no programa *WinCC*, para o DCS (Detector Control System ) do detetor.

O objetivo do trabalho apresentado nesta tese, do todo que é a conceção desta nova versão do sistema de controlo da alta tensão aplicada aos PMTs, é a programação do módulo TIBBO EM1206-EV e a criação do painel DCS que servirá de interface para o controlo dos vários canais (PMTs) do detetor e de outros componentes das placas *HV-Remote*. O módulo TIBBO será utilizado para comunicar com os vários componentes da placa, via uma interface SPI (Serial Peripheral Interface), e estará também ligado a um computador através de uma ligação *Ethernet* e comunicará com o painel DCS através do protocolo TCP/IP, utilizando *sockets*.

## 1.2. Organização do trabalho

O presente trabalho está organizado em 7 capítulos. No capítulo 2 é feita uma descrição da experiência ATLAS e dos vários componentes que a constituem, tais como o detetor interno e os seus constituintes, os tipos de calorímetros, o espectrómetro de muões e o sistema magnético.

O capítulo 3 é dedicado ao calorímetro hadrónico *TileCal* e à sua arquitetura. São também dadas algumas breves noções de calorimetria, é retratado o percurso do sinal induzido pela captura de uma partícula desde a interação com a matéria, onde há deposição de energia, passando pelas fibras WLS até aos PMTs, até à reconstrução final levada a cabo pelo sistema de leitura.

O capítulo 4 é dedicado ao DCS do ATLAS, focando-se a sua organização e estrutura, e no final é estabelecido um paralelo com o DCS do *TileCal*.

Os capítulos 5 e 6 são dedicados ao trabalho prático desenvolvido, começando-se no capítulo 5 por descrever o equipamento utilizado e o seu teste. No capítulo 6 é abordada a ida do autor ao CERN, é feita uma descrição breve da nova versão da placa *HV-Remote* que o grupo se encontra a desenvolver, e, no final do capítulo, são apresentados e descritos os programas executados no módulo TIBBO e outros que implementam o painel DCS. São reportados alguns testes feitos ao sistema com os componentes que constituem a placa *HV-Remote*.

O capítulo 7, e último, é dedicado às conclusões e ao trabalho futuro.

Para além destes capítulos, apresentam-se 3 apêndices; nos 2 primeiros encontra-se parte do código desenvolvido, e no último mostra-se o certificado do prémio de "melhor poster", obtido com um relato do trabalho do nosso grupo no *TileCal*, na Conferência de Eletrónica, Telecomunicações e Computadores (CETC) de 2016.

Nota: Os termos utilizados em inglês são mantidos na língua original devido à sua utilização ser rotineira na experiência ATLAS.

# 2. A experiência ATLAS

A experiência ATLAS [29] é uma das sete experiências, de deteção de partículas, construída ao longo do acelerador LHC. Os aceleradores, como o LHC, têm como objetivo aumentar, sucessivamente, a energia dos feixes de partículas, até valores de energia muito elevados, e após esse processo fazer as partículas colidirem entre si, ou contra alvos estacionários. Os resultados dessas colisões são observados e registados pelos detetores, para serem posteriormente analisados.

O acelerador LHC é o maior acelerador de partículas do mundo, construído num túnel com cerca de 27 km de diâmetro, a uma profundidade de 175 metros, cujo principal objetivo é obter informação sobre colisões de feixes de várias partículas, tanto de protões (energia máxima registada: 13 TeV, valor obtido a 20 de maio de 2015), como de iões pesados, como é o exemplo do chumbo, que liberta uma energia, durante uma colisão, superior a 1250 TeV.

O detetor ATLAS foi projetado para tirar vantagem da energia disponível no LHC e observar fenómenos que envolvem partículas cujas massas são bastante elevadas, mas que não eram observáveis com os aceleradores mais antigos que dispunham de uma energia mais baixa. O detetor em si, tem 44 m de comprimento, 22 m de diâmetro e pesa cerca de 7000 toneladas. No seu interior, existem uma série de cilindros concêntricos em torno do ponto de interação, onde os feixes de partículas, nomeadamente de protões, colidem. O detetor está dividido em quatro partes principais:

- Detetor interno;
- Calorímetros: Eletromagnético e Hadrónico;
- Espectrómetro de muões;
- Sistemas magnéticos.

Cada uma destas divisões é composta por muitas outras camadas. O detetor interno (Figura 2.2) é constituído por um detetor de pixéis, por um detetor do tipo *tracker* de radiação de transição (TRT), e por um *tracker* semicondutor (SCT), e tem a função de rastrear as partículas em campos magnéticos bastante intensos, os calorímetros (Figura 2.5) de árgon líquido (calorímetro eletromagnético Figura 2.6) e de telhas (calorímetro hadrónico Figura 2.7), medem a energia das partículas, quando estas interagem com a matéria, o espectrómetro de muões (Figura 2.10), colocado em vários pontos do detetor, que faz medições do momento dos muões que atravessam o detetor. O detetor interno está imerso num campo magnético solenoidal e o espectrómetro de muões está sujeito a um campo magnético anelar, ambos produzidos pelo sistema magnético do detetor.

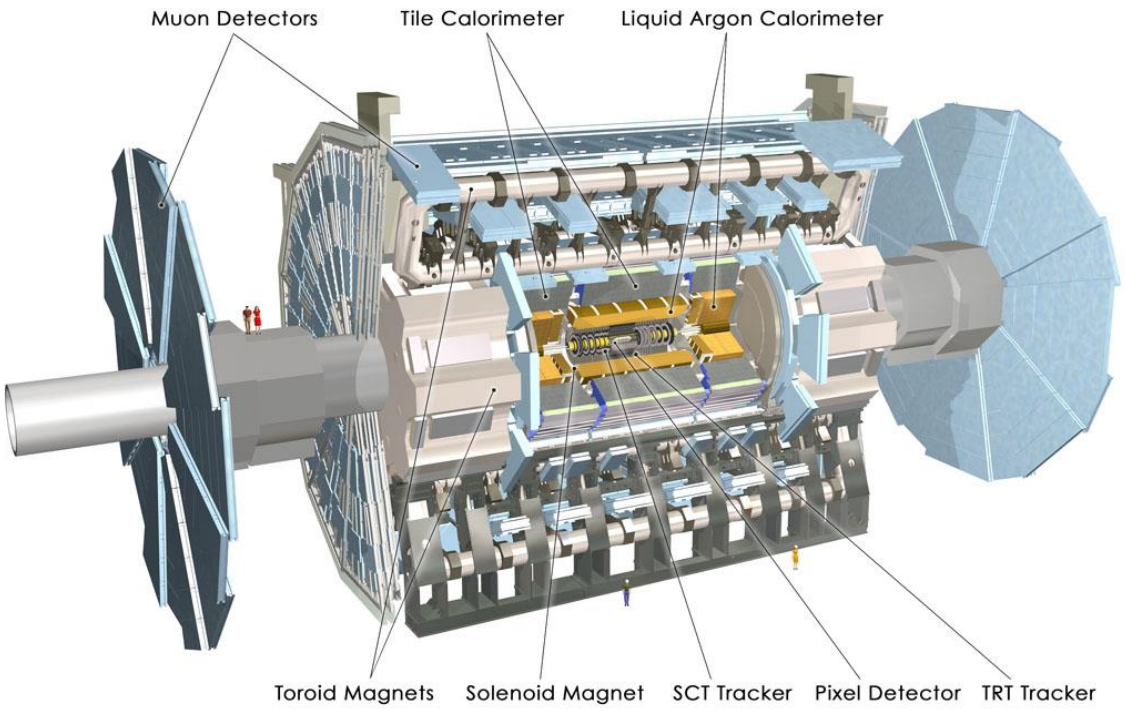


**Figura 2.1:** Ilustração do Detetor ATLAS

## 2.1. Detetor Interno

O detetor interno [1], [25] está concebido para que a sua função principal seja rastrear os trajetos das partículas, através da deteção da interação dessas partículas com a matéria, em pontos predeterminados, permitindo obter informações detalhadas sobre o tipo de partícula e sobre o seu momento, com elevada precisão.

O campo magnético em torno do detetor interno leva a que as partículas carregadas curvem e a concavidade dessa curvatura permite determinar a carga da partícula, e o raio de curvatura permite determinar o momento dessa partícula.

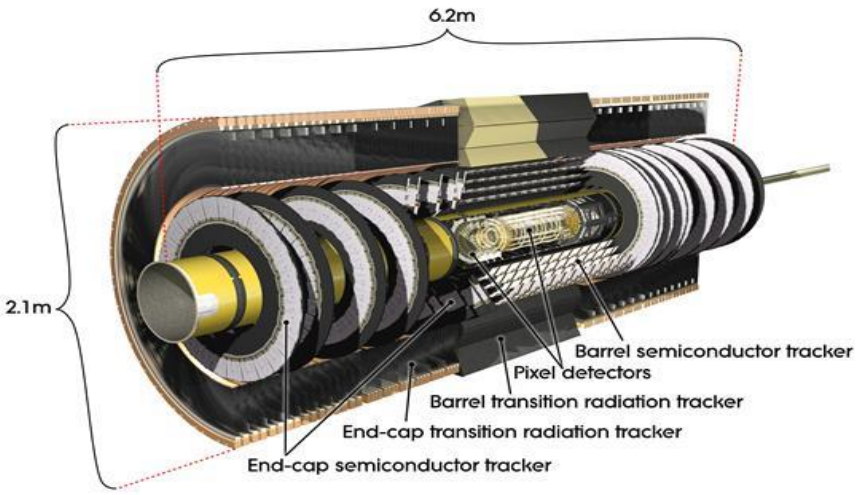


**Figura 2.2:** Detetor Interno do ATLAS

O detetor interno está dividido em três partes; um detetor de pixéis, um detetor semicondutor, e um *tracker* de radiação de transição. A descrição de cada uma apresenta-se em baixo.

### 2.1.1. Detetor de Pixéis

O detetor de pixéis encontra-se na parte interna do detetor interno, foi concebido para operar e ter a maior precisão possível perto do ponto de interação, levando a que esteja sujeito a níveis de radiação extremamente elevados, e assim os componentes têm de ser bastante resistentes para suportarem as doses a que estão sujeitos, durante o funcionamento.

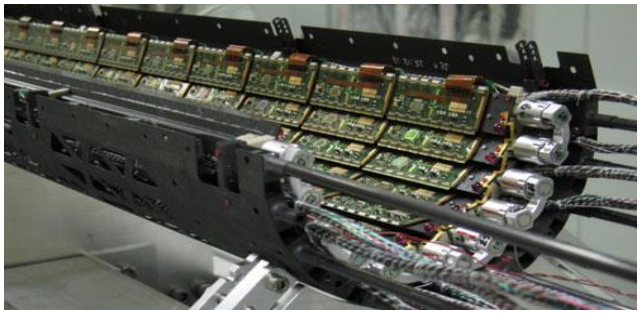

**Figura 2.3:** Detetor de pixéis do ATLAS

### 2.1.2. *Tracker* Semicondutor (SCT)

É o componente que se encontra no centro do detetor interno e está concebido para ter uma resolução bastante elevada na medição do momento das partículas. A região do barril do SCT permite obter pontos de elevada precisão em torno do plano R-φ (coordenadas cilíndricas) e ao longo da coordenada z, utilizando um ângulo pequeno.

### 2.1.3. *Tracker* de Radiação de Transição (TRT)

Consiste numa série de tubos de deriva proporcionais (palhetas) e tem como função identificar eletrões uma vez que essas palhetas têm no seu interior gás xénon, que quando se ioniza cria um padrão de palhetas atingidas que permite determinar o trajeto de uma partícula.

**Figura 2.4:** TRT à frente, SCT no fundo

## 2.2. Calorímetros

Um calorímetro [2] é um detetor que tem a tarefa de medir a energia das partículas que nele incidem. É um bloco de matéria, no qual as partículas interagem e transformam, parcial ou totalmente, a sua energia em quantidades mensuráveis proporcionais à sua energia incidente. O sinal só é produzido no chamado meio ativo do calorímetro (realizado com cristais, plásticos cintiladores, gases ionizantes ou líquidos, entre outros) do calorímetro e pode ser, como no caso dos calorímetros de árgon líquido, corrente elétrica, ou pode ser um sinal ótico, como no caso do detetor de telhas (*TileCal*). Existem dois sistemas de calorímetros:

- Calorímetro eletromagnético;
- Calorímetro hadrónico.

Ambos são calorímetros de amostragem, isto é, absorvem a energia e fazem amostras periódicas da forma da cascata de partículas resultante, inferindo a energia da partícula original a partir desta medição. Uma vez que se pretende que absorvam o máximo da energia depositada por uma partícula no menor espaço possível, limitando assim o tamanho do detetor, utilizam para o efeito materiais como o chumbo, o ferro, o tungsténio e o cobre.

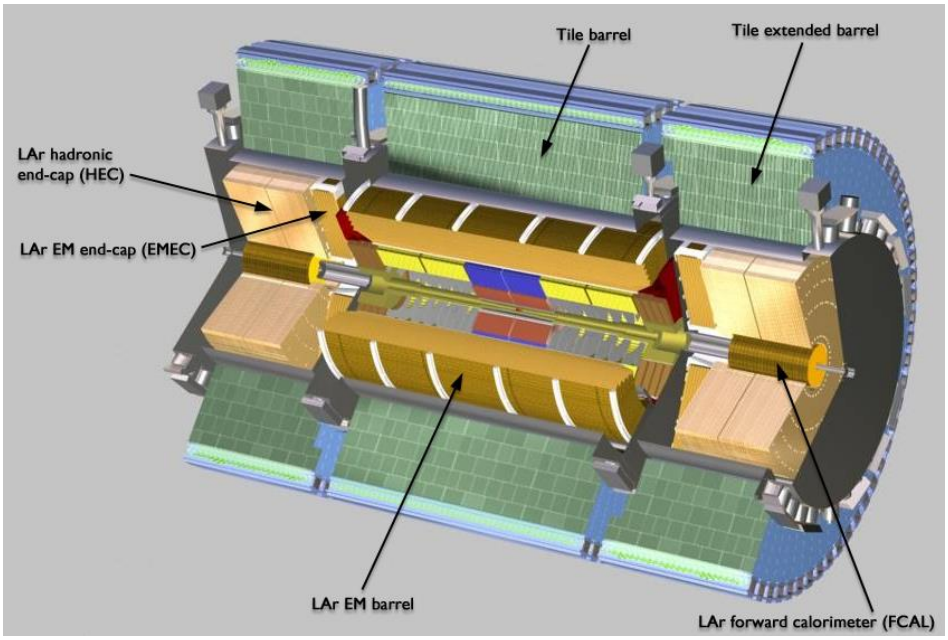


**Figura 2.5:** Calorimetria do ATLAS

### 2.2.1. Calorímetro Eletromagnético

O calorímetro eletromagnético [11] é um detetor de chumbo/árgon líquido, com elétrodos em forma de acordeão, e as placas absorvedoras são de chumbo. O árgon líquido, neste caso é o meio ativo, e é utilizado para fazer a amostragem.

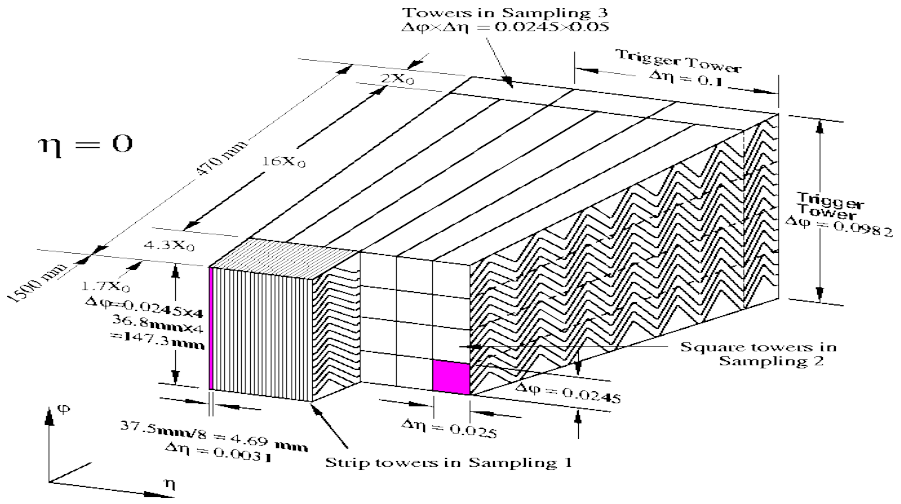


**Figura 2.6:**Ilustração do calorímetro eletromagnético

## 2.2.2. Calorímetro Hadrónico

A principal função do calorímetro hadrónico [8] é identificar, medir e reconstruir jatos de partículas. É um calorímetro de amostragem, cujo material de absorção é o ferro e utiliza placas cintilantes como meio ativo. Como o calorímetro hadrónico e o sistema de controlo são os objetos de estudo desta tese, serão abordados com mais detalhe adiante.

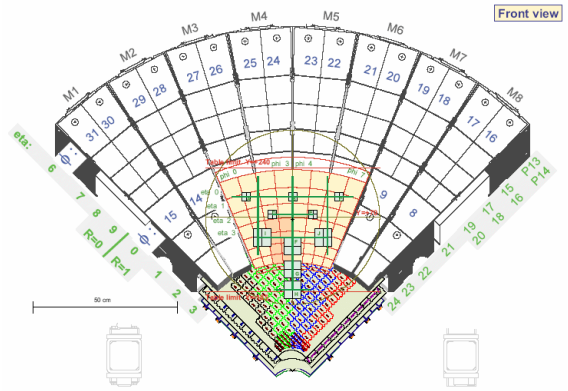


**Figura 2.7:** Ilustração do calorímetro hadrónico ou de telhas (*TileCal*)

# 2.3. Sistema Magnético

O ATLAS é caracterizado por dois sistemas de campo magnético [27] diferentes necessários para identificar partículas e efetuar medições de momentos. Um desses sistemas é constituído por um íman solenoidal supercondutor que gera um campo magnético axial de 2T, que envolve os *trackers* do detetor interno. O outro sistema é constituído por três ímanes em forma de anel, cujo núcleo está envolto em ar, e gera um campo magnético com uma intensidade de 1,5 T. Este segundo campo magnético concede ao espectrómetro de muões uma maior resolução, uma vez que está colocado fora da zona onde estão os calorímetros.

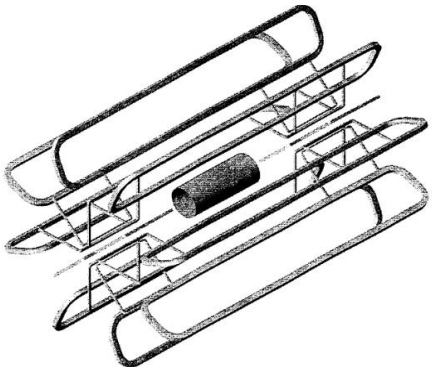

**Figura 2.8:** Representação do sistema magnético a 3 dimensões

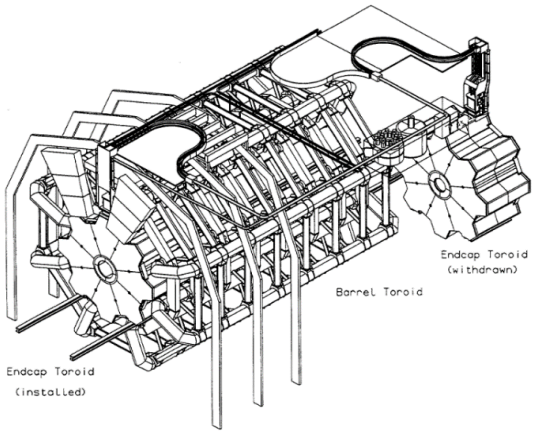


**Figura 2.9:** Sistema magnético incorporado no detetor ATLAS

## 2.4. Espectrómetro de Muões

O espectrómetro de muões [12] do ATLAS é baseado na deflexão das trajetórias descritas pelos muões no campo magnético gerado pelos ímanes em forma de anel supercondutores.

Na região do barril, as câmaras em torno do eixo do feixe estão dispostas em três camadas cilíndricas. Nas regiões de transição e nas extremidades, as câmaras são instaladas verticalmente, também em três camadas.

Esta disposição permite que as partículas atravessem sempre três câmaras, a partir do ponto de interação com o detetor, e estão otimizadas para medir com a máxima precisão, o momento dos muões.

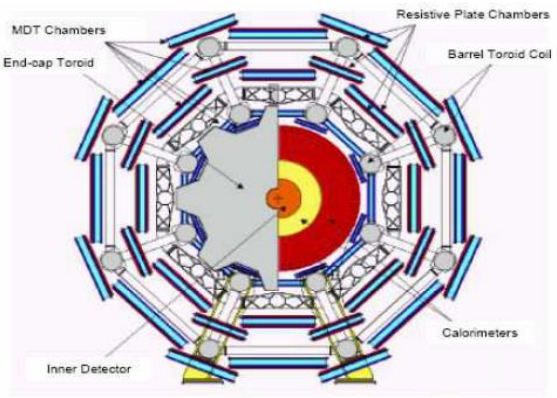


**Figura 2.10:** Espectrómetro de muões do ATLAS

## 2.5. Sistemas de Controlo

Existem três sistemas principais necessários ao funcionamento coerente dos diferentes subdetetores, e são eles o sistema de *Trigger*, o sistema de Aquisição de Dados (DAQ) e do Sistema de Controlo do Detetor (DCS). Em seguida apresenta-se uma descrição muito breve dos sistemas de *Trigger* e de aquisição de dados (TDAQ). Quanto ao DCS, será abordado mais à frente, visto ser o ponto principal desta tese.

## 2.6. Sistemas de *Trigger* e de Aquisição de Dados

O sistema de *Trigger* [14] do ATLAS tem a responsabilidade de escolher quais os eventos que possam ser considerados como "interessantes", do ponto de vista da Física, para o ATLAS. Para que um determinado evento seja classificado como "interessante", tem de ser definido pelo ATLAS e identificado pelo sistema de *Trigger* do detetor. A assinatura de eventos “interessantes” no sistema de *Trigger* é conseguida usando determinadas situações físicas, tais como: leptões carregados com baixo e alto momento, cascatas com elevado momento, originadas por quarks e gluões, entre outros.

A informação necessária para se chegar a estas situações físicas, provém dos vários subdetetores e funciona em três níveis de seleção de eventos:

- Nível 1 tem a função de reduzir a taxa de eventos de 40 MHz para 75 KHz.
- Nível 2 reduz esta taxa para 1 KHz.
- Nível 3, também chamado de filtro de eventos, finaliza o processo reduzindo a frequência de eventos para 10 Hz.

Cada nível de *trigger* refina a decisão tomada no nível anterior e, se necessário, aplica critérios de seleção adicionais. Um esquema simplificado do sistema de *trigger* do ATLAS é apresentado na Figura 2.11.

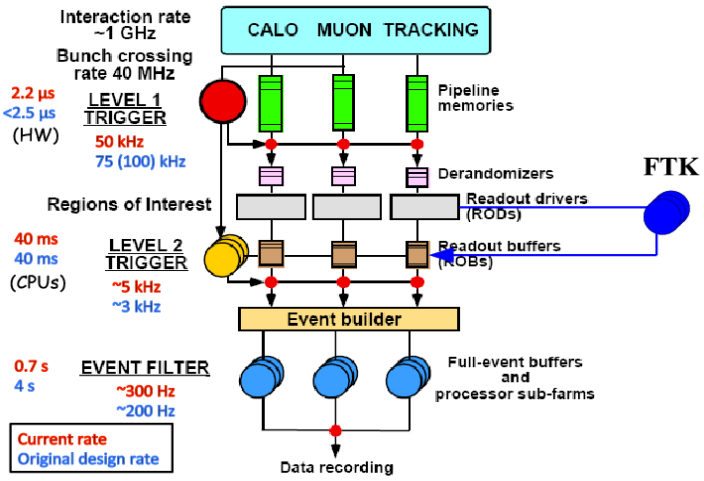


**Figura 2.11:** Sistema de *trigger* do ATLAS

# 3. O Calorímetro Hadrónico *TileCal*

O *TileCal* [9], [26] (Figura 3.1) é um calorímetro hadrónico localizado na região central do detetor ATLAS. É um calorímetro baseado numa técnica de amostragem hadrónica onde telhas de plástico cintilador estão embebidas em placas de ferro que têm como função absorver a luz, e estão conectadas a fibras óticas WLS (WaveLenght Shifter).

O *TileCal* apresenta um novo conceito de calorímetro cintilador para hadrões [6], no qual as telhas são colocadas perpendicularmente ao feixe do LHC e são escalonadas em profundidade. Esta deposição tem como grande vantagem facilitar a leitura do sinal luminoso através das fibras óticas WLS, mantendo uma boa hermeticidade a um baixo custo. Existem várias secções deste tipo de fibras agrupadas em células, em que cada célula é lida por dois PMTs (PhotoMultiplier Tube).

A sua principal função é a reconstrução, em energia, das cascatas de partículas produzidas devido às interações protão-protão, e a identificação de muões com baixo momento. Devido aos requisitos de funcionamento, o calorímetro enfrenta alguns problemas que têm implicações no seu desempenho, tais como o nível de radiação a que está sujeito, o efeito dos dois campos magnéticos, o campo magnético interno gerado pelo íman solenoidal, e o campo magnético externo gerado pelos ímanes anelares.

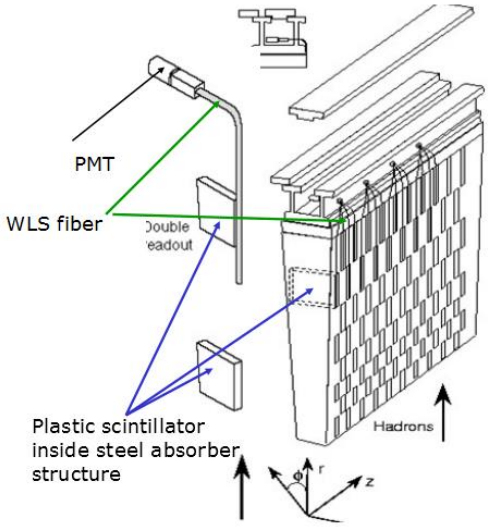


**Figura 3.1:** Constituição do *TileCal*

## 3.1. Princípios de Calorimetria

Um calorímetro [2] é um detetor que mede a energia de partículas de alta energia. É basicamente um bloco de matéria com a qual as partículas interagem e transformam parcial ou totalmente a sua energia numa quantidade mensurável. Existem dois tipos de calorímetros:

- Calorímetros homogéneos, constituídos apenas por uma região, e essa região é a zona ativa;
- Calorímetros de amostragem, compostos por duas regiões, uma ativa e uma passiva. A região ativa é constituída por um material cintilador, que pode ser sólido, líquido ou gasoso, e a região passiva em qual é feita de um material com um número atómico elevado, tal como o urânio, chumbo, cobre ou ferro, pois serve para travar as partículas.

A região ativa de um calorímetro de amostragem tem como função produzir o sinal, enquanto que a região passiva tem a função de travar as partículas no menor percurso possível e para tal utilizam-se elementos com elevado número atómico. Tal como já foi referido anteriormente, na experiência ATLAS existem dois sistemas de calorímetros:

- Calorímetro eletromagnético;
- Calorímetro hadrónico.

Os calorímetros eletromagnéticos são concebidos para medir principalmente eletrões e fotões, enquanto que o calorímetro hadrónico serve para medir hadrões simples ou cascatas de hadrões. Designa-se por cascata, um elevado número de partículas (hadrões nesta situação) produzidas através da interação forte[1]. O calorímetro eletromagnético está colocado imediatamente antes do calorímetro hadrónico, e isto acontece porque os hadrões têm maior comprimento de interação[2] dando origem a cascatas muito mais amplas, do que as produzidas por eletrões ou fotões.

## 3.2. Arquitetura do *TileCal*

O calorímetro *TileCal* (Figura 3.2) tem uma estrutura [5] cilíndrica oca com um raio interior de 2,28 m e um raio exterior de 4,230 m (onde se localizam os PMTs), e está dividido em três seções, um barril central (*Tile barrel*) (dividido em 2 partições de células de telhas LBA e LBC Figura 3.3), e duas seções de barril estendido (*Tile extended barrel*) (também dividido em 2 partições de células EBA e EBC, Figura 3.3) , sendo que este último pode mover-se. Envolve ainda o calorímetro eletromagnético de árgon líquido e as extremidades dos calorímetros eletromagnético e hadrónico.

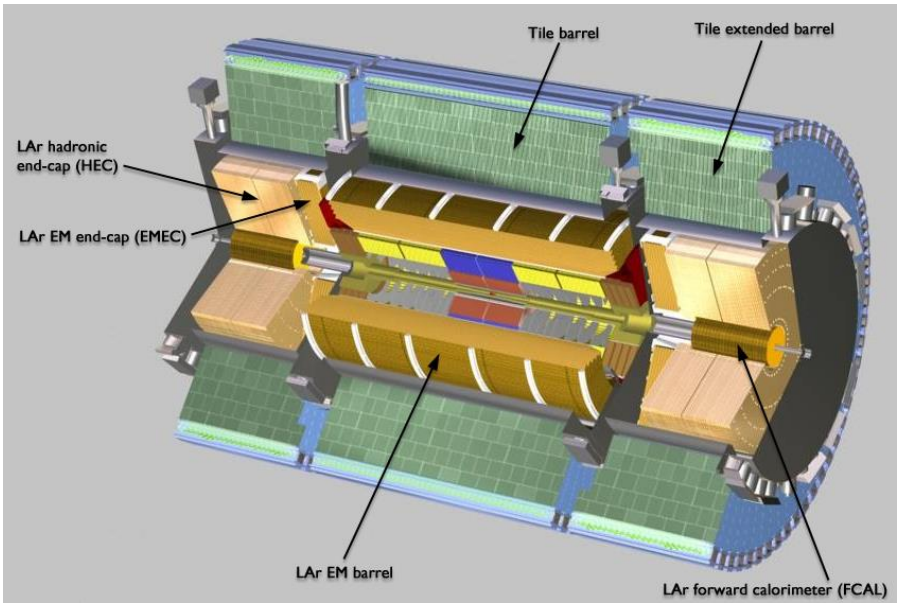


**Figura 3.2:** Ilustração dos barris do *TileCal*

[1] A interação forte é o mecanismo responsável pela interação nuclear forte, mediada por gluões, e mantém os quarks confinados em hadrões, dando origem aos protões e neutrões.
[2] Comprimento de interação é definido como a distância média que um hadrão de alta energia consegue viajar num meio antes de ocorrer uma interação nuclear.

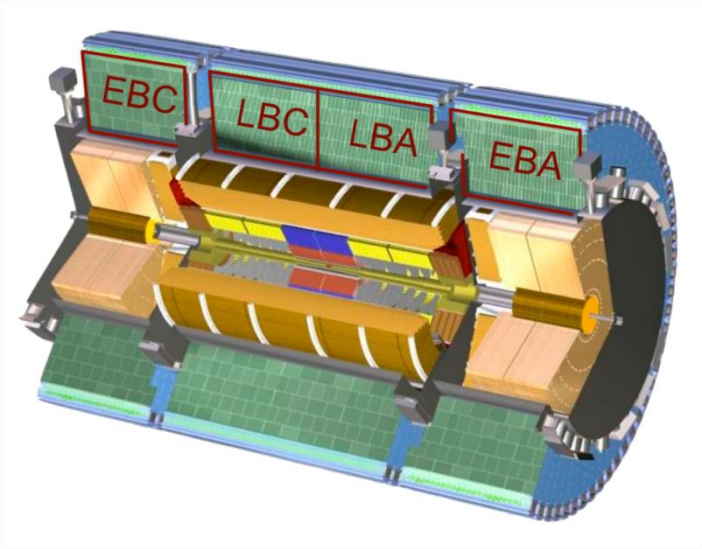


**Figura 3.3:** Ilustração da disposição das células de telhas do *TileCal*

O barril central e os barris estendidos estão separadas por um intervalo de cerca de 700 mm, necessário devido aos cabos que transportam o árgon líquido e à eletrónica do detetor interno. Cada uma das suas secções tem 64 módulos independentes, colocados ao longo da direção azimutal, e os módulos do calorímetro estão segmentados na direção axial (ao longo de z) e na direção longitudinal (ao longo de r). No seu interior, em cada módulo, existem várias estruturas de células [7] (Figura 3.4).

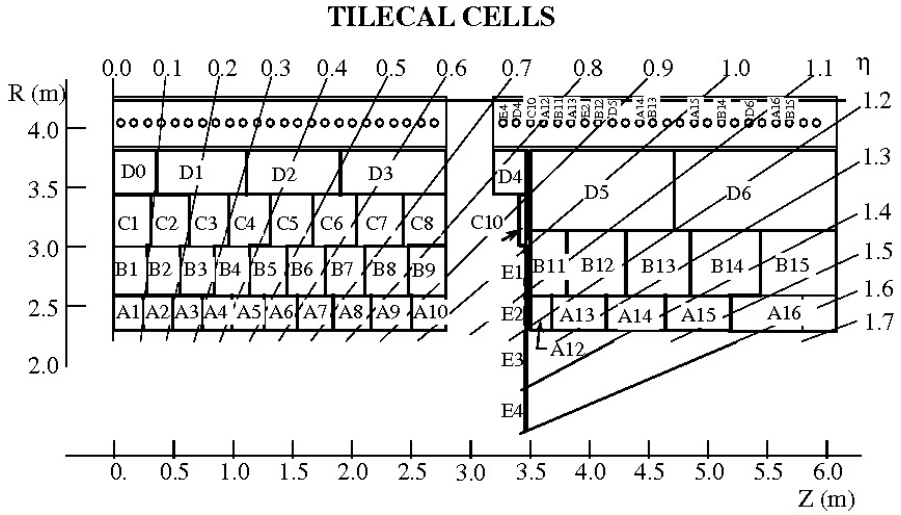


**Figura 3.4:** Ilustração das células de um módulo do ATLAS

No total, o *TileCal* tem aproximadamente 5000 células e 10000 canais (PMTs).

## 3.3. Fibras óticas WLS

A luz produzida por partículas que atravessam as telhas cintilantes propaga-se através dessas telhas até às suas extremidades, onde é absorvida pelas fibras WLS e é posteriormente guiada até aos PMTs. Existe um misturador de luz que está colocado entre os agrupamentos de fibras e um fotocátodo para homogeneizar a recolha de luz sobre a superfície do PMT.

Uma fibra WLS é um tipo de fibra ótica dopada com um material fotofluorescente que absorve fotões de frequência elevada e emite fotões com frequência mais baixa. Na maioria dos casos, um material deste tipo, quando absorve um fotão de alta energia, emite vários fotões de energia mais baixa. Na Figura 3.5 apresenta-se a instrumentação de fibras óticas WLS num dos barris do *TileCal*.

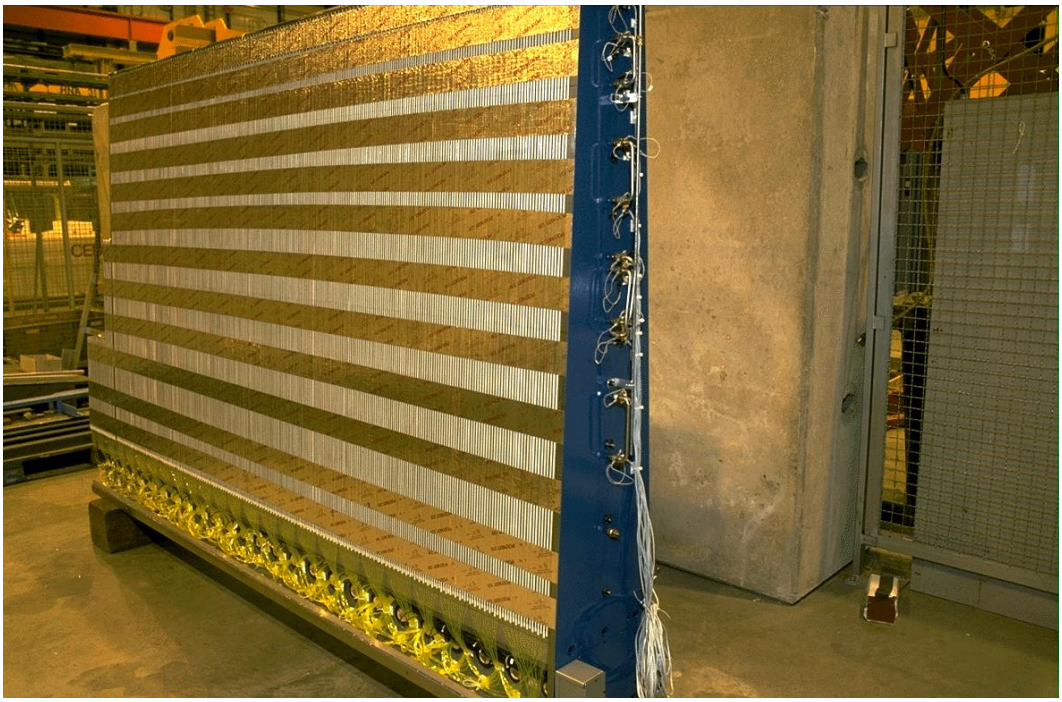

**Figura 3.5:** Módulo do calorímetro com as fibras WLS

O número total de fibras que equipam os 64 módulos do barril central e os 128 módulos de cada barril estendido é de cerca de 570 000. Na região do barril central, o comprimento das fibras varia entre 85 cm e 210 cm, e no barril estendido o seu comprimento varia de 90 cm a 230 cm. O comprimento total de fibra necessário para fazer o *TileCal* é de 1120 km.

## 3.4. Cintiladores

Um cintilador é um material que tem a capacidade de emitir luz quando radiação ionizante incide sobre si, uma vez que absorve a energia dessa radiação. Os cintiladores utilizados no *TileCal* são de poliestireno, um polímero, cujo pico de emissão de luz se situa nos 420 nm.

Para equipar os 192 módulos do *TileCal* do ATLAS foram produzidas meio milhão de telhas cintilantes, de 11 tamanhos trapezoidais diferentes. As dimensões das telhas variam desde 200 mm até 400 mm em comprimento, e de 100 mm a 200 mm de largura, todas com 3 mm de espessura. O peso total das telhas é cerca de 60 toneladas.

## 3.5. Fotomultiplicadores

Os fotões produzidos pela interação das partículas ao longo do cintilador são recolhidos e guiados pelas fibras WLS até à janela do PMT e interagem com o fotocátodo do PMT onde são convertidos em fotoeletrões devido ao efeito fotoelétrico.

Os eletrões produzidos no fotocátodo são então focados numa série de elétrodos, denominado dínodos, que multiplicam os eletrões numa emissão secundária. Os eletrões multiplicados são em seguida, recolhidos pelo o ânodo e a leitura é efetuada. A Figura 3.6 é uma ilustração do funcionamento geral de um PMT.

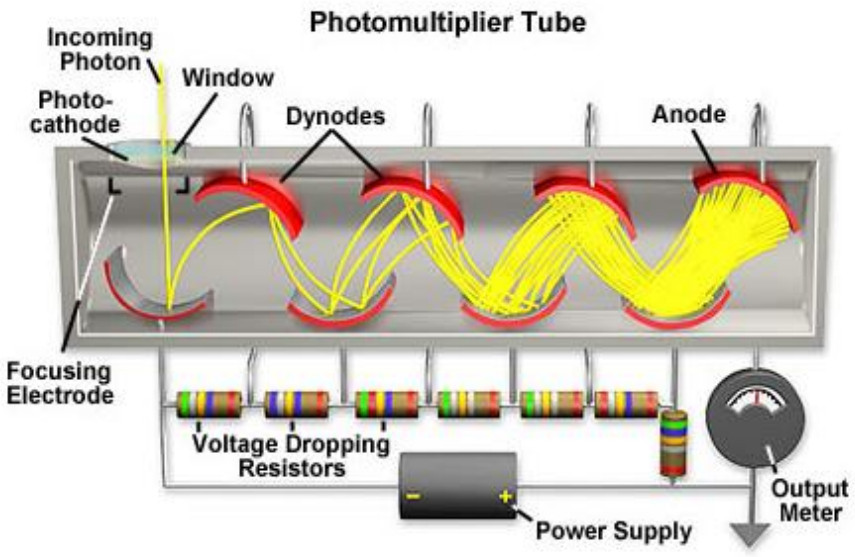


**Figura 3.6:** Ilustração do funcionamento de um PMT

O processo de multiplicação é conseguido aplicando uma alta tensão entre o primeiro e último dínodo. Os dínodos estão dispostos numa estrutura linear, em que os eletrões são progressivamente focados e multiplicados. Este tipo de disposição minimiza o tempo de trânsito dos eletrões entre os estágios de multiplicação, permitindo obter uma resposta muito rápida.

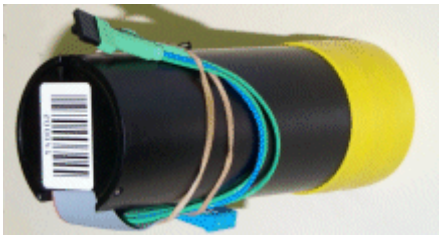

**Figura 3.7:** Exemplo de PMT

Um bloco PMT, como o da Figura 3.7, é composto por um tubo fotomultiplicador, um misturador de luz, um divisor de alta tensão, e um integrador.

O funcionamento dos PMTs é crucial, sendo necessária a sua constante monitorização, e para tal este sistema utiliza um laser, com o propósito de controlar o ganho de cada PMT em cada módulo. O sistema envia a cada PMT pulsos de luz, com a mesma duração e intensidade, em comparação com os pulsos produzidos por partículas no cintilador, permitindo isolar efeitos puramente devido ao PMT, tais como desvios no ganho ou na eletrónica associados ao sistema de leitura.

## 3.6. Sistema de leitura

A eletrónica de leitura rápida [10], [13] confere forma, amplifica e digitaliza os sinais provenientes dos PMTs do *TileCal* que depois são amostrados por um ADC de 10 bits, no LHC, a cada intervalo de 25 ns.

Os sinais provenientes das 2000 "*trigger* towers" devem ser empacotados e enviados para a eletrónica do sistema de *trigger* de nível 1 (sistema utilizado para selecionar eventos) a cada 25 ns. Estes

sinais sofrem um atraso temporal (latência) até cerca de 2,5 μs, tempo necessário para que o sistema de *trigger* aceite os dados e os envie à eletrónica de FE.

Toda a eletrónica de FE e os fotomultiplicadores estão localizados na parte de trás dos calorímetros, num sistema de gavetas (*drawers*) removíveis. Estas gavetas são compostas por duas mini gavetas que apresentam metade do número de PMTs (mini *drawer*), cada, de uma gaveta. Uma gaveta é necessária para fazer a leitura de cada módulo do barril estendido, e são necessárias duas mini gavetas para cada módulo do barril central.

Cada gaveta pode conter até 48 blocos de PMTs e também contém placas eletrónicas que fornecem os níveis de alta tensão necessários ao funcionamento dos destes PMTs, e que ainda processam os sinais por estes produzidos.

Como já foi dito anteriormente, o número total de PMTs é de cerca de 10000, e as placas eletrónicas de leitura estão ligadas em cada gaveta a uma motherboard, que se encontra disposta na parte superior e na parte inferior de cada gaveta, como mostra a Figura 3.8.

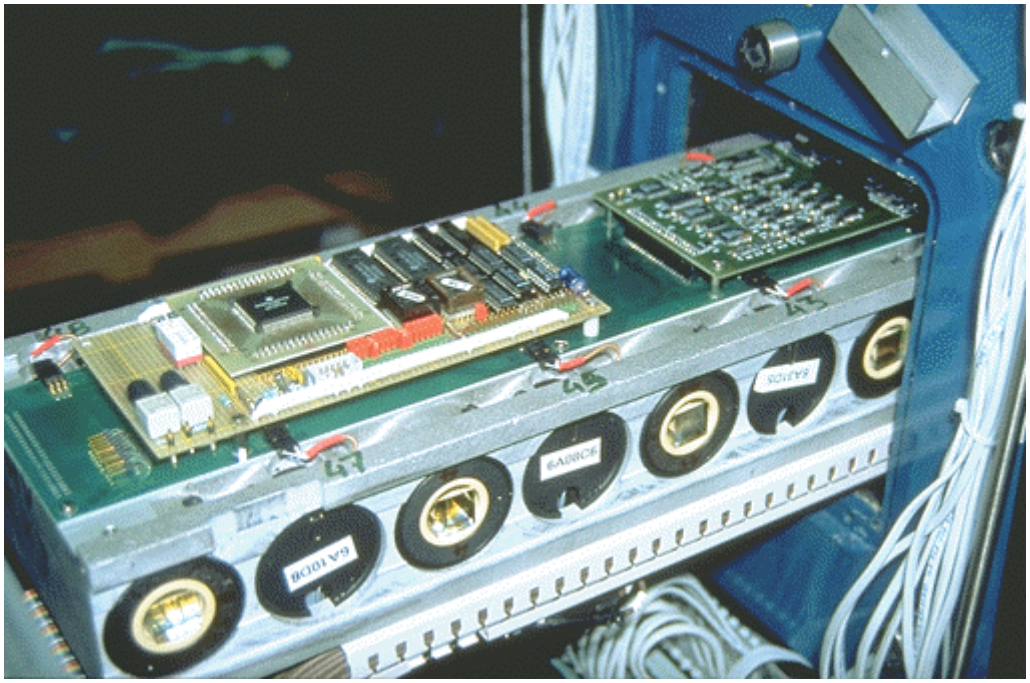

**Figura 3.8:** Gaveta do *TileCal*

## 3.7. Conclusão

Neste capítulo foi apresentado uma breve introdução da arquitetura do calorímetro hadrónico *TileCal* e dos seus blocos constituintes, começando pelas telhas (matéria) com as quais as partículas interagem, a forma como o sinal é conduzido (via fibras WLS) até aos PMTs e finalmente analisado pelo sistema de leitura onde é feita a reconstrução do sinal em energia das várias partículas que incidiram no calorímetro.

# 4. Enquadramento do DCS no *TileCal*

O DCS [3], [21], [24] faz todo o controlo, monitorização e operação de cada partição do *TileCal*, e do seu ponto de vista, todas estas operações são iguais. Mesmo nos casos em que cada detetor é responsável pela organização interna dos seus próprios subsistemas, devem ser cumpridos os requisitos definido pelo DCS do ATLAS. Isto implica que o DCS do *TileCal* deve seguir a arquitetura do DCS do ATLAS, tanto quanto possível, a não ser que haja determinados impedimentos devido a especificações locais.

## 4.1. Breve descrição do sistema de controlo (DCS)

A principal tarefa do Sistema de Controlo do Detetor ATLAS ("*Detector Control System*", o sistema de controlo de alto nível da experiência ATLAS) é permitir o funcionamento correto do detetor. Todas as ações realizadas pelo utilizador e todos os erros, avisos e alarmes relativos ao hardware do detetor são monitorizadas pelo DCS. A arquitetura do DCS do ATLAS consiste num sistema distribuído de *Back-End* (BE) em computador e em vários sistemas de *Front-End* (FE). A função do software de BE é adquirir dados da eletrónica de FE e oferecer funções de controlo e supervisão.

O sistema DCS está organizado hierarquicamente em três níveis: estações de controlo globais (GCS), estações de controlo de subdetetor (SCS), e estações de controlo locais (LCS). Esta hierarquia permite que a experiência esteja dividida em partições independentes que têm a capacidade de operar em modo autónomo ou integrado.

No sistema que se encontra em funcionamento, a comunicação entre o DCS e as placas *HV-Opto* é feita através de uma placa dedicada que inclui um microprocessador, denominada HV Micro, que neste momento está obsoleta, razão pela qual tem de ser substituída por uma opção mais moderna e rápida.

## 4.2. Arquitetura do DCS do *TileCal*

A estrutura lógica [20], [23] do DCS do *TileCal* está subdividida em blocos funcionais, seguindo uma sequência de critérios funcionais, estruturados em forma de árvore onde todos os blocos funcionais são executados autonomamente, como se pode ver na Figura 4.1. O sistema apresenta os seguintes blocos funcionais:

- Sistema de alta tensão, necessário para o funcionamento dos PMTs;
- Sistema de baixa tensão, necessário para a eletrónica de leitura e para regular a alta tensão;
- Sistema de arrefecimento que monitoriza as temperaturas dentro das gavetas;
- Sistemas autónomos do *TileCal* que contêm os seus próprios sistemas de controlo independentes, embora com interface para o DCS do *TileCal* para troca de dados, para garantir que o detetor opera em segurança.

A implementação da estrutura lógica segue a arquitetura do DCS do ATLAS o que significa que o DCS do *TileCal*, como um subdetetor do ATLAS, tem a sua própria estação de controlo de subdetetores (SCS), o que permite que opere todos os componentes do subdetetor e de uma forma completa e independente.

A ligação ao sistema TDAQ, ao sistema de calibração e à infraestrutura do detetor tem lugar ao nível da SCS, para garantir que o funcionamento do detetor está sincronizado com a tomada de dados. No nível inferior da hierarquia temos o LCS que lida com a monitorização e controlo dos sistemas de alta e baixa tensões do subdetetor. O LCS também é responsável por executar os comandos recebidos do SCS.

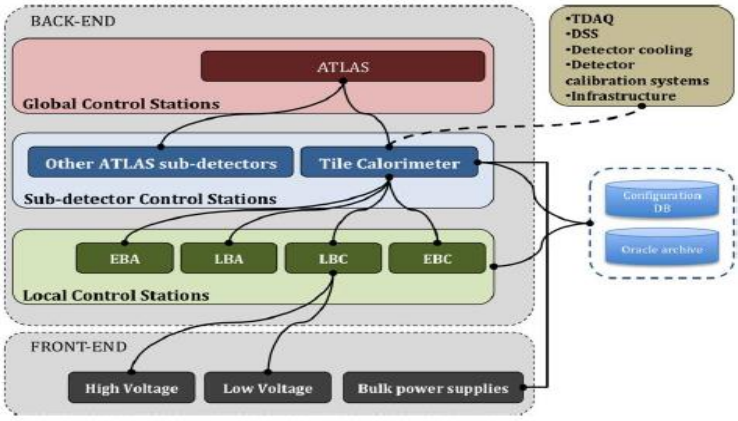


**Figura 4.1:** Hierarquia do DCS do *TileCal*

## 4.3. Sistema de distribuição de alta tensão

A principal função do sistema de distribuição de alta tensão [30] é definir a alta tensão de todos os fotomultiplicadores (PMTs) do calorímetro. Os principais requisitos para o sistema de alimentação de alta tensão são:

- Garantir uma estabilidade a curto e longo prazo da tensão fornecida (inferior a 0,5 V).
- Garantir baixo ruido no sistema de leitura (200 mV pico a pico).
- Garantir uma temperatura muito estável, a fim de minimizar a sensibilidade térmica dos PMTs.
- Garantir que o intervalo de tensão permitida para cada PMT individualmente está limitado ao intervalo entre 400 V e 900 V.

O *TileCal* é composto por 256 gavetas, com 45 PMTs em cada gaveta pertencente ao barril central do calorímetro e 32 PMTs para cada barril estendido. Na tabela seguinte é apresentado o número de PMTs utilizados no *TileCal*, por gaveta.

| Super-Drawer | Sampling | Number of PMTs | Number of PMTs |
|---|---|---|---|
| LB | A | 20 | 45 |
| | BC | 18 | |
| | D | 7 | |
| EB | A | 10 | 32 |
| | B | 10 | |
| | D | 6 | |
| | C | 2 | |
| | E | 4 | |

**Tabela 4.1:** Número de PMTs necessário por gaveta

No total, são necessários cerca de 10100 PMTs. Este valor é obtido da seguinte forma:

$$45\ \text{PMTs} \times 2\ \text{lados} \times 66\ \text{módulos} = 5940\ \text{PMTs}$$

que corresponde ao número de PMTs necessário para as gavetas dos LB, e

$$32\ \text{PMTs} \times 2\ \text{lados} \times 65\ \text{módulos} = 4160\ \text{PMTs}$$

que corresponde ao número de PMTs necessário para as gavetas dos EB.

Todas as gavetas são iguais do ponto de vista da alta tensão e cada uma destas gavetas é alimentada exclusivamente por um único canal de alta tensão. As tensões individuais aplicadas em cada PMT são definidas em pares dentro de uma gaveta por um distribuidor de tensão. A fim de alimentar todas as gavetas das 4 partições, existem 16 fontes de alimentação de alta tensão (*crates* HVPS (High Voltage Power Supply)), localizadas na caverna USA15 e distribuídas em duas prateleiras (*racks*).

O sistema distribuidor de alta tensão, do ponto de vista do DCS, é composto principalmente por dois dispositivos:

- As placas divisoras da gaveta que estão localizados dentro dos módulos do *TileCal*;
- Fonte de alimentação (*Bulk Power Supply* 800 V ou PS 800 V), cuja tarefa principal é fornecer as tensões necessária para o funcionamento dos PMTs e está localizada na caverna USA15.

As gavetas ainda se podem dividir em dois dispositivos, a Placa *HV Micro* e a Placa *HV-Opto*.

Em seguida será apresentada uma descrição dos quatros sistemas mencionados.

## 4.4. Fonte de alimentação (800V PS)

A fonte de tensão 800V PS tem como função fornecer o valor de alta tensão à gaveta de alta tensão. Cada canal desta fonte pode fornecer dois valores de saída: HVout=-830 V ou HVout=-950 V, com um consumo nominal de 11mA (a corrente de saída máxima é 20mA). Os dispositivos destas fontes de alta tensão estão localizados nas prateleiras da caverna USA15, cada unidade tem 16 canais de saída e cada canal alimenta uma gaveta. O protocolo de comunicação utilizado é o Modbus/RTU.

## 4.5. Placa *HV-Opto*

A função da placa de *HV-Opto*, da Figura 4.2, é ajustar a tensão aplicada aos conjuntos de 24 PMTs.

**Figura 4.2:** Placa *HV-Opto*

Cada gaveta é controlada por duas placas deste tipo, e estas placas apresentam as seguintes características:

- Tem a possibilidade de ligar/desligar os canais pares ou ímpares para cada conjunto de 24 PMT;

- Incluem 24 malhas de regulação que são realizados usando um acoplador ótico de alta tensão (MOTOROLA MOC8204), que à saída apresenta uma alta tensão, cujo valor varia no intervalo (HVin-360 V; HVin), com HVin=-830 V ou HVin=-950 V;

- Inclui 6 conversores digital-analógico (DAC), contendo 4 canais de 12 bits (MAXIM MAX536) que aplicam os valores definidos nas 24 malhas de regulação;

- Inclui 1 conversor analógico-digital (ADC), (ADS 7820) multiplexador com 32 canais de 12 bits.

- Inclui uma memória EEPROM[3] de 2 KB (AT25160) para registo dos parâmetros necessários para a configuração das placas.

Os comandos enviados para regular as altas tensões individuais são realizados em vários passos. O primeiro passo começa com a leitura, realizada pelo microcontrolador dos parâmetros de configuração da placa, que estão armazenados na sua memória RAM (Random Access Memory) interna. Após a conclusão da leitura, um conjunto de comandos associados às tensões são enviados para o DAC de 12 bits, que define os valores das tensões num ciclo de regulação. Este ciclo amplifica a diferença de tensão e envia os comandos para o díodo do acoplador ótico que, finalmente, regula a alta tensão.

## 4.6. Placa *HV-Micro*

A função da placa HV Micro é controlar as duas placas *HV-Opto* existentes em cada gaveta. Todas as comunicações entre esta placa e o DCS são feitas com o protocolo de comunicação CANbus. A placa *HV-Micro* controla 68 canais de 5 tipos diferente:

- Altas tensões de 48 PMTs;

[3] EEPROM é um tipo de memória, não volátil, utilizada para guardar pequenas quantidades de dados.

- 4 altas tensões de entrada, que correspondem a um quarto de uma gaveta, alimentando 12 PMT cada;

- 7 canais de temperatura;

- 7 canais de baixa tensão;

- 2 canais de ADC.

Os principais componentes da placa *HV-Micro* são uma memória Flash, um microcontrolador (MC68376), uma EEPROM, uma memória RAM de 256 KB e uma interface para CANbus, como se pode ver na Figura 4.3.

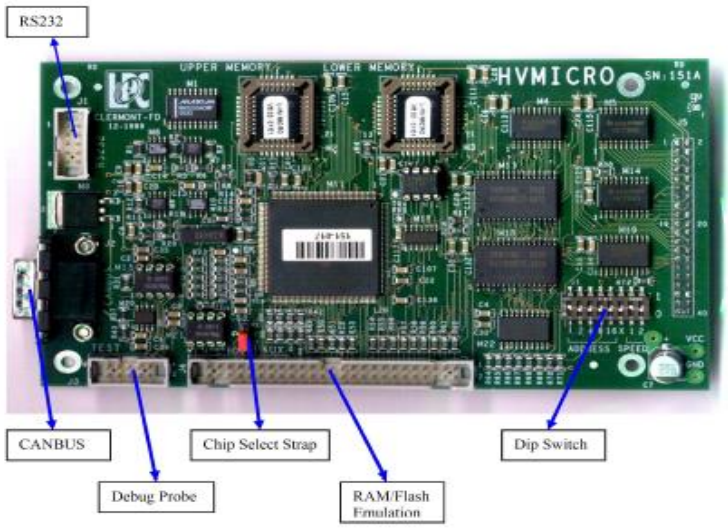


**Figura 4.3:** HV Micro

## 4.7. Sistema de distribuição de baixa tensão

A função do sistema de baixa tensão do *TileCal* é a de alimentar a eletrónica dos distribuidores de alta tensão e a leitura de FE. Ambos os sistemas estão dentro das gavetas, e para evitar interações entre ambos, são alimentados por conversores de baixa tensão diferentes.

O sistema de alimentação de baixa tensão é um sistema de duas fases. Na primeira fase, converte a tensão de entrada de 400 V AC numa tensão de saída de 200 V DC, na caverna USA15. A segunda fase é feita no detetor e consiste em converter a tensão de 200V DC para 8 níveis independentes de tensões, na gama de -15 V a +15 V.

Do ponto de vista do DCS, o sistema é composto por três dispositivos:

- A fonte de alimentação “dedo” de baixa tensão (fLVPS (finger Low Voltage Power Supply)), está localizada num lugar chamado “dedo”, que fica próxima da eletrónica de FE do detetor;

- A placa auxiliar (placa AUX), cuja tarefa principal é fornecer as tensões necessárias para a operação dos fLVPS. Encontra-se na caverna USA15;

- As fontes de alimentação que fornecem os 200 V DC necessários para alimentar os fLVPS e também estão localizados na caverna USA15.

## 4.8. Fonte de alimentação (PS 200 V)

A fonte de tensão de 200 V tem a função de converter a tensão de entrada de 400 V AC numa tensão de saída de 200 V DC, que é usada para alimentar os dispositivos fLVPS. Os dispositivos da PS 200 V estão localizados nas prateleiras na caverna USA15 e cada unidade tem 3 canais de saída e cada canal alimenta 4 fLVPS. A tensão de saída nominal para cada canal individual é de 200 V, com um consumo nominal de corrente entre 4 A e 5 A. O protocolo de comunicação utilizado é também o Modbus/RTU.

## 4.9. Fonte de alimentação "dedo" de baixa tensão (fLVPS)

Estas fontes de alimentação, como a da Figura 4.4; funcionam como um conjunto de conversores DC-DC, estão localizadas no *TileCal*, próximas da eletrónica de FE.

**Figura 4.4:** Fonte de alimentação "dedo" de baixa tensão

Têm como função converter os 200 V DC provenientes da PS 200 V em oito tensões de saída diferentes no intervalo de -15 V a +15 V, com uma flutuação relativa máxima de <0,01%. Os conversores estão agrupados em dois grupos:

- Um grupo alimenta o sistema distribuidor de alta tensão;
- O outro grupo alimenta o resto da eletrónica de FE.

## 4.10. Placa Auxiliar (Aux Board)

A placa auxiliar (Aux Board), da Figura 4.5, tem a função de fornecer as tensões operacionais à ELMB e à motherboard dos fLVPS e, ao mesmo tempo fornecer as malhas de corrente que permitem ativar ou desativar as tensões de saída dos fLVPS. Estão localizadas na caverna USA15 e cada placa auxiliar pode alimentar quatro fLVPS e duas malhas de corrente para cada grupo individual de fLVPS. Em resumo, a funcionalidade da placa auxiliar é a seguinte:

**Figura 4.5:** Placa auxiliar

- Alimentar a parte de entrada analógica (AI) da ELMB e da motherboard colocado dentro dos fLVPS;

- Fornecer duas malhas de corrente independentes para cada um dos fLVPS, que ativam ou desativam as tensões de saída dos dois grupos.

- Fornecer um impulso inicial, de curta duração (cerca de ~1 s), necessário para ligar os conversores. Após os conversores estarem ligados, o controlo é feito pelas malhas de corrente.

## 4.11. Sistema de arrefecimento

O consumo total, em potência, por módulo é de ~300 W, e assim, se as temperaturas das gavetas forem mantidas a 18 ºC, com este consumo total em potência, as temperaturas dos chips situam-se entre 40 e 50 ºC, que é considerada uma boa temperatura de funcionamento. Nesta situação, um sistema de arrefecimento sem fugas, que funciona a água a uma pressão inferior à pressão atmosférica pode ser utilizado para manter esta temperatura nas gavetas. Em suma, os requisitos gerais do sistema de arrefecimento são:

- *Dissipação de potência*: o sistema de refrigeração deve ser capaz de dissipar 300 W durante um longo período de tempo, mantendo a temperatura no interior com variações de cerca de 0,5 ºC, a fim de assegurar uma temperatura constante da eletrónica.

- *Uniformidade*: a temperatura ao longo de uma gaveta deve ser uniforme. Para monitorizar a temperatura no interior das gavetas, seis sondas de temperatura estão instaladas dentro de cada gaveta, quatro perto da placa do microprocessador e as duas restantes dentro de cada gaveta, em lados opostos.

- *Segurança*: fugas no sistema de refrigeração podem causar danos permanentes no sistema eletrónico.

O DCS do sistema de arrefecimento é baseado num Controlador Lógico Programável (PLC) que realiza um controlo apertado da temperatura e da pressão da água de refrigeração. Este sistema de controlo está ligado aos DCS através do protocolo de comunicação OPC. Existe ainda um sistema adicional de monitorização, utilizado para monitorizar de forma passiva os seguintes parâmetros:

- As temperaturas do líquido de arrefecimento ao longo dos tubos (gavetas exteriores);

- Temperaturas das sondas colocadas dentro de gavetas;

- Estados e alarmes dos dispositivos que integram a unidade de refrigeração.

O sistema de arrefecimento do *TileCal* está organizado em ciclos de refrigeração independentes, seis para cada partição com todas as malhas fornecidas por uma única planta de refrigeração. A responsabilidade do DCS do *TileCal* é monitorizar os parâmetros dessas malhas, sem poder efetuar qualquer ação direta sobre o sistema de arrefecimento. A comunicação com o PLC é conseguida utilizando um *gateway*[4].

## 4.12. Base de dados do DCS do *TileCal*

O *TileCal* DCS utiliza ambas as bases de dados do DCS do ATLAS, já referidas anteriormente, a base de dados de configuração (ConfDB) e a base de dado de condições (CondDB). O uso destas bases de dados é feito tanto pelo sistema de alta tensão, como pelo sistema de baixa tensão. O acesso à base de dados de configuração é limitado e é usado para armazenar os valores de calibração dos fLVPS do sistema de baixa tensão e também para armazenar os valores das tensões dos PMTs, necessários ao sistema de alta tensão.

A base de dados de condições também é utilizada nos sistemas de alta e baixa tensão e neste caso serve para armazenar as tensões aplicadas aos PMTs e medir as temperaturas dos dispositivos que se encontram dentro das gavetas, e serve também para guardar os dados dos fLVPS, da placa auxiliar e da PS 200 V.

## 4.13. TDAQ/DCS

A comunicação entre o TDAQ e o DCS no *TileCal* é bastante limitada e está confinada ao envio do estado global das partições e dos módulos da seguinte forma:

- Os estados globais do DCS do *TileCal* são enviados para o TDAQ através do pacote DDC_MT;

- Os estados globais de cada módulo são enviados para TDAQ através do pacote DCS2DAQ;

- O estado global do DAQ, bem como os dados são enviados para o *TileCal* através do pacote DAQ2DCS.

## 4.14. Interações com os sistemas de calibração

A interação do DCS do *TileCal* com os sistemas de calibração do *TileCal* é implementada tanto para o sistema de calibração de césio como para o sistema Laser. A interação com o sistema do laser é muito limitada e apenas existe um botão de emergência disponível ao nível da FSM. Em relação ao sistema de calibração de césio, a interação é bastante complexa, porque o sistema de calibração de césio fornece um novo conjunto de valores de tensões para todos os PMTs, em cada tomada de dados, utilizando uma fonte de césio (*cesium run*). Estes valores são baseados nas leituras da corrente de todos os PMTs, o que permite equalizar a resposta de todas as células do *TileCal* com uma precisão de ~1 %.

[4] O gateway é utilizado para interligar várias redes que utilizem protocolos de comunicação diferentes.

## 4.15. Máquina de estados finita do *TileCal*

O BE do DCS no ATLAS está organizado em três camadas funcionais e a FSM [28] é a ferramenta principal para a implementação da cadeia de controlo. A hierarquia de controlo é feita com base no *WinCC* e no *toolkit* SMI++, presente na Figura 4.6, fornece uma imagem abrangente do comportamento do *TileCal* ao operador do ATLAS, mostrando o estado atual dos sistemas de subdetetor do *TileCal*.

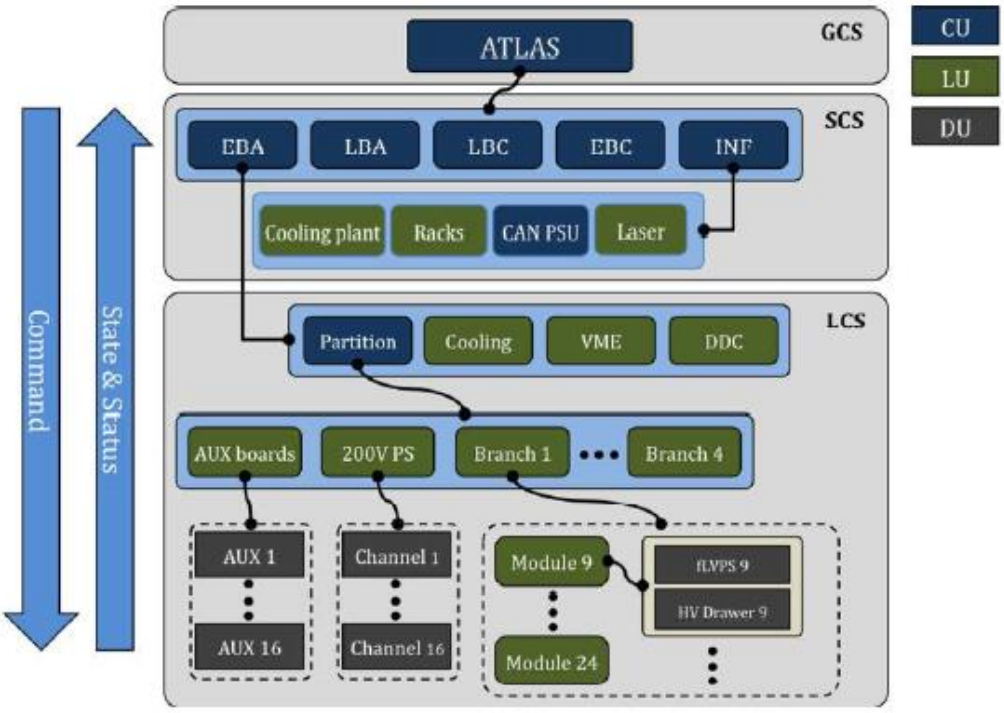


**Figura 4.6:** Ilustração da arquitetura da FSM do *TileCal*

Os elementos básicos da FSM são a Unidade de dipositivo (DU) e a Unidade de controlo (CU);

As unidades de dispositivo fornecem interface com o equipamento, enquanto as unidades de controlo integram as unidades de dispositivos na hierarquia. A FSM do *TileCal* vai incorporar 21 unidades de controlo e 600 unidades de dispositivos.

## 4.16. Alarmes

O ecrã de alarme usado no *TileCal* é o mesmo usado pelo ATLAS, que é implementado no ecrã de alerta JCOP. Como podemos ver na Figura 4.7, o ecrã de alarme exibe uma lista de alarmes ativos de todo o DCS, permitindo ao operador a identificação rápida dos elementos problemáticos e, se necessário, tomar as medidas corretivas necessárias.

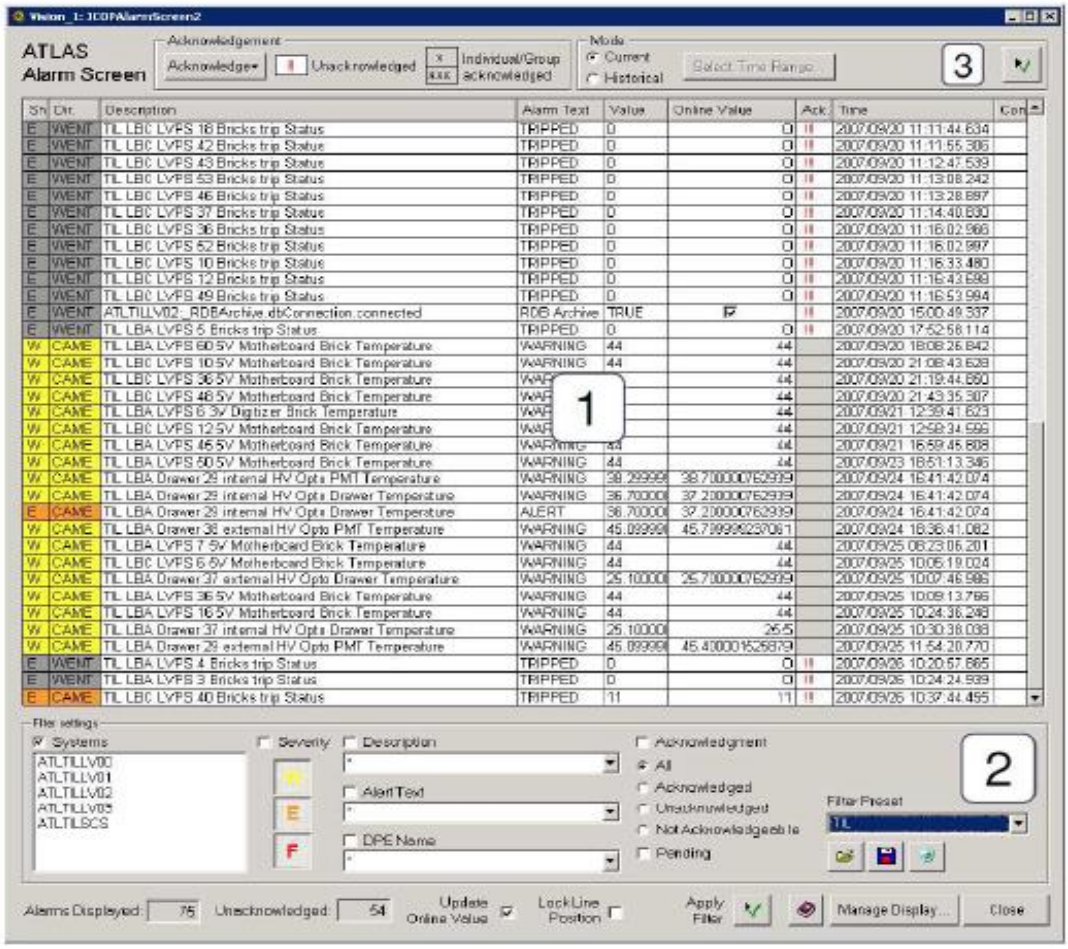


**Figura 4.7:** Ecrã de alarmes

Os elementos da interface do utilizador são:

1: Tabela de alarme e para cada alarme existe uma linha correspondente cuja ordem é a seguinte:

- Gravidade do alarme;
- Pequena descrição do alarme do detetor;
- Descrição do problema, apenas necessária se uma breve descrição não for suficiente;
- Valor atual e notificação de hora correspondente no momento do problema;
- Confirmação, se possível, dependente do utilizador;
- Marca de hora do alarme.

2: Configurações de filtro, isto é, é possível filtrar o ecrã de alarme com base nos seguintes parâmetros:

- Estação de controlo do subdetetor;
- Gravidade;

- Tipo de elemento;
- Tipo de confirmação.

3: Seleção de modo: permite mudanças entre dois modos de operação:

- Exibição atual de alarmes ativos;
- Visor histórico no qual os alarmes não ativos atualmente podem ser exibidos.

Durante períodos regulares de tomada de dados, sempre que um alarme é acionado, está associado a um determinado nível de gravidade, sendo por ordem do menos para o mais importante, “AVISO”, “ERRO” e “FATAL”. Os menos importantes dizem respeito a ações preventivas, e os mais importantes dizem respeito a interrupção da tomada de dados.

## 4.17. *WinCC*

O *WinCC* é um programa orientado para dispositivos, onde as variáveis são combinadas em pontos de dados hierarquicamente estruturadas (*datapoints*). Apresenta uma arquitetura distribuída, e cada aplicação é composta por vários processos, chamados “managers”. Estes “managers” comunicam entre si, com base no princípio cliente-servidor, usando o protocolo TCP/IP.

A arquitetura *WinCC* está centrada no “*Event Manager* (Figura 4.8)” que lida com a comunicação com todos os outros módulos, e vários projetos podem ser ligados via LAN para formar um sistema distribuído.

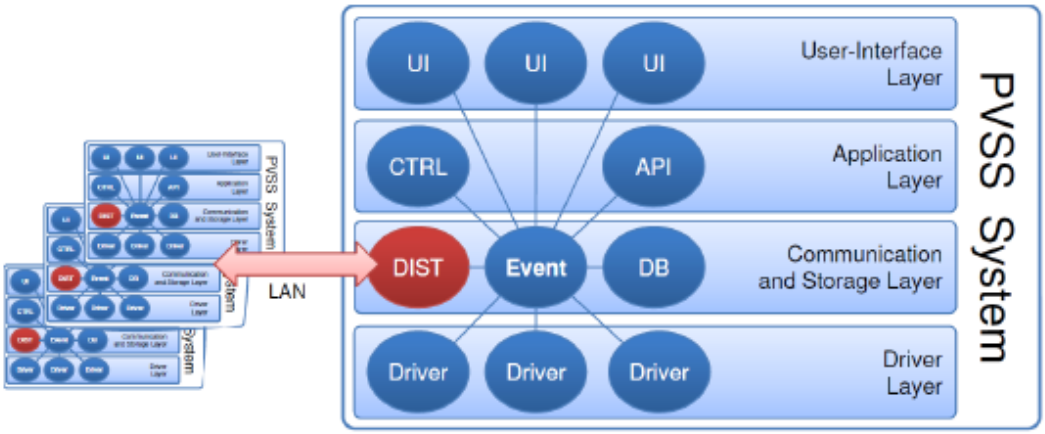


**Figura 4.8:** Arquitetura do "manager" do *WinCC*

- "*Event Manager*" é Responsável por todas as comunicações, incluindo a distribuição de dados. Recebe dados do sistema de FE e envia-os para o “*Database Manager”* para que possam ser armazenados na base de dados.
- “*Control Manager*” contém aplicações de controlo que monitorizam e controlam o detetor.
- “*User Interface Managers*”, tratam das interações externas, onde um utilizador pode obter os dados através da base de dados, ou descarregar dados na base de dados para serem enviados para os dispositivos. Pode também pedir para ser informado quando novos dados chegam.

- A interface FE (FEI), proporciona comunicações com o equipamento por meio de drivers dedicados ou padrões de comunicação.

- Os "*API Managers*", permitem que os utilizadores escrevam os seus próprios programas, (c++) usando a API (*Application Programming Interface*) do *WinCC* para aceder à base de dados.

# 5. Projeto e interface de controlo

Este trabalho tem como objetivo a criação de um sistema de controlo para as placas *HV-Remote*, e para tal é necessário o estudo da tecnologia de comunicação *Ethernet* e os protocolos associados, a implementação do protocolo TCP/IP e aplicações do tipo Cliente/Servidor (*Sockets*), e ainda a familiarização com os ambientes de programação *WinCC*, fundamental para a criação da interface de utilizador para a placa, e com o TIDE, utilizado para programar o módulo TIBBO (Figura 5.1), traduzindo-se numa experiência de constante aprendizagem.

**Figura 5.1:** Módulo programável TIBBO EM1206

**Figura 5.2:** Placa de testes TIBBO EM1206-EV

Além das várias partições que constituem o sistema final de controlo, é também necessário conhecer o sistema a controlar (*HV-Remote*). A placa que está a ser desenvolvida pelo grupo é composta por um nivelador de tensão, por três expansores digitais, por um ADC, por três DACs e um multiplexador, pelo que é necessária a leitura e compreensão dos *datasheets* de cada um destes integrados.

Para realizar este trabalho [31], [32] foi adquirida uma placa de testes TIBBO EM1206-EV, como a da Figura 5.2, com o intuito de servir de ligação entre a placa *HV-Remote* e o sistema de controlo (DCS) do detetor ATLAS, uma vez que comunica via *Ethernet*, com o DCS, e via protocolo SPI, com os componentes da placa *HV-Remote*, nomeadamente com o expansor digital de portas (MCP23017), com o Conversor Digital Analógico (DAC7568) e com o Multiplexador (MPC507).

Numa primeira abordagem são realizados testes à velocidade de comunicação da placa de teste, via SPI, uma vez que um dos requisitos do projeto é que a comunicação entre o DCS e a placa *HV-Remote* seja feita de segundo a segundo, isto é, que neste intervalo de tempo envie comandos para efetuar leituras ou escritas de informação dos vários PMTs que constituem o detetor. Os testes da placa são realizados com o *Arduino Uno* por ser uma opção de custo bastante acessível, mas também porque permite estabelecer comunicação entre dispositivos a velocidades elevadas.

O primeiro passo, antes de começar a descrição do projeto propriamente dito, será o de descrever as várias componentes técnicas envolvidas, nomeadamente o protocolo de comunicação SPI.

## 5.1. Interface

O barramento SPI é um protocolo de comunicação série síncrono utilizado em comunicações a curta distância, principalmente em sistemas embebidos. Os dispositivos SPI comunicam em modo "full duplex" com base na arquitetura "*Master-Slave*" com um único mestre. O dispositivo *Master* dita os termos segundo os quais a troca de dados é feita entre dispositivos, sendo possível utilizar mais do que um dispositivo *Slave*. Cada Slave tem a sua própria linha de seleção individual (CS ou SS). Este protocolo especifica quatro sinais lógicos, MISO, MOSI, SLCK e SS.

O SCLK impõe a taxa de transmissão de dados entre o *Master* e o *Slave*, o MOSI diz respeito aos dados que são enviados do *Master* para o *Slave*, o MISO corresponde aos dados que são enviados do *Slave* para o *Master*, e o SS é o sinal que permite selecionar um dispositivo como *Slave*.

## 5.2. Modo operacional

Para iniciar a comunicação, o relógio do *Master* tem de ser configurado, usando uma frequência suportada pelo dispositivo *Slave*, que normalmente é de alguns MHz. Após a frequência (de relógio) estar definida, o *Master* seleciona o dispositivo *Slave* com um sinal lógico 0 na linha de seleção (SS). Durante cada ciclo de relógio, ocorre uma transmissão de dados em modo full-duplex. O dispositivo *Master* envia um bit na linha MOSI para o dispositivo *Slave*, e simultaneamente, o dispositivo *Slave*, envia um bit na linha MISO para o dispositivo *Master*.

As transmissões de dados envolvem deslocamento de registos, um no dispositivo *Master* e outro no dispositivo *Slave*, que estão associados à quantidade de informação que se pretende transmitir. Os dados são enviados simultaneamente, do *Master* para o *Slave*, de um registo para outro, sendo que quer no *Master*, quer no *Slave*, o primeiro bit a ser enviado é o MSB (Most Significant Bit), passando a ser o novo LSB (Least Significant Bit) no registo do outro dispositivo. Após a troca de bits estar concluída, o *Master* e o *Slave* trocam valores de registo. Se houver necessidade de transmissão de mais dados, os registos são recarregados e o processo é repetido. A transmissão é feita durante um determinado número de ciclos de relógio. Quando o processo está completo, o *Master* desativa o sinal do relógio, podendo ou não desmarcar o *Slave*.

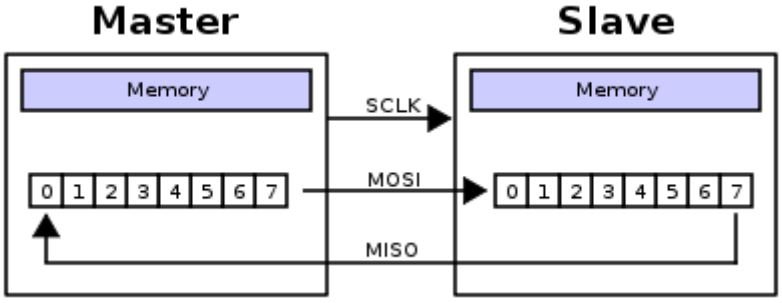


**Figura 5.3:** Ilustração da troca de bits entre *Master* e *Slave*

Além de definir a frequência do relógio, o *Master* também deve configurar a polaridade do relógio CPOL e fase do relógio CPHA.

O diagrama de temporização apresenta-se em seguida, assim como a sua descrição que é válida para *Master* e *Slave*.

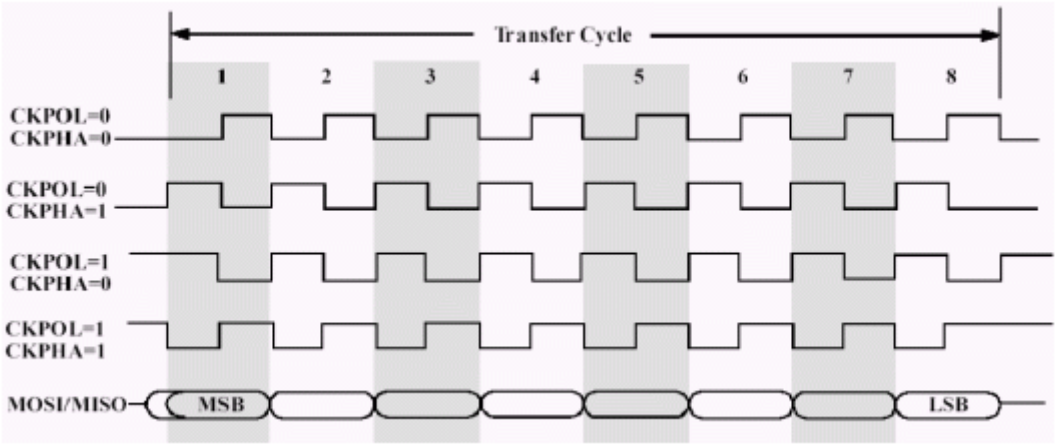


**Figura 5.4:**Diagrama de temporização

Quando CPOL=0 o valor de base do relógio é 0, ou seja, o estado ativo é 1 e estado inativo (idle) é 0.

- Para CPHA=0, os dados são recebidos em transições ascendentes do sinal de relógio e são enviados em transições de descida.
- Para CPHA=1, os dados são capturados em transições descentes do sinal de relógio e são enviados em transições ascendentes.

Quando CPOL=1 o valor de base do relógio é um, ou seja, o estado ativo é 0 e estado inativo (idle) é 1.

- Para CPHA=0, os dados são recebidos em transições descendentes do relógio e são enviados em transições ascendentes.
- Para CPHA=1, os dados são recebidos em transições ascendentes do sinal de relógio e são enviados em transições descendentes.

Assim pode concluir-se que quando CPHA=0, a amostragem é feita na primeira transição de relógio, enquanto CPHA=1 significa que a amostragem é feita na segunda transição do relógio, independentemente de estas transições serem ascendentes ou descendentes. Por outras palavras, CPHA=0 significa que a transmissão de dados é feita do estado ativo para inativo, e CPHA=1 significa que os dados são transmitidos do estado inativo para o estado ativo. Note-se que, se a transmissão ocorre num dado flanco, a receção desses dados irá acontecer no flanco oposto (isto é, se a transmissão ocorrer num flanco descendente, a receção acontece num flanco ascendente e vice-versa).

## 5.3. Números dos modos

As combinações de polaridade e de fase são muitas vezes referidas como modos que são vulgarmente numerados de acordo com a convenção seguinte, com CPOL como o bit de ordem alta e CPHA como o bit de ordem inferior.

| Modo | CPOL | CPHA |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |

**Tabela 5.1:** Tabela resumo da numeração de modos

## 5.4. Configurações

Existem dois tipos de configurações possíveis para a comunicação utilizando o protocolo SPI, configuração simples e configuração em cadeia. Em seguida serão abordadas ambas de forma breve, adiantando já que a que tem maior importância para este trabalho é a configuração simples.

### 5.4.1. Configuração simples

Na configuração de *Slaves* simples ou independentes existe uma linha de escolha de chip (SS: *Slave Select*) independente para cada dispositivo. O mesmo é válido para os pinos MISO (*Master In, Slave Out*) e MOSI (*Master Out, Slave In*). A designação *Slave Select*, significa que é um sinal enviado para um determinado dispositivo que o configura como Slave, MISO, significa que os dados são enviados do *Slave* para o *Master*, na designação MOSI, os dados fazem o trajeto inverso ao do MISO. Esta é a forma mais comum de utilizar a comunicação SPI.

### 5.4.2. Configuração em cadeia

Na configuração em cadeia o sinal lógico MOSI do dispositivo *Master* está ligado à entrada (MOSI) do dispositivo *Slave* 1, tal como acontece na configuração anterior. A diferença reside no facto de que a saída (MISO) do dispositivo *Slave* 1 está ligada à entrada (MOSI) do dispositivo *Slave* 2 e assim por diante até toda a rede de dispositivos estar conectada. A porta SPI de cada escravo é projetada para enviar durante o segundo grupo de pulsos de relógio uma cópia exata dos dados recebidos durante o primeiro grupo de impulsos de relógio, o que permite concluir que toda a cadeia atua como um registo de deslocamento de comunicação. Este tipo de encadeamento é muitas vezes feito com registos de deslocamento para fornecer um banco de entradas ou saídas através de SPI, o que requer apenas uma única linha SS do dispositivo *Master*, em vez de uma linha SS (Slave Select) separada para cada *Slave* da cadeia.

A imagem da Figura 5.5 diz corresponde à configuração simples e a Figura 5.6 diz respeito à configuração em cadeia.

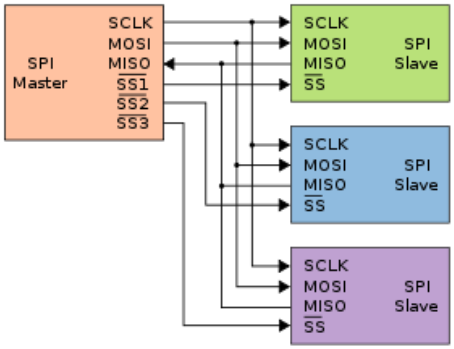

**Figura 5.5:** Configuração SPI simples

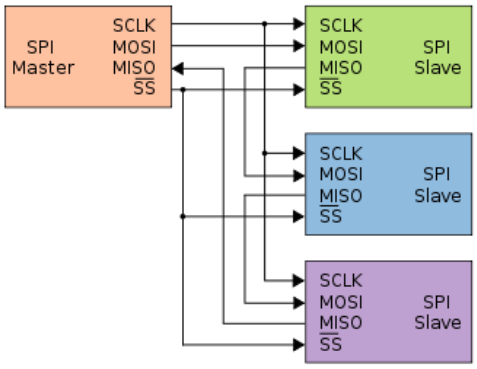


**Figura 5.6:** Configuração SPI em cadeia

## 5.5. *Ethernet* e TCP/IP

A comunicação via *Ethernet* é uma forma de comunicação que descreve como vários dispositivos de uma rede podem formatar dados para transmissão. Cada ponto da rede tem uma chave de 48 bits única, conhecida como endereço MAC (Media Access Control), para assegurar que todos os sistemas tenham endereços distintos. Este tipo de comunicação utiliza vários protocolos, de entre os quais se destaca o TCP/IP, que é um conjunto de protocolos utilizados na comunicação entre dispositivos em rede.

## 5.6. O módulo EM1206

Nesta secção será feita uma breve introdução ao módulo TIBBO e referido o seu papel no projeto final. Além disto serão referidos os testes a que foi sujeito no início, testes estes que consistiam em perceber a velocidade e a qualidade de comunicação deste módulo, e para este efeito utilizaram-se *Arduinos*, como parceiros.

### 5.6.1. Introdução

O módulo EM1206+RJ203 é um módulo comercial desenvolvido pela TIBBO. Este módulo de pequenas dimensões (34,5 x 19 mm) é programável em BASIC e em C e permite a conversão do protocolo 100BaseT[5] *Ethernet* para o protocolo SPI (sinais de entrada/saída da placa de controlo da *HV-Remote* e respetivo endereço). Na figura seguinte é possível ver um esquema, onde estão presentes os vários elementos do projeto e como irão comunicar entre si, quando o projeto final estiver concluído.

[5] Também conhecido como *Fast Ethernet*, é um termo coletivo para uma série de padrões *Ethernet* que transportam informação a uma velocidade de 100 Mbit/s.


**Figura 5.7:** Comunicação entre um computador e a placa de controlo.

## 5.6.2. Testes ao módulo

Após o levantamento da informação necessária, passou-se então aos testes com a placa TIBBO. Numa primeira abordagem foi utilizado o *Arduino Uno*, e para se estabelecer ligação entre ambos é necessário identificar os pinos correspondentes, em cada dipositivo. Na Figura 5.8 apresenta-se uma imagem dos pinos associados à placa de testes TIBBO, e na Tabela 5.2 encontra-se a correspondência de cada pino com a linha SPI. A numeração é do bloco realçado na figura abaixo é feita de baixo para cima e da esquerda para a direita.

Os pinos associados ao módulo EM1206 são descritos na seguinte tabela.

| Pino | Linhas placa EM 1206-EV |
|---|---|
| 1 | GND |
| 2 | Vcc (3,3 V) |
| 3 | GPIO15 |
| 4 | GPIO10 |
| 5 | GPIO13 |
| 6 | GPIO9 |
| 7 | GPIO11 |
| 8 | GPIO8 |
| 9 | GPIO12 |
| 10 | GPIO14 |

**Tabela 5.2:** Descrição dos pinos GPIO

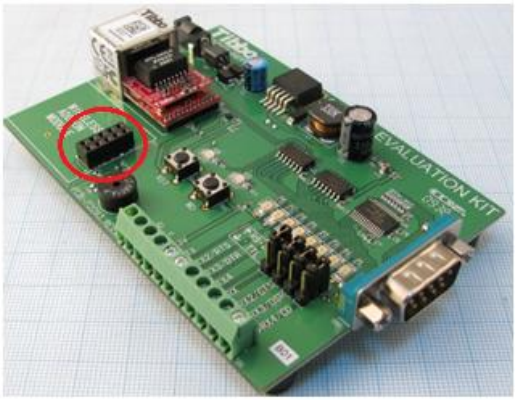

**Figura 5.8:** Pinos utilizados para o teste da placa EM1206-EV

Por seu lado, o *Arduino Uno* apresenta pinos (digitais) especificados, por defeito (Figura 5.9), para estabelecer comunicação SPI, que se encontram na tabela seguinte (Tabela 5.3).

| Número do pino | Sinal SPI |
|---|---|
| 10 | CLK |
| 11 | MOSI |
| 12 | MISO |
| 13 | SS |

**Tabela 5.3:** Pinos associados ao *Arduino* para o SPI


**Figura 5.9:** Identificação dos pinos SPI no *Arduino Uno*

Com base na informação apresentada nas Tabela 5.2 e Tabela 5.3 estabeleceram-se as ligações SPI físicas entre as duas interfaces, e neste caso prático utilizaram-se dois *Arduinos Uno*, em configuração simples, como *Slaves*, uma vez que, tal como já foi dito, pretende-se estudar a resposta do módulo em funcionamento como *Master* com o intuito de saber se é uma hipótese viável para controlar os componentes da placa *HV-Remote* que está a ser desenvolvida pelo grupo. A montagem experimental pode ser vista na figura seguinte. A utilização dos LEDs será explicada quando for feita a exposição do código desenvolvido quer para a placa TIBBO, quer para os *Arduinos*.

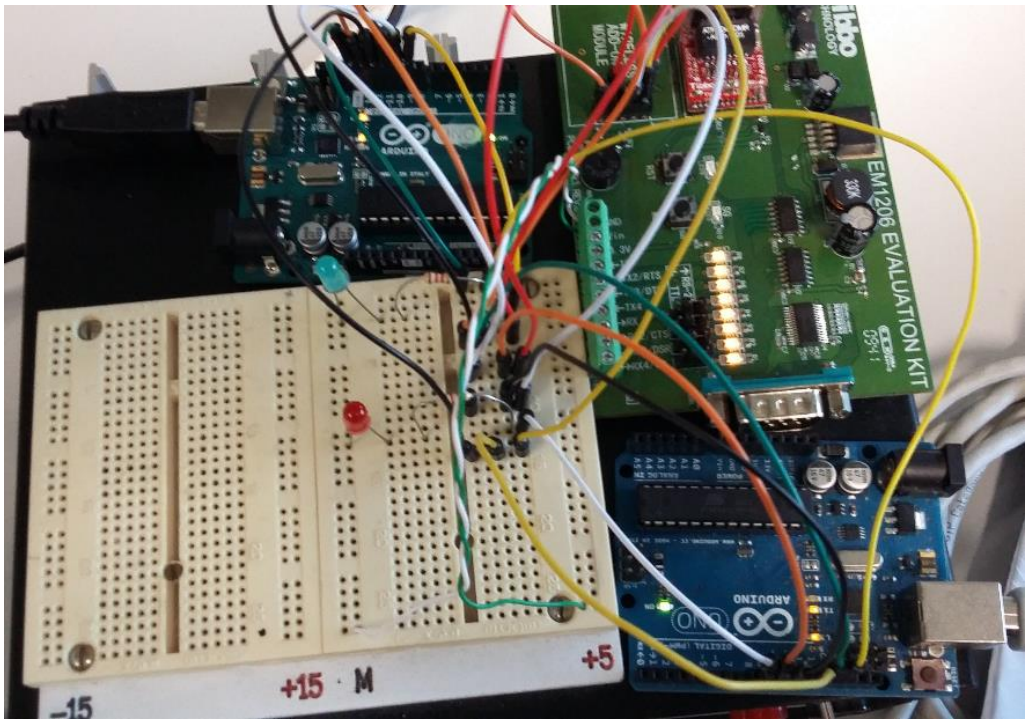


**Figura 5.10:** Montagem experimental nos testes da SPI, incluindo dois *Arduinos* e uma TIBBO EM1206.

## 5.6.3. Descrição do código desenvolvido

Ambos os dispositivos são programados com software de desenvolvimento executado em Windows. Os *Arduinos*, são ligados ao computador através da porta USB e programados utilizando o software proprietário do *Arduino*, e o módulo TIBBO é ligado ao computador via *Ethernet* e é programado com o software oficial da TIBBO, o TIDE.

### 5.6.3.1. Código da placa TIBBO EM1206-EV (para teste da interface SPI)

O código começa com a definição dos pinos SPI da placa de testes. Nesta parte é descrito como se pretende que a comunicação seja estabelecida. Após ser atribuída a cada linha GPIO o respetivo pino, é configurado o seu estado lógico (LOW ou HIGH), que é irrelevante nesta fase uma vez que pode ser alterado em qualquer momento durante a execução do código. Finalmente termina-se com a designação destes pinos de input (MISO)/output (MOSI, SCLK, SS).

```
//GPIO-15, Linha 3 do módulo de internet sem fios
    io.num=SPI_SS;
    io.state = LOW;
    io.enabled=YES;

//GPIO-14, Linha 10 do módulo de internet sem fios
    io.num=SPI_SCLK;
    io.state = HIGH;
    io.enabled=YES;

//GPIO-13, Linha 5 do módulo de internet sem fios
    io.num=SPI_MOSI;
    io.state = LOW;
    io.enabled=YES;

//GPIO-12, Linha 9 do módulo de internet sem fios
    io.num=SPI_MISO;
    io.state = LOW;
    io.enabled=NO;
```

**Figura 5.11:** Configuração dos pinos SPI da placa de testes

Em seguida é configurado o canal SSI da placa de testes: este governa toda a comunicação SPI. São declarados os sinais lógicos indicando-se o bit que é enviado em primeiro lugar (neste caso o MSB) e a taxa de transferência de relógio (*baudrate*), definida por um byte (1-255), sendo que 1 corresponde à velocidade de transferência mais rápida, e 255 corresponde à velocidade mais lenta.

```
//Configuração do canal SSI
    ssi.channel=0;
    ssi.enabled=NO;
    ssi.mode=PL_SSI_MODE_0;
    ssi.clkmap=SPI_SCLK;
    ssi.dimap=SPI_MISO;
    ssi.domap=SPI_MOSI;
    ssi.zmode=PL_SSI_ZMODE_ALWAYS_ENABLED;
    ssi.direction=PL_SSI_DIRECTION_LEFT;
    ssi.baudrate=1;
    ssi.enabled=YES;
```

**Figura 5.12:** Configuração do canal SSI do módulo

Com base na equação seguinte, escolheu-se um "baud rate" igual a 40, que corresponde a um período de relógio T=5,288µs ou, em termos de frequência de relógio, cerca de 200kHz. Este valor foi escolhido para se adaptar à experiência.

$$T = 0{,}8\ (\mu s) + \text{ssi.baudrate} \times 0.112(\mu s) \qquad (1)$$

A função que se segue tem a finalidade de enviar os dados que se pretendem para os dois dispositivos *Slaves* (*Arduinos*).

```
void Send_Data_Values(unsigned char data,int size)
{
    io.lineset(SPI_SCLK,LOW);
    ssi.value(data,size);
    io.lineset(SPI_SCLK,HIGH);
}
```

**Figura 5.13:** Função de envio de dados

Na figura seguinte está uma imagem retirada de um osciloscópio e é possível confirmar que a taxa de transferência em frequência (retângulo vermelho) apresenta um valor em torno dos 200 kHz.

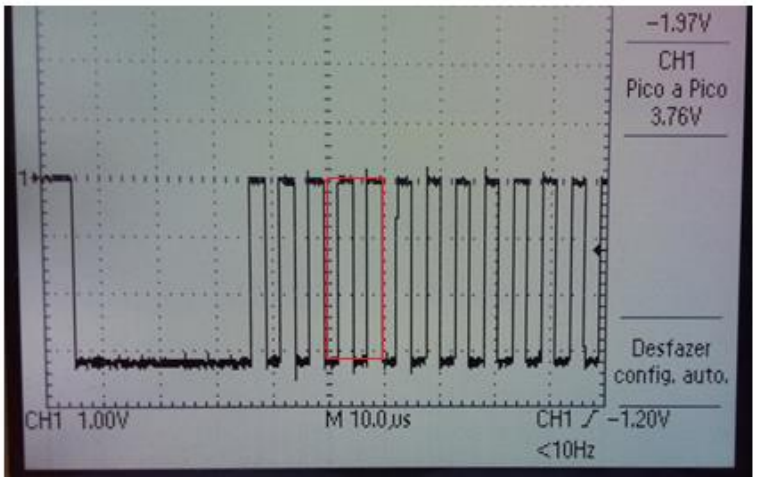


**Figura 5.14:** Frequência de relógio da placa TIBBO

### 5.6.3.2. Código para o *Arduino Uno*

Nesta situação o código começa por verificar se o dispositivo é *Slave* (mais uma vez com o intuito de servir de *debug*) e caso seja, a primeira ação é colocar o pino MISO como output para enviar informação ao *Master*. Em seguida é declarada a variável "rx", responsável por receber os dados que vêm do módulo TIBBO. Existe uma variável de comparação declarada igual a 1. Se a comunicação for estabelecida com sucesso, a TIBBO enviará um valor, em decimal que será convertido em byte, por SPI para cada um dos *Slaves* e estes farão piscar os LEDs que se encontram na *breadboard* da Figura 5.10. No caso de o valor enviado não corresponder ao da variável de comparação, o *Slave* responderá com comando desconhecido e desmarca o dispositivo como *Slave*, ficando à espera de nova informação.

```
 //Verifica o último estado
if (SSlast != LOW)
{
  pinMode(MISO, OUTPUT); //Pino MISO passa a ser output
  Serial.println("***Slave Enabled.");
  // Variável com informação do Master
  byte rx = SPItransfer(255);

  //Caso a informação passada seja igual à variável de controlo
  if (rx == cmdBtn)
  {
    byte rx = SPItransfer(cmdBtn);
    // Altera o estado digital do LED
    ledState = !ledState;
    digitalWrite(led, ledState);
  }
  //Caso contrário
  else
  {
    //Envia uma mensagem de erro
    byte rx = SPItransfer(255);
    Serial.println("Unrecognized Command.");
  }
  //Coloca o último estado como LOW
  SSlast = LOW;
 }
}
```

**Figura 5.15:** Verificação de *Slave* por parte do *Arduino*

A figura anterior (Figura 5.15) mostra a função desenvolvida para testar a comunicação SPI do *Arduino* com a TIBBO. Por fim, o estado da variável de controlo do *Slave* é alterado e caso seja diferente de HIGH os *Arduinos* deixam de ser *Slaves* e o programa termina.

```
else
{
  //Se o último estado for LOW passa a HIGH
  if (SSlast != HIGH)
  {
    pinMode(MISO, INPUT);
    Serial.println("Slave Disabled.");
    SSlast = HIGH;// Atualiza o último estado
  }
}
```

**Figura 5.16:** Desmarcação do *Arduino* como *Slave*

O desenvolvimento do sistema de controlo começou com a familiarização com o protocolo SPI e com a placa de testes TIBBO. Além da placa de testes, utilizaram-se dois *Arduinos Uno* (Anexo A). Foram feitos testes com um único *Arduino Uno*, e com dois deles em configuração normal, isto é, sem interação entre si. Neste caso, pretendia-se apenas testar a velocidade de comunicação da placa de testes cuja frequência máxima do relógio é de ~1,1 MHz (situação em que em (1) ssi.baudrate=1 byte) correspondente a um período de ~0,912 µs. Nos testes, não foi utilizado este valor máximo, foi utilizado um ssi.baudrate= 40 que corresponde a cerca de ~200 kHz devido às restrições impostas pelo microprocessador do *Arduino Uno*. Na versão final do projeto, esta limitação terá de ser tida em conta, uma vez que o expansor utilizado suporta uma frequência de relógio máxima de 10 MHz. Para verificação e validação do funcionamento utilizou-se um LED (Light Emission Diode); sempre que era transferido um byte com o valor correto (o programa escrito para o *Arduino* utiliza uma variável de comparação), este acendia. O mesmo foi feito na situação em que foram utilizados dois *Arduinos*.

# 6. Placa *HV-Remote* e Sistema de Controlo

Neste capítulo, começa-se por fazer uma referência sobre a estadia do autor no CERN, seguindo-se uma breve apresentação da nova versão da carta *HV-Opto*, denominada *HV-Remote*, que o grupo LIP/FCUL/Inesc-ID está a desenvolver com vista a ser usada no *update* do *TileCal*. Será explicado brevemente o seu funcionamento e a arquitetura em termos de componentes, e esses componentes serão descritos sumariamente. No fim do capítulo será então apresentado o sistema de controlo da *HV-Remote* desenvolvido neste projeto.

## 6.1. Estadia ao CERN

Com a participação neste projeto surgiu a oportunidade de ir ao CERN participar no "test beam" que decorreu duas vezes no ano de 2016, a primeira em junho e a segunda entre 18 de setembro e 5 de outubro. Foi no segundo "test beam" que estive presente na experiência.

O "test beam" decorreu na zona norte do CERN, em Prévessin, França, no edifício 887 do complexo, classificado como área supervisionada, incluindo as salas de controlo. A supervisão requer por parte dos colaboradores a apresentação de dosímetro (Figura 6.1) quando se dirigem para o serviço.


**Figura 6.1:** Dosímetro

O dosímetro é pessoal, intransmissível, e para se obter um dispositivo como o da figura anterior no CERN é necessário completar o curso de proteção radiológica para zonas de radiação com supervisão, e além do dosímetro para ter acesso à zona onde decorre a experiência é necessário usar um capacete.

Além do "test beam" da experiência ATLAS decorriam, em simultâneo no mesmo edifício, outras experiências, tais como a CMS, a COMPASS, a LHCb, entre outras, pelo que espaço onde decorrem estas experiência está devidamente separado, por experiência, e devidamente blindado devido às elevadas doses de radiação existentes quando o feixe de partículas está em funcionamento. A figura seguinte revela o que foi dito em relação à divisão e isolamento/blindagem do local dos testes, embora

mostre só uma pequena parte visto que a dimensão real desse local, que na verdade é um hangar, não é discernível.

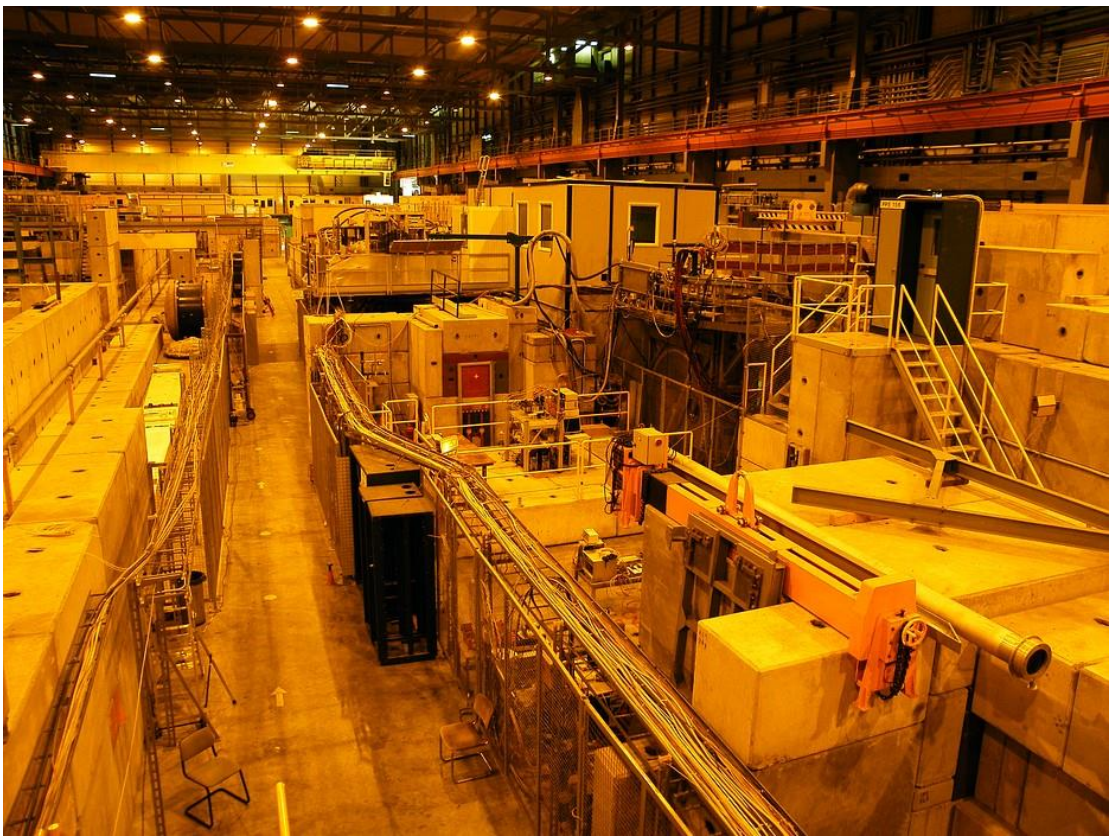

**Figura 6.2:** Disposição do edifício 887

A zona onde decorreu o teste em que participei é designada área H8A e tem aproximadamente 6 m de largura e o feixe de partículas encontra-se a cerca de 1,27 m do chão.

**Figura 6.3:** H8A na direção do feixe

A zona onde decorreu o teste, que pode ser vista em detalhe na figura seguinte, era apenas acessível quando o feixe de partículas estivesse em paragem técnica, devido a uma eventual avaria ou caso fosse necessário encher as botijas de oxigénio, que se encontram no início da cadeia de aceleração. O teste consistia em fazer incidir partículas, de três tipos: eletrões, piões ou muões, nos módulos semelhantes aos do calorímetro *TileCal*, como os que se encontram na Figura 6.4.

**Figura 6.4:** Mesa móvel (em baixo) e módulos do *TileCal* (em cima)

O teste em si tem como objetivo assegurar o melhoramento constante do desempenho do calorímetro *TileCal* por forma a que venham a ser produzidas medições cada vez mais precisas que permitam, juntamente com aquelas obtidas por todas as outras experiências do LHC, explicar eventualmente a estrutura e a origem do nosso universo. Durante o tempo que decorreu o teste, cerca de 18 dias, fiz parte de uma equipa de técnicos de operação. O trabalho, que correspondeu a 5 turnos de 8 horas para um par de pessoas, consistia em registar dados resultantes da incidência das partículas nos módulos do *TileCal*. Basicamente esses dados dividem-se em três tipos: eficiência luminosa do calorímetro devido à incidência das partículas; estudos de irradiação; e varrimento angular. Os módulos podiam ser movidos uma vez que estavam assentes sobre uma mesa móvel, colocada sobre carris como se pode observar na figura, que permitia deslocações angulares e transversais. Todos os resultados obtidos foram guardados em servidores do CERN para serem sujeitos a análise futura pelos muitos cientistas que colaboram com a experiência, espalhados um pouco por todo o mundo.

Além do trabalho como técnico de operação, houve a oportunidade de adquirir conhecimentos sobre o DCS, estabelecer contacto com experientes técnicos envolvidos na sua operação há vários anos, e isto contribuiu para concretizar o trabalho efetuado nesta tese.

A visita ao CERN foi uma oportunidade singular que contribuiu para o desenvolvimento pessoal do autor, foi uma experiência fantástica onde pôde observar como se trabalha num dos mais desenvolvidos laboratórios do mundo focados na área da física de altas energias e aprofundar conhecimentos lidando com engenheiros e cientistas que operam na vanguarda daquela área científica.

## 6.2. Placa de Controlo *HV-Remote*

O novo formato da placa *HV-Opto*, denominado *HV-Remote*, mantém na generalidade os circuitos elétricos da *HV-Opto* com exceção de alguns componentes que já se encontram obsoletos e, por isso, têm de ser substituídos. A diferença entre aquelas duas cartas reside no facto da *HV-Remote* não necessitar de uma placa de controlo dedicada (a *HV-Opto* é controlada por uma "placa irmã" denominada HV Micro), uma vez que o controlo agora será feito com software dedicado executado num computador, e a *HV-Remote* comunica com esse computador através dos protocolos *Ethernet* e SPI

mediados pelo módulo TIBBO (ou outro equivalente que eventualmente venha a ser selecionado no futuro). A versão final da placa, designada por *HV-Remote*, será composta por uma parte digital e outra analógica.

## 6.2.1. Funcionamento da Placa de Controlo

Para testar o hardware que é novo na placa *HV-Remote*, relativamente ao que se encontra na *HV-Opto*, foi projetada uma carta, denominada aqui "Placa de Controlo" *HV-Remote-Ctrl*, que essencialmente serve para testar as interfaces *Ethernet* e SPI. Esta placa de testes consiste numa versão mais simples da *HV-Remote* uma vez que não contém a parte analógica.

O diagrama de blocos [33], [34] da *HV-Remote-Ctrl* encontra-se apresentado na Figura 6.5.

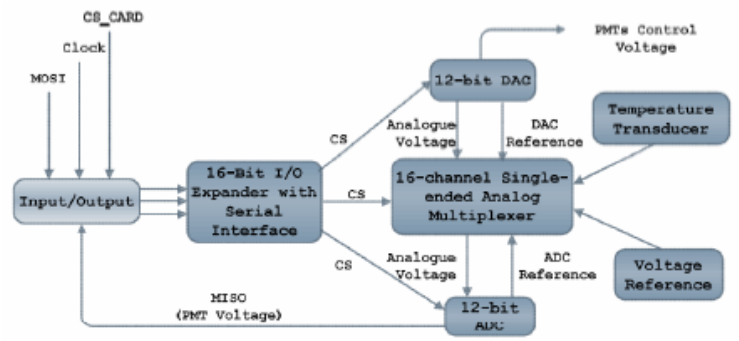


**Figura 6.5:** Esquema funcional do protótipo *HV-Remote-Ctrl* contendo a interface da *HV-Remote*.

A principal diferença entre esta nova versão da placa *HV-Opto*, denominada *HV-Remote*, e a anterior reside no caminho entre o sistema DCS e as 256 placas *HV-Remote* ligadas ao sistema que com esta nova alteração consistirá de uma arvore de ligações *Ethernet*. Esta arvore será implementada com um único computador ligado a uma rede local, que funcionará como servidor, e que também comunica (por outra porta *Ethernet*) com o DCS. Os novos componentes eletrónicos que constituem a interface da placa *HV-Remote* são apresentados de seguida, acompanhados por uma breve descrição do seu funcionamento.

### 6.2.1.1. MCP23017

Este dispositivo [18] dispõe de interface SPI e funciona essencialmente como expansor de portos digitais, dispondo de 16 bits de GPIO (*General Purpose I/O*). A comunicação SPI funciona com uma frequência máxima de relógio de 10 MHz. O componente permite configurações individuais para os 16 bits/pinos que podem ser usados com input ou output, com seleção de inversão de polaridade. No barramento SPI este dispositivo funciona como *slave*, e o *master* configura os GPIOs como input ou output escrevendo nos registos de configuração (IODIRA/B). Os pinos GPIO consistem em dois bancos/portos de 8 bits cada (PORTA e PORTB) e podem ser configurados para funcionar como 2 x 8 bits ou como 16 bits através do registo IOCON.BANK. O componente permite mais configurações complexas e dispõe de portas de interrupção (*interrupt*). Os detalhes encontram-se na sua folha de características.

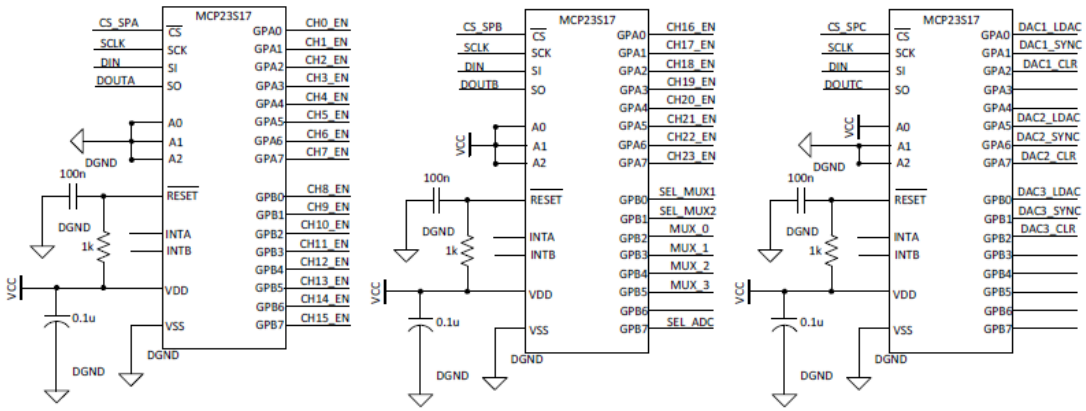


**Figura 6.6:** Esquemas elétricos dos MCP23017 da placa

Este chip converte dados série provenientes dos (poucos) pinos do módulo TIBBO para um formato paralelo de modo a que possa ser feita a comunicação com e o controlo dos restantes chips que constituem a placa. A carta *HV-Remote-Ctrl* dispõe de 3 chips MCP23017.

### 6.2.1.2. MPC507A

O MPC507A [19] é um *Multiplexer* diferencial de 8 canais, que permite implementar várias configurações que podem expandir a capacidade (número de canais) até 64, seja de forma singular (nó simples 64×1) ou numa configuração múltipla (8×8). Em seguida apresenta-se uma ilustração do chip conforme aparece na carta *HV-Remote-Ctrl*.

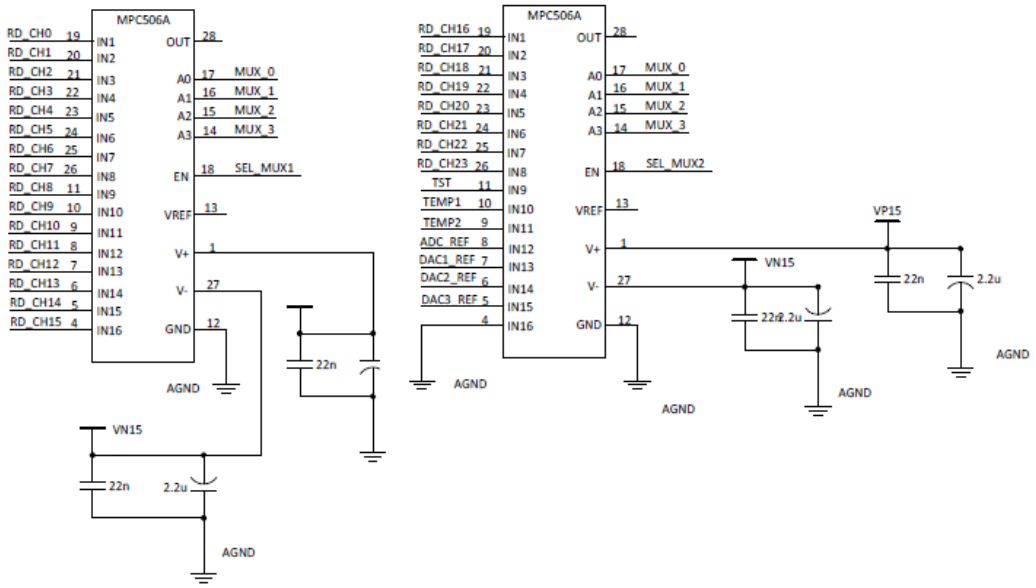


**Figura 6.7:** Esquemas elétricos dos MPC507A da placa

### 6.2.1.3. MAX3002

Este chip [16] pode ser visto como um conversor de nível (de tensão) de 8 canais que permite transferência de dados em sistemas com tensões diferentes. As tensões externas aplicadas ($V_{CC}$ e $V_L$) definem os níveis de tensão lógicos em ambos os lados do chip. Este dispositivo apresenta uma

arquitetura especificamente desenhada para ser automaticamente bidirecional sem ser necessário recorrer a um registo de definição dos sentidos I/O.

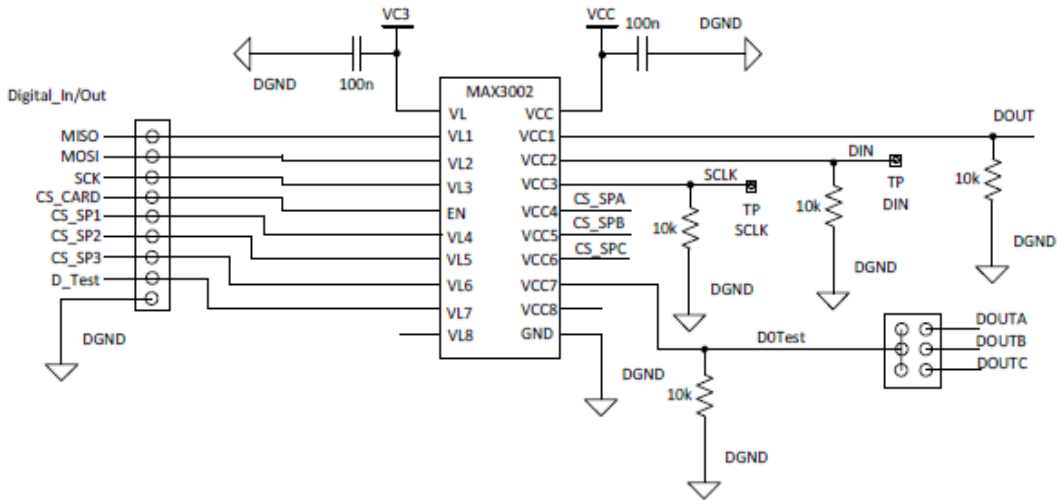


**Figura 6.8:** Esquema elétrico do MAX3002 da placa

### 6.2.1.4. DAC7568

O DAC7568 [17] é um conversor digital-analógico de 8 canais, de 12 bits de resolução e de baixo consumo. Na imagem seguinte apresenta-se uma ilustração dos esquemas elétricos dos DAC7568 utilizados na *HV-Remote-Ctrl*. A arquitetura do chip DAC7568 consiste em 8 conversores digital-analógico com cadeias resistivas (*resistor strings*) com interruptores, cada uma acoplada a um buffer (seguidor) de saída.

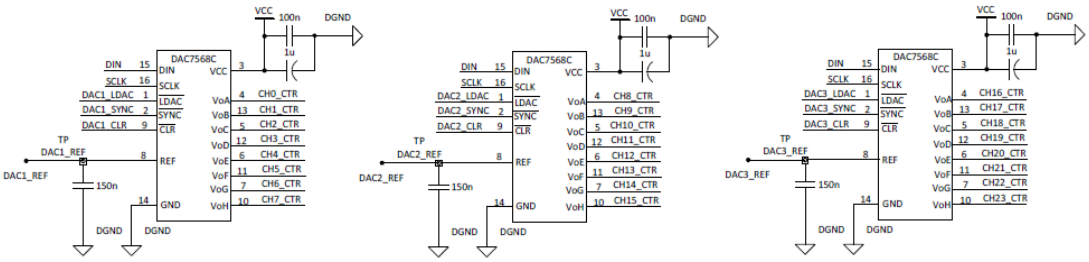


**Figura 6.9:** Esquemas elétricos dos DAC7568

Estes DACs têm como função fornecer o sinal analógico do valor de tensão que permite a regulação individual da alta tensão dos PMTs.

### 6.2.1.5. TLV2541

É um chip [22] conversor analógico digital de elevado desempenho, de 12 bits de resolução, de baixo consumo, que funciona através de uma interface SPI com um relógio limitado superiormente a 20 MHz. A frequência máxima de funcionamento do chip depende do modo de operação e tem um relógio de conversão interno de 3,5 µs. O chip TLV2541 é um ADC de aproximações sucessivas que utiliza um DAC de redistribuição de carga.

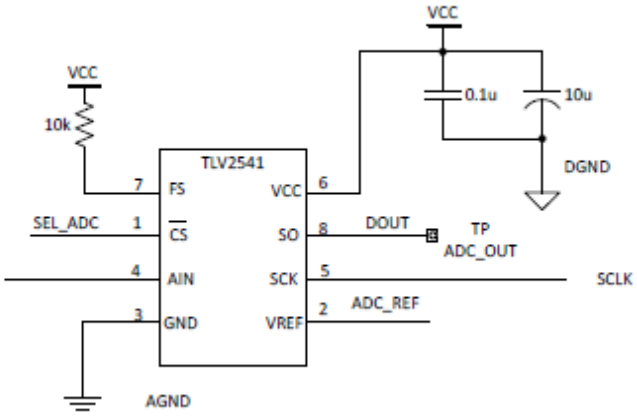


**Figura 6.10:** Esquema elétrico do TLV2541

Para automatizar os testes do DAC, os seus sinais de saída são convertidos pelo ADC e enviados para o computador. O ADC (Figura 6.10) também permite a leitura das tensões de referência dos conversores utilizados (ADC e DAC Figura 6.9), da tensão de saída dos dois sensores de temperatura (Figura 6.11) e de uma tensão de referência de teste (Figura 6.12).

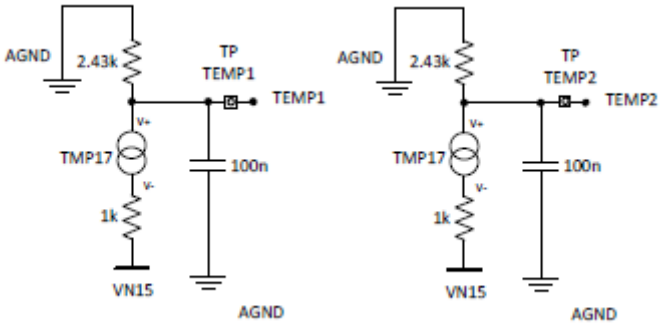


**Figura 6.11:** Sensores de temperatura da placa *HV-Remote-Ctrl*

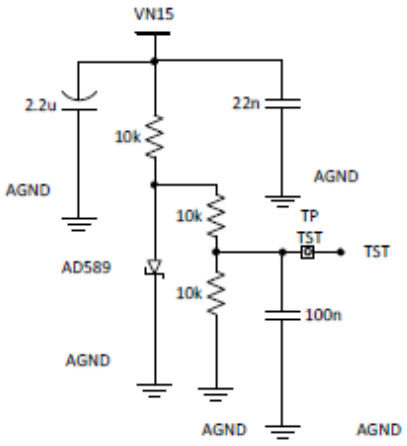


**Figura 6.12:** Tensão de referência de teste

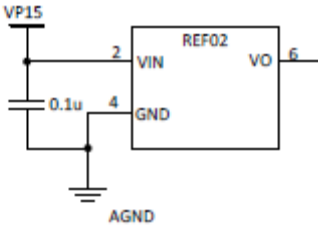


**Figura 6.13:** Tensão de referência ADC

Além dos chips mencionados, a seguir aos multiplexadores está colocado um circuito inversor, uma vez que o sinal que provém desses multiplexadores é negativo e tem de ser invertido antes de chegar ao ADC.

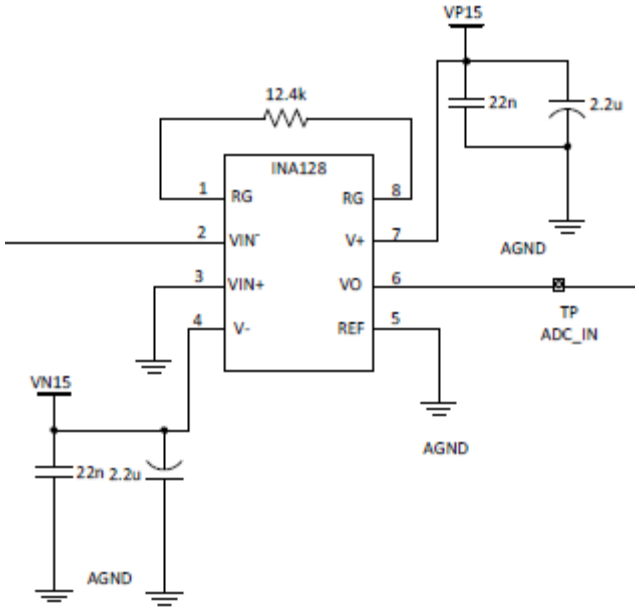


**Figura 6.14:** Esquema elétrico do Inversor INA128

## 6.3. Sistema de Controlo

O código implementado no módulo TIBBO EM1206 está desenvolvido em linguagem C, e tem por objetivo controlar as placas *HV-Remote* (ou *HV-Remote-Ctrl*), acedidas por expansores série-paralelo que permitem aumentar o número de portas para comunicação com os outros chips, tais como DACs, ADCs e *Multiplexers* que fornecem a alta tensão aos PMTs do detetor TileCal. Este código funciona em conjunto com o código executado no DCS desenvolvido para o *WinCC*, em linguagem C++, em que a comunicação entre os dois (DCS e TIBBO) é estabelecida através de *Sockets* sobre um canal *Ethernet*, mais concretamente, através do DCS o utilizador consegue enviar comandos de escrita e leitura de informação para a placa *HV-Remote*. Em seguida apresenta-se a descrição do código desenvolvido para as duas plataformas, começando com o do módulo TIBBO.

### 6.3.1. Código do módulo TIBBO

Para haver ligação entre o módulo TIBBO e o DCS não basta existir uma ligação física (via *Ethernet*, neste caso): é necessário software que sincronize as duas extremidades da ligação. Nesta situação é uma

ligação por *Sockets*, que não são mais do que um mecanismo abstrato de comunicação usado normalmente para implementar um modelo cliente/servidor.

```
net.ip="10.101.162.201";
sock.num=1;
sock.protocol=PL_SOCK_PROTOCOL_TCP;
sock.inconmode=PL_SOCK_INCONMODE_ANY_IP_ANY_PORT;
sock.reconmode=PL_SOCK_RECONMODE_3;
sock.localportlist="1000";
sock.rxbuffrq(1);
sock.txbuffrq(1);
sys.buffalloc();
```

**Figura 6.15:** Configuração dos *Sockets* na TIBBO.

Na Figura 6.15 está presente a função que configura o canal de *Sockets* do módulo TIBBO. Como qualquer ligação cliente/servidor é necessário referir o IP (net.ip) e a porta (sock.localportlist) de ligação. Após declarado o IP ao qual se pretende estabelecer ligação é enumerado o canal de Socket (sock.num), neste caso é o canal 1, segue-se o tipo de ligação (sock.protocol) que será TCP/IP, e a forma de como será estabelecida a ligação (sock.inconmode e sock.reconmode).

No final é alocada 1 página de memória no módulo com cerca de 256 bytes, suficiente, para controlar a quantidade de informação que tem de ser enviada e que chega aos buffers do módulo.

O próximo passo é configurar o canal SSI (Figura 6.16) do módulo, que é responsável por toda a ligação SPI com a placa *HV-Remote-Ctrl*.

```
//MOSI pin
io.num=SPI_MOSI;
io.state=LOW;
io.enabled=YES;

//MISO pin
io.num=SPI_MISO;
io.state=LOW;
io.enabled=NO;

//CS_CARD pin
io.num=SPI_CS_CARD;
io.state=HIGH;
io.enabled=YES;

//CS_SP1 pin
io.num=SPI_CS_SP1;
io.state=HIGH;
io.enabled=YES;

//CS_SP2 pin
io.num=SPI_CS_SP2;
io.state=HIGH;
io.enabled=YES;

//CS_SP3 pin
io.num=SPI_CS_SP3;
io.state=HIGH;
io.enabled=YES;

//D_TEST pin
io.num=SPI_D_Test;
io.state=LOW;
io.enabled=NO;

//CLOCK pin
io.num=SPI_CLK;
io.state=HIGH;
io.enabled=YES;

//ssi channel configuration
ssi.channel=0;
ssi.mode=PL_SSI_MODE_3;
ssi.clkmap=SPI_CLK;
ssi.dimap=SPI_MISO;
ssi.domap=SPI_MOSI;
ssi.zmode=PL_SSI_ZMODE_ALWAYS_ENABLED;
ssi.direction=PL_SSI_DIRECTION_LEFT;
ssi.baudrate=40;
ssi.enabled=YES;
```

**Figura 6.16:** Configuração do canal SSI

Primeiramente são configurados os pinos do módulo que vão ser utilizados para a ligação SPI, toda esta configuração é uma repetição, embora com mais pinos, da que foi feita no código para testar a ligação SPI do módulo com os *Arduinos*. Os vários pinos configurados dizem respeito aos necessários à troca de informação (MISO e MOSI), o pino do relógio (CLK), os vários pinos de seleção de *SLAVE* (CS_CARD, CS_SP1, CS_SP2 e CS_SP3). Nestes 4 pinos, o primeiro diz respeito a seleção de uma *HV-Remote* das 256 que existem em torno do detetor TileCal, e os restantes 3 pinos, após a placa ser

selecionada dizem respeito aos 3 expansores MCP23017 existentes na placa. Tem ainda um pino de teste (D_TEST) que à partida só vai ser ligado quando for necessário realizar testes a um determinado componente de uma determinada placa. Após a configuração dos *Sockets* é necessário discriminar no canal SSI a forma segundo a qual os dados vão ser transmitidos (ssi.zmode). É selecionado o canal 0 por opção e configuram-se o mapa de relógio (ssi.clkmap), os mapas de transmissão de dados (ssi.dimap e ssi.domap), a taxa de transferência de dados (ssi.baudrate), e no final é accionado o funcionamento (ssi enabled).

Foram também desenvolvidas duas funções responsáveis pela seleção e desmarcação de *SLAVES* em cada placa.

```
//Chip select for port expander A
if(w_r==64 || w_r==65)
{
    io.lineset(SPI_CS_SP1,LOW);
}
//Chip select for port expander B
else if(w_r==78 || w_r==79)
{
    io.lineset(SPI_CS_SP2,LOW);
}
//Chip select for port expander C
else if(w_r==76 || w_r==77)
{
    io.lineset(SPI_CS_SP3,LOW);
}
```

**Figura 6.17:** Função de seleção de *SLAVE*

```
//Chip deselect for port expander A
if(w_r==64 || w_r==65)
{
    io.lineset(SPI_CS_SP1,HIGH);
}
//Chip deselect for port expander B
else if(w_r==78 || w_r==79)
{
    io.lineset(SPI_CS_SP2,HIGH);
}
//Chip deselect for port expander C
else if(w_r==76 || w_r==77)
{
    io.lineset(SPI_CS_SP3,HIGH);
}
```

**Figura 6.18:** Função de desmarcação de *SLAVE*

No protocolo SPI um dispositivo é selecionado como *SLAVE* sempre que recebe um sinal lógico 0 (LOW). As duas funções apresentam vários ramos de seleção, um para cada um dos três chips que estão presentes nas placas, que têm em conta o formato da mensagem enviada ao módulo, e será explicado quando for estudado esse formato da mensagem que é enviada ao módulo.

Quanto à mensagem enviada, é uma *string* de vários caracteres, em formato decimal, e pode ter uma de duas das seguintes formas, que dependem do valor de *x* (constante de seleção):

**Nota:** embora o formato das mensagens seja apresentado como se segue, na realidade não existem vírgulas a separar a informação, apenas foram adicionadas aqui para efeitos de explicação do seu funcionamento e constituição.

Se x=0, então a mensagem é da seguinte forma:

0, Ca, Ra, Value

E a função responsável que controla a informação é a seguinte (Figura 6.19).

```
//Writes the chip address
data_0[0]=mid(split,2,2);

//Writes the chip's register
data_0[1]=mid(split,4,2);

//Writes the value on said register
data_0[2]=right(split,3);
```

**Figura 6.19:** Controlo de informação (x=0)

1. O primeiro caracter "0", diz ao programa que pode ou não escrever para Muxs ou ADC, mas escreve sempre para um expansor.
2. Os dois caracteres "Ca" correspondem ao endereço do expansor, configurado pelos pinos ($A_2$ $A_1$ e $A_0$ de cada expansor), e podem ser um dos seguintes "64" ou "76" em caso de escrita e no caso de leitura "65" ou "77".
3. Os dois caracteres "Ra" dizem respeito ao endereço de registo do expansor, e podem ser "18", associado ao GPIOA ou "19", associado ao GPIOB.
4. O campo "Value" corresponde ao valor que se pretende escrever nos registos do expansor; nesta situação, esses valores (1 byte) estão compreendidos entre 0 e 255.

Na situação em que x=1, a mensagem toma a seguinte forma:

1, Ca, Ra, Value, PC, AD, F

E a função responsável pelo controlo dessa informação é a seguinte (Figura 6.20).

```
//Writes the chip address
data_1[0]=mid(split,2,2);

//Writes the chip's register
data_1[1]=mid(split,4,2);

//Writes the value on said register acording to DAC location
data_1[2]=mid(split,6,3);

//Writes and combines the 4 prefixed bits and the 4 control bits
data_1[3]=mid(split,9,2);

//Writes and combines the 4 address bits and the first 4 bits of data
data_1[4]=mid(split,11,3);

//Writes the remaining 8 bits of data
data_1[5]=mid(split,14,3);

//Writes remaining info (don't care)
data_1[6]=mid(split,17,4);
```

**Figura 6.20:** Controlo de informação (x=1)

1. No caso em que o primeiro caracter é "1", escreve para os DACs, passando sempre por um expansor.
2. Os dois caracteres "Ca" correspondem ao endereço do expansor a partir do qual se pode aceder aos DACs, e é o "78" em caso de escrita, e toma o valor "79" em caso de leitura.
3. Os dois caracteres "Ra" dizem respeito ao endereço de registo do expansor, e podem ser "18", associado ao GPIOA ou "19", associado ao GPIOB.

4. O campo “Value” corresponde ao valor que se pretende escrever nos registos do expansor; nesta situação, esses valores estão compreendidos entre 0 e 255.
5. Os caracteres “PC” dizem respeito a bits prefixos e aos bits de controlo do DAC, que por defeito têm o valor 0x00 (valor nulo em hexadecimal).
6. Os caracteres “AD” dizem respeito aos bits de endereço (4 bits) do DAC e à informação (12 bits) que vai ser enviada para ele.
7. Finalmente, o caracter “F” (8 bits), diz respeito aos bits de “feature” que também não foram considerados relevantes, e como tal tem o valor 0x00 (valor nulo em hexadecimal).

Quando for dada a ordem para enviar informação para determinada zona da placa, o módulo TIBBO converte a informação para formato ASCII que vai ser descodificado pela eletrónica da *HV-Remote*. Em anexo é possível rever o código descrito em cima em maior detalhe bem como inspecionar outras funções importantes, necessárias ao funcionamento pretendido para o módulo TIBBO.

### 6.3.2. Código desenvolvido para o DCS

Na imagem seguinte mostra-se o painel de controlo feito com o *WinCC* para o DCS. Este painel, tem como função servir de janela para o controlo de 24 PMTs (meia *drawer*), permite que o utilizador envie comandos de leitura e escrita para o módulo TIBBO, através dos vários botões disponíveis, e só é possível enviar esses comandos quando os canais estão ativos (em inglês “enabled”).

Para começar a descrever este painel é necessário abordar primeiro as funções responsáveis por estabelecer uma ligação TCP e pelo envio de informação, que são as seguintes (Figura 6.21 e Figura 6.22).

```
dyn_errClass err;
int timeout = 10;
socket = tcpOpen("10.101.162.201", "1000", timeout);
DebugTN(" hv_Rem_initTCP >> tpc open at..." +socket);
err=getLastError();
if (dynlen(err)!=0)
{
  tcpClose(lastOpendSocket);
  DebugTN("Error: "+ err);
  return -1;
}
return socket;
```

**Figura 6.21:** Função que estabelece ligação TCP

```
hvRem_initTCP();
dyn_errClass err; //error class
int ret = tcpWrite(socket, command); //execute command
err=getLastError();
if (dynlen(err)!=0) // in case of error prints error and returns -1
{
  DebugTN("Error: "+ err);
  return -1;
}
delay(1);
int read = tcpRead(socket, messageReceived, 1);//
tcpClose(socket);
return 0;
```

**Figura 6.22:** Função que controla o envio de informação

As funções das Figura 6.21 e Figura 6.22 são a base da ligação TCP do sistema projetado. A ligação TCP no DCS especifica as quatro funções necessárias:

1. tcpOpen(ip,port,timeOut): esta função permite estabelecer ligação TCP/IP com outro dispositivo, sendo conhecido o IP a que se pretende ligar, a porta de ligação e o tempo de espera admitido para se estabelecer a ligação;
2. tcpClose(Socket): esta função fecha o Socket com qual se estabeleceu a ligação;
3. tcpRead(Socket,messageReceived,confirm): esta função é responsável por fazer leituras de mensagens em *Sockets* na ligação TCP.
4. tcpWrite(Socket,command): esta função é parecida à função tcpRead() em termos de funcionamento, com a diferença de que esta é utilizada para escrever mensagens obre *Sockets*.

Após estar aberta uma ligação TCP entre o DCS e o módulo TIBBO, este painel apresenta várias formas de enviar comandos de escrita e de leitura. Comecemos pelos botões “enable” e “disable” para cada canal individual. Estes botões permitem ativar ou desativar um determinado canal sem que haja interferência com os demais (Figura 6.23 e Figura 6.24).

```
dyn_int tibbo_command=makeDynString(6418000, 6419000, 7818000);
bit32 bitArray_enableDisable;
if (channel >=24 && channel<1 )
{
    DebugTN ("Wrong channel");
    return -1;
}
channel = channel-1;
int chanGroup=1+channel/8;
hvRem_getEnableStatus( drawerNumber, chanGroup, bitArray_enableDisable );
int bitPos = channel-(chanGroup-1)*8;
setBit(bitArray_enableDisable, bitPos, enableDisable);
int command = tibbo_command[chanGroup] +bitArray_enableDisable;
int iResult= hvRem_sendReceiveTCP( "0" +command, messageReceived);
if (iResult < 0)
{
    DebugTN("hvRem_enable >> sendReceive failled");
    return -1;
}
channel++;
return 1;
```

**Figura 6.23:** Código que define a funcionalidade dos botões "enable" e "disable"

```
chanGroup=chanGroup-1;
for (int i=1;i<=8;i++)
{
    bool stat;
    dpGet(getSystemName()+"hvRemDrawer"+drawerNUmber+".pmt"+(i+chanGroup*8)+".enabled",stat);
    setBit(chanStatus,(i-1),stat);
}
DebugTN("chanStatus", chanStatus);
```

**Figura 6.24:** Código de verificação do estado de um canal

O código da Figura 6.23 separa os canais em grupos de 3, uma vez que estão “ligados” a diferentes expansores na placa *HV-Remote* pelo que vão ter o seu respetivo comando de mensagem. Após a separação correspondente do grupo a que pertence determinado canal, o programa verifica o estado desse canal quando se carrega num botão “enable” ou “disable” de um determinado canal e tal verificação é feita pela função da Figura 6.24. Após estar tudo confirmado o programa converte a

mensagem a enviar no código correto associado a esse canal e envia a informação à função responsável por enviar mensagens via TCP.

No caso de um utilizador querer ativar ou desativar os canais todos de uma vez, o painel dispõe de dois botões, um de “enable all” e outro de “disable all”; tem também um botão “Read All” que permite efetuar leituras de todos os canais, sucessivamente, e de uma só vez (Figura 6.25).

```
int command;
string messageReceived;
dyn_string tibbo_Commands;
if (enableDisable)
{
  tibbo_Commands = makeDynString ("06418255", "06419255", "07818255");
}
else
{
  tibbo_Commands = makeDynString ("06418000","06419000","07818000");
}
for (int i=1 ; i<= dynlen(tibbo_Commands); i++)
{
  int iResult= hvRem_sendReceiveTCP(tibbo_Commands[i], messageReceived);
  if (iResult < 0)
  {
    DebugTN("hvRem_enableDisable >> sendReceive failled");
    return -1;
  }
}
return 1;
```

**Figura 6.25:** Função de "enable all" ou "disable all"

O código da Figura 6.25 é uma versão simplificada das funções anteriores de “enable” e “disable” uma vez que tem como objetivo ativar ou desativar todos os canais de uma só vez, mas tal como acontecia com aquelas, esta função também necessita de confirmar o estado em que estão os canais.

Esta função, em conjunto com as anteriores, contribui para que, dependendo da situação em que se encontra o detetor, seja possível desativar todos os canais de uma vez ou individualmente. O mesmo é válido no caso da ativação.

O painel possui ainda um botão “apply” que permite aplicar determinados valores de tensão. Junto desse botão existe um campo de escrita “set HV” que se destina à inserção de valores de alta tensão, e em cima tem um campo de leitura de valores de alta tensão “hv Out” que permite ler a tensão que foi aplicada ao canal, por forma a garantir que está tudo conforme o que foi introduzido (Figura 6.26).

```
int firstCommand;
string messageReceived;
string hvSetCommand;
int maxHV=950;
int conv=ceil(hv*4095/maxHV);
channel = channel-1;
if (channel <=7)
{
  firstCommand=7718224;
}
else if (channel >=7 && channel <=15)
{
  firstCommand=7718007;
}
else if (channel >=15 && channel <=23)
{
  firstCommand=7719224;
}
else
{
  DebugTN ("Channel does not exist...exiting");
  return -1;
}
DebugTN("1" +(string)firstCommand);
hvRem_sendReceiveTCP(firstCommand,messageReceived);
hvSetCommand=dacWrite(channel,conv);
int setHV=hvRem_sendReceiveTCP(hvSetCommand,messageReceived);
return setHV;
```

**Figura 6.26:** Função que implementa a aplicação da tensão nos PMTs.

A função da Figura 6.26 é responsável pela tensão que é aplicada aos PMTs do detetor. Esta está limitada superiormente a uma tensão de 950 V, que corresponde ao valor 4095 no DAC, isto é, quando todos os bits apresentam o valor 1. Qualquer valor de tensão introduzido no painel no campo "hv" será convertido para binário e este resultado será enviado para um dos DACs, consoante o canal visado.

Em cima, na secção "other parameters" apresentam-se campos que correspondem a leituras de parâmetros associados a outros componentes da placa *HV-Remote*, já mencionados anteriormente, como sensores de temperatura "Temp 1" e "Temp 2", as tensões de referência que são aplicadas no ADC e nos DACs, e um campo associado ao canal de testes "D_Test". Todas as funções apresentadas anteriormente e as funções que dizem respeito a estes parâmetros podem ser consultadas em anexo.

Na próxima página apresenta-se uma imagem do painel. No painel é possível observar alguns quadrados vermelhos, o que significa que o canal (canais) não está ativo (ativos). Quando é feito o "enable" de um canal, o campo vermelho passa a verde, isto é o canal fica ativo.

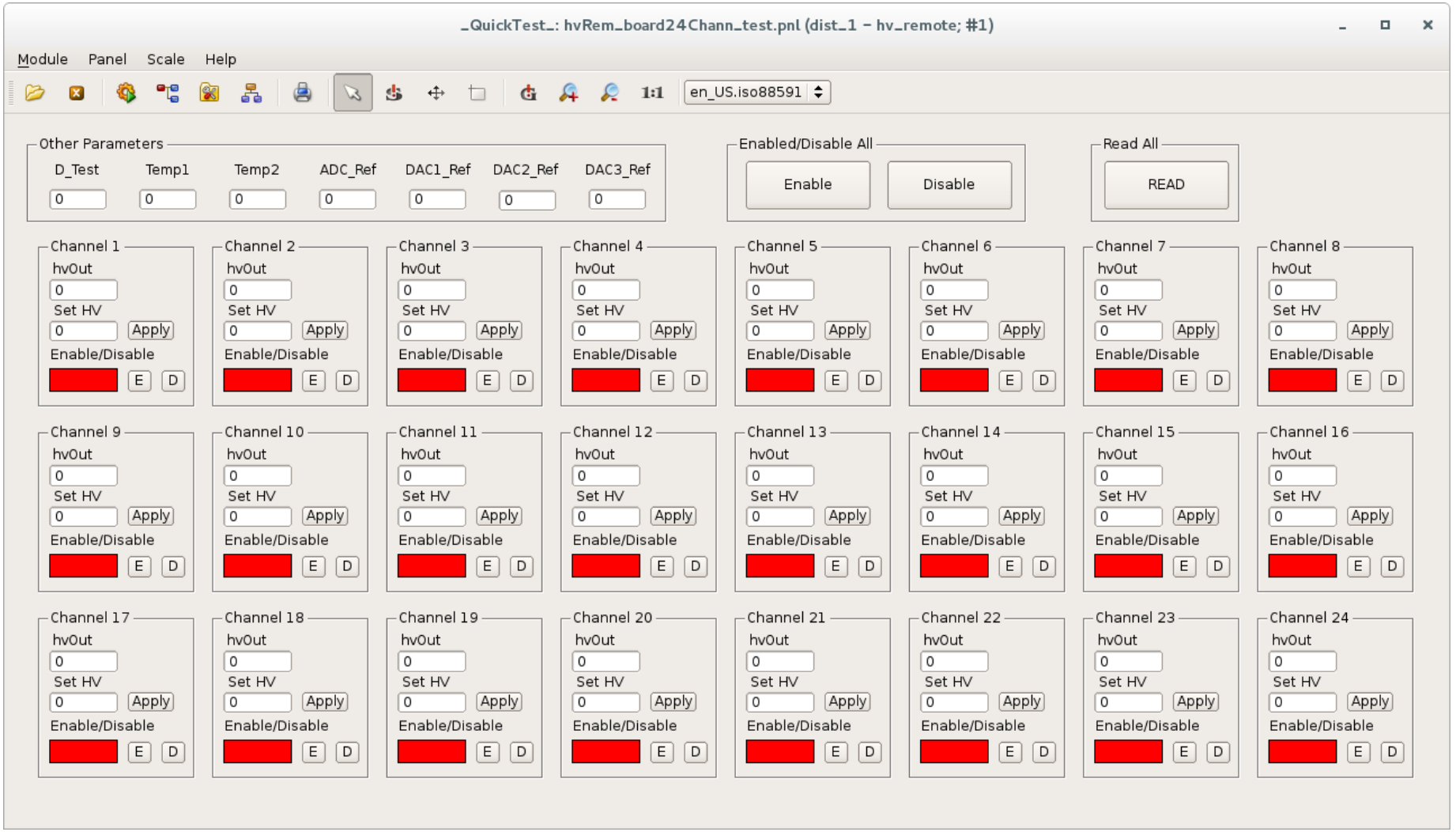


**Figura 6.27:** Painel do DCS

Todos os campos descritos no painel da figura anterior estão relacionados com "*datapoints*" (objetos criados na memória para guardar valores de medições, que permanecem na memória até serem apagados ou alterados) que vão atualizando os valores lidos e escritos ao longo tempo. A estrutura de "*datapoints*" chama-se "*datapoint type*" e cada estrutura define apenas um tipo de "*datapoints*". A estrutura de *"datapoints"* criada para a *HV-Remote* encontra-se na figura seguinte.

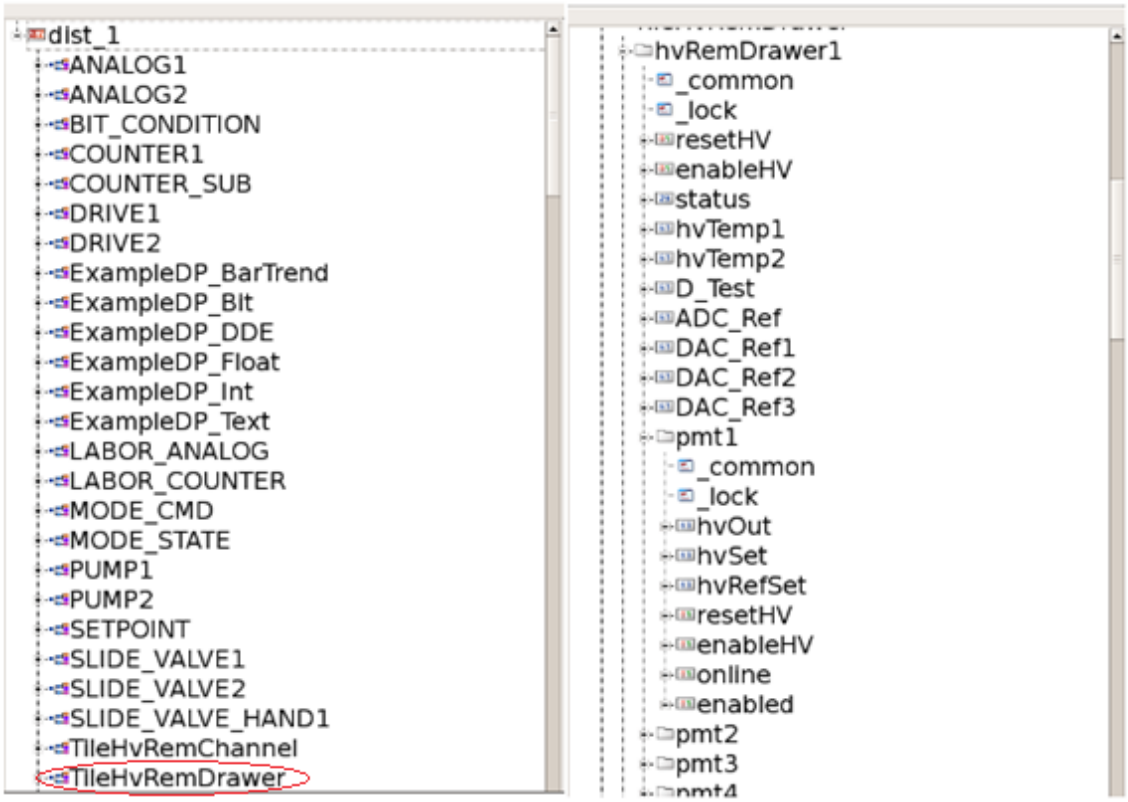


**Figura 6.28:** Estrutura dos *datapoints*

Na Figura 6.28 apresenta-se uma vasta lista de "*datapoints*" (do lado esquerdo da figura) e aquele de interesse encontra-se assinalado a vermelho. No lado direito da figura é possível ver o conteúdo do "*datapoint type*" TileHvRemDrawer, que apresenta um "*datapoint element*" para cada um dos parâmetros da *HV-Remote*. Não é possível confirmar pela imagem, mas este "*datapoint type*" está preparado para a versão final do painel que contará com 48 PMTs em vez de 24, isto é, será controlada uma *drawer* completa, embora cada placa *HV-Remote* seja responsável apenas por 24 PMTs, o que vai requerer duas cartas *HV-Remote* por *drawer*.

### 6.3.3. Testes ao sistema

Os testes realizados ao sistema em laboratório foram limitados, uma vez que quando a programação do painel e do módulo TIBBO foram concluídas, nem a placa *HV-Remote*, nem a placa *HV-Remote-Ctrl* se encontravam prontas para ser testadas. Para ultrapassar esta situação, utilizou-se uma *breadboard* onde apenas foi testado o componente expansor (MCP23017 e MCP23S17) pois não dispúnhamos dos restantes componentes que constituem a placa (DAC, ADC e MUX) com encapsulamento DIL (ou DIP), apenas existiam com encapsulamentos apropriados a soldaduras SMD que não são compatíveis com uma *breadboard*. Nos testes, utilizaram-se LEDs com o intuito de simular canais ativos (LEDs acessos) ou inativos (LEDs apagados). Utilizou-se também o osciloscópio, para observar os sinais, inferir se a funcionalidade da interface SPI estava presente e para confirmar que as mensagens eram enviadas corretamente, como se mostra na Figura 6.29.

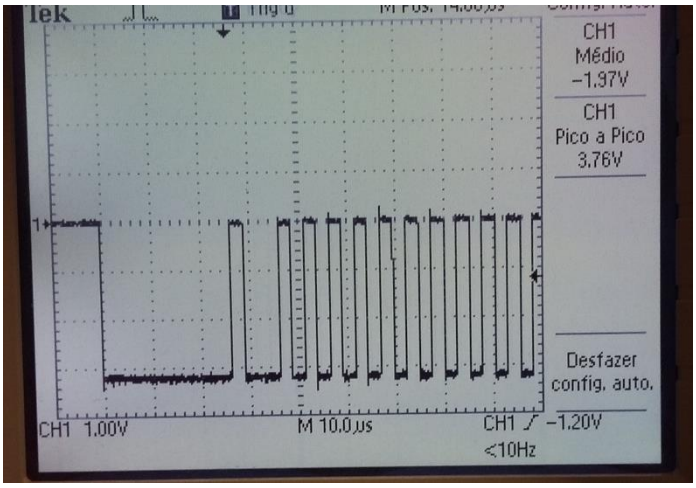


**Figura 6.29:** Sinal no osciloscópio correspondente a uma mensagem enviada.

Sempre que uma mensagem é enviada corretamente, e dependendo do tamanho da mensagem, são vistos 8 impulsos de relógio por cada bloco de informação (endereço do chip, endereço de um registo, valor enviado para esse registo).

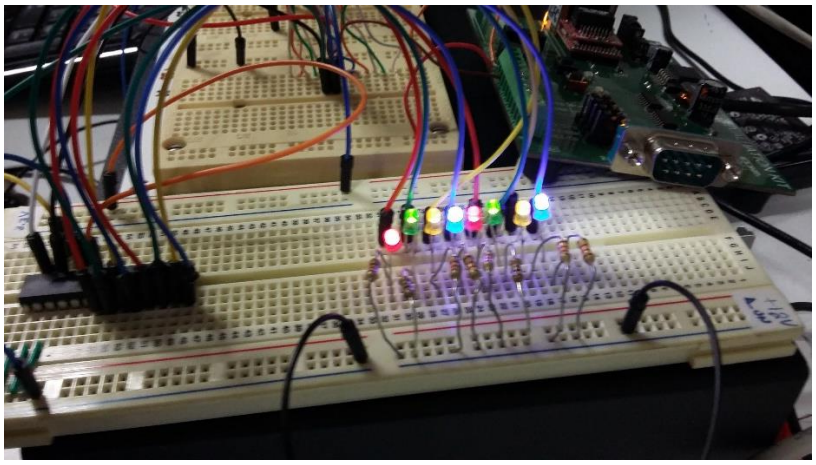

**Figura 6.30:** Montagem experimental

Na Figura 6.30 é apresentado o exemplo em que todos os canais estão ativos. Nesta configuração é utilizado um chip com os pinos de endereço todos ligados à massa ($A_2$=0, $A_1$=0, $A_0$=0). Na Figura 6.31 estão ativos os canais impares e inativos os pares; na Figura 6.32 estão ativos os canais pares e inativos os canais impares.

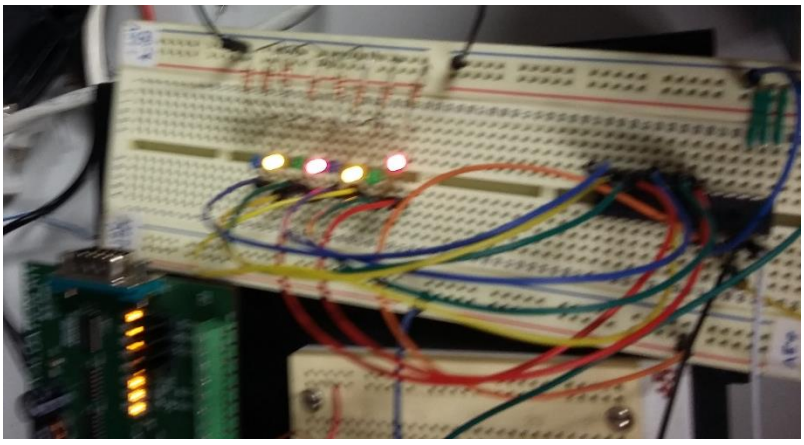

**Figura 6.31:** Canais impares ativos

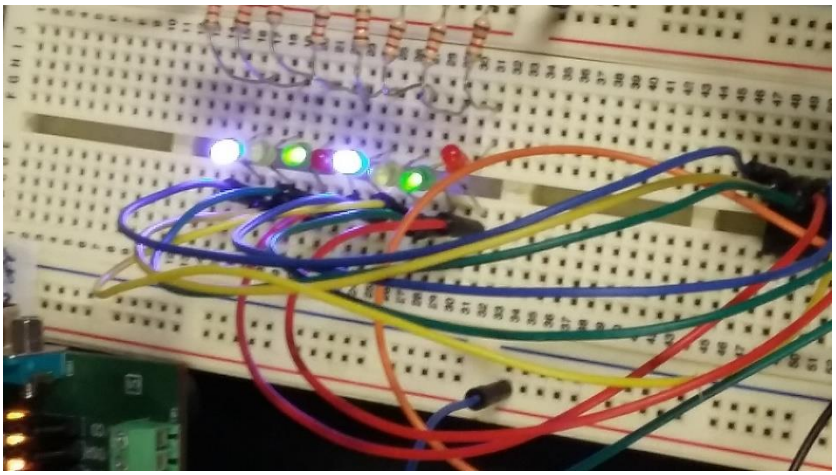

**Figura 6.32:** Canais pares ativos

Será necessário, no futuro, testar o sistema com os restantes componentes, mais concretamente ligado à placa *HV-Remote* uma vez que é para esta finalidade que o sistema de controlo se destina.

# 7. Conclusão

Tendo em conta os objetivos estabelecidos no início do trabalho, conclui-se que o principal foi alcançado, isto é, o sistema de controlo ficou operacional, embora não tenha sido possível testá-lo com o protótipo da carta *HV-Remote* porque este ainda não se encontra disponível. O sistema de controlo foi testado em conjunto com a placa de teste TIBBO e com um chip expansor, uma vez que os restantes componentes não estavam disponíveis com encapsulamento DIP (Dual In-Line Package) ou DIL e a placa *HV-Remote-Ctrl* ainda se encontrava a ser verificada e testada.

Terminada a primeira parte do projeto, foi necessário estudar o funcionamento dos chips que constituem o protótipo da carta *HV-Remote* que pretende vir a substituir o atual sistema em funcionamento no detetor. Embora tenha tido contacto com circuitos eletrónicos ao longo do percurso académico, esta foi a primeira experiência real que envolveu a configuração de registos internos de integrados algo complexos por forma a realizar instruções de controlo, sendo de destacar o expansor, o DAC e o mux. Este contacto com sistemas eletrónicos reais já com alguma complexidade foi uma experiência muito valiosa relativamente à participação em projetos semelhantes no futuro.

O programa desenvolvido para o módulo TIBBO, para funcionar em conjunto com o sistema de controlo (Anexo B), apresenta uma estruturação semelhante ao do *Arduino* em termos de software, isto é, após ser executado o *setup* no início, fica a funcionar em *loop*. Apresenta dois ramos de execução, um que permite escrever apenas para os DACs, e o outro ramo que permite escrever para os restantes componentes, passando a informação em ambas as situações através do expansor. O código foi estruturado desta forma, pois o DAC é o componente que apresenta um procedimento de controlo mais moroso, uma vez que para se lhe aceder é necessário enviar uma mensagem maior em comparação à dos restantes componentes.

Do lado do sistema de controlo (DCS), a aprendizagem continuou, uma vez houve que lidar novamente com uma situação nova. A interface desenvolvida tem um código bastante extenso com várias centenas de linhas. O código principal, responsável por estabelecer comunicação via *Ethernet* com a placa de testes TIBBO, utilizando o barramento SPI, foi estruturado como uma biblioteca, o que permite que as funções possam ser usadas individualmente. Contém uma função de leitura e uma função de escrita gerais e várias outras funções, cada uma delas associada a um componente do protótipo da placa *HV-Remote*.

No final, e tendo em conta que apenas existiam expansores avulso disponíveis para realizar testes em *breadboard*, ligou-se-lhes o sistema e testou-se a comunicação. Tal como na situação anterior que envolvia *Arduinos*, foram utilizados vários LEDs para verificar se as instruções eram enviadas e recebidas corretamente. Mais uma vez, os resultados foram positivos pelo que o objetivo principal deste trabalho foi concluído com sucesso. A ponte *Ethernet* SPI entre o DCS do detetor ATLAS e as placas *HV-Remote* foi estabelecida e encontra-se a funcionar, apesar de ainda estar num estádio de desenvolvimento correspondente a um protótipo.

## 7.1. Trabalho futuro

No futuro, e uma vez que este projeto tem como destino o detetor TileCal da experiência ATLAS, este painel de controlo, caso o projeto do grupo seja selecionado para uma implementação no detetor,

vai ter de ser aplicado a 256 placas *HV-Remote*, cada uma com o seu respetivo endereço de IP, como se pode ver no esquema da Figura 7.1.

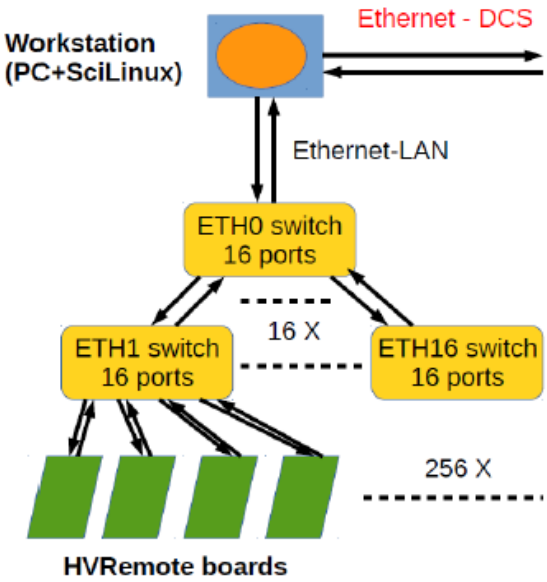


**Figura 7.1:** Arquitetura do sistema de controlo.

Com esta situação em mente adquiriu-se um *switch* Ethernet para testar o comportamento do sistema quando existem múltiplas placas na mesma ligação. O sistema respondeu de forma positiva tendo em conta que nesta fase ainda se tem de configurar o IP manualmente. Na versão final, após a configuração de todos os IPs, o sistema terá uma lista com todos os endereços, o que vai permitir que a navegação entre placas seja automática.

Além desta situação, pretende-se que o painel desenvolvido seja ampliado para funcionar numa gaveta, isto é, seja adaptado para controlar os 48 PMTs, em vez de 24, o que vai requerer duas placas *HV-Remote* por gaveta, sendo cada uma delas responsável por 24 PMTs. Nesta situação, o painel passará a ter o dobro dos PMTs que apresenta de momento, bem como o dobro dos restantes componentes associados a cada placa *HV-Remote*. Pretende-se também que as leituras sejam feitas de forma automática, isto é, sempre que uma determinada placa for selecionada e após ser dado o comando "Enable All", o sistema fará uma leitura automática, a cada segundo, dos valores de tensão aplicados aos PMTs e aos restantes componentes dessa placa. Isto implica o desenvolvimento de um "control script" adequado ao efeito, que irá substituir o botão "Read All" do painel.

# 8. Referências


[1] Maxim Integrated, “MAX3000E/MAX3001E/MAX3002–MAX3012,” Maxim Integrated, San Jose, CA, 2008.

[2] Texas Instruments, “12-Bit Digital-to-Analog Converter,” Texas Instruments Incorporated, Dallas, Texas, 2014.

[3] Microchip, “MCP23017/MCP23S17,” Microchip Technology Incorporated, 2007.

[4] Texas Instruments, “MPC506A CMOS Analog Multiplexer,” 2003.

[5] H. J. Burckhart, J. Cook, V. Filiminov, T. Franz, O. Gutzwiller, V. Khomutnikov e S. Schlenker, “The Common Infrastructure Control of the ATLAS experiment,” Naxos, Grécia, 2008.

[6] A. B. Poy, H. Burckhart, L. Carminati, J. Cook, V. Filimonov, V. Khomutnikov, Y. Ryabov e F. Varela, “Hierarchical Control for the ATLAS Experiment,” Genebra, 2005.

[7] G. Ribeiro, H. Santos, F. Vinagre, S. Nemecek, G. Arabidze e P. Lafarguette, “Detector Control System of the ATLAS Tile Calorimeter,” Grenoble, França, 2011.

[8] ATLAS Inner Detector Community, “Inner Detector Technical Design Report Volume I,” 1997.

[9] ATLAS Collaboration, “ATLAS Liquid Argon Calorimeter Phase-I Upgrade Technical Design Report,” 2013.

[10] ATLAS/Magnet Project Collaboration, “ATLAS Magnetic System Technical Report,” 1997.

[11] S. Schlenker e A. P. Barriuso, “FSM Integration Guideline,” 2007.

[12] J. Sabino, “Realização de uma ponte Ethernet-SPI para comunicação e controlo de placas de sensores (HVopto) do detetor TileCal da experiência ATLAS do LHC – implementação e teste da interface SPI,” Lisboa, 2016.

[13] C. Rato, “Realização de uma Placa de Controlo da Carta de Alta Tensão do Calorímetro TileCal,” Lisboa, 2016.

[14] M. G. D. Gilchriese, “The ATLAS Inner Detector,” SLAC Beam Line 27N4, Livermore, 1997.

[15] A. Valero, “A new read-out architecture for the ATLAS Tile Calorimeter,” Valência, 2015.

[16] B. D. Girolamo, “An overview of the ATLAS TILECAL hadronic calorimeter,” Pisa, 1997.

[17] P. Puzo, “ATLAS calorimetry,” Orsay Cedex, 2002.

[18] Y. Ermoline, H. Burckhart, D. Francis e F. J. Wickens, “ATLAS DAQ/HLT rack DCS,” 2007.

[19] Z. Meng, “Performance of the ATLAS liquid argon calorimeter,” 2010.

[20] C. A. L., “Performance of the ATLAS Tile Calorimeter,” Valência, 2015.

[21] C. Rato, J. Sabino, L. Gurriana, L. Seabra, A. Gomes, G. Evans, J. S. Augusto e A. Maio, “The Interface and Control System of the Upgraded HVOpto/HVRemote Card of the TileCal,” Lisboa, 2016.

[22] C. Fischer, “Study of TileCal Scintillator Irradiation using the Minimum Bias Integrators,” Barcelona, 2016.

[23] A. Solodkov, “Performance of the ATLAS Tile Calorimeter,” São Petersburgo, 2015.

[24] ATLAS Collaboration, “The ATLAS Experiment at the CERN Large Hadron Collider,” Genebra, 2008.

[25] A. Henriques, “The ATLAS Tile Calorimeter,” CERN, 2015.

[26] H. Santos, “The ATLAS/TileCal Detector Control System,” IEEE, 2010.

[27] M. Imhäuser, K. H. Becks, S. Kersten, P. Kind, P. Mättig e J. Schultes, “The Control System for the ATLAS Pixel Detector,” ICALEPCS, Kobe, 2009.

[28] H. P. Wellisch, W. Roberts e C. J. Oram, “The High Voltage Distribution System of the ATLAS Hadronic Calorimeter,” Vancouver, British Columbia, 1995.

[29] S. Palestini, “The Muon Spectrometer of the ATLAS Experiment,” Siena, 2002.

[30] F. Carrió, D. Álvarez, V. Castillo, L. Cerdá, A. Ferrer, L. Fiorini, Y. Hernández, E. Higón, P. Moreno, C. Solans e A., “The PreProcessors for the ATLAS Tile Calorimeter,” IEEE, 2015.

[31] J. Pina, A. Gomes, C. Nuno, T. Vieira, L. Cardoso, B. Peralva, G. Arabidze, N. Giokaris, M. Ouchrif e L. Sargsyan, “The TileCal Detector Control System,” ICALEPCS, Knoxville, Tennessee, 2007.

[32] Texas Instruments, “TLV2541 Serial Analog-to-Digital Converter,” Texas Instrumentes, 2000.

[33] E. V. Santurio, “Upgrade of Tile Calorimeter of the ATLAS Detector,” Estocolmo, 2016.

[34] G. Evans, J. Augusto, A. Gomes e L. Gurriana, “HVOPTO Board,” Lisboa, 2015.

# Anexo A

Excerto do código desenvolvido para o teste do módulo TIBBO EM1206

## A.1 Código *Arduino*

A função seguinte configura os pinos do *Arduino* e ativa o modo SLAVE.

```
void setup()
{
  Serial.begin(9600); //inicializa-se a porta série para debug.
  pinMode(led, OUTPUT); //Inicializa-se o pino do LED do slave.
  digitalWrite(led, ledState);
  SlaveInit(); //Inicializa o slave.
  Serial.println("Slave Initialized");
}
```

A função seguinte rege o comportamento do SLAVE, começando por verificar se está ou não selecionado para o efeito.

```
void loop()
{
  if (!digitalRead(SS)) //Slave ativo?
  {
    if (SSlast != LOW)
    {
      pinMode(MISO, OUTPUT); //Se sim, colocar o pino MISO como Output.
      Serial.println("***Slave Enabled."); //Debug
      byte rx = SPItransfer(255);
      if (rx == cmdBtn)
      {
        byte rx = SPItransfer(cmdBtn); //Reconhecimento de cmdBtn
        ledState=!ledState; //Altera o estado do LED
        digitalWrite(led, ledState);
      }
      else
      {
        byte rx = SPItransfer(255); //Comando não reconhecido
      }
      SSlast = LOW; //Actualiza SSlast.
    }
  }
  // Caso o slave não esteja selecionado
  else
  {
    if (SSlast != HIGH) //Caso o ultimo estado de selecção do slave seja LOW, ou seja, não
esteja selecionado.
    {
      pinMode(MISO, INPUT); //Pino MISO é colocado como Input
      SSlast = HIGH; //Actualiza SSlast.
    }
  }
}
```

## A.2 Código do módulo TIBBO EM1206

A função seguinte configura o canal de Sockets do módulo.

```
void Socket_select()
{
//Configuração do módulo Socket
   net.ip="169.254.35.211";
   sock.num=1;
   sock.protocol=PL_SOCK_INTERFACE_NET;
   sock.inconmode=PL_SOCK_INCONMODE_ANY_IP_ANY_PORT;
   sock.reconmode=PL_SOCK_RECONMODE_3;
   sock.localportlist="1000";
   sock.rxbuffrq(1);
   sock.txbuffrq(1);
   sys.buffalloc();
}
```

A função seguinte configura os pinos SPI (CS, CLK, MOSI e MISO) e o canal SSI do módulo para a comunicação.

```
void setup_ssi()
{
//GPIO-15, Linha 3 do módulo de internet sem fios
   io.num=SPI_CS;
   io.state = LOW;
   io.enabled=YES;
//GPIO-14, Linha 10 do módulo de internet sem fios
   io.num=SPI_CLK;
   io.state = HIGH;
   io.enabled=YES;
//GPIO-13, Linha 5 do módulo de internet sem fios
   io.num=SPI_MOSI;
   io.state = LOW;
   io.enabled=YES;
//GPIO-12, Linha 9 do módulo de internet sem fios
   io.num=SPI_MISO;
   io.state = LOW;
   io.enabled=NO;
//Configuração do canal SSI
   ssi.channel=0;
   ssi.enabled=NO;
   ssi.mode=PL_SSI_MODE_0;
   ssi.clkmap=SPI_CLK;
   ssi.dimap=SPI_MISO;
   ssi.domap=SPI_MOSI;
   ssi.zmode=PL_SSI_ZMODE_ALWAYS_ENABLED;
   ssi.direction=PL_SSI_DIRECTION_LEFT;
   ssi.baudrate=40;
   ssi.enabled=YES;
}
```

A função seguinte descreve como são enviados os dados do módulo (Master) para o *Arduino* (Slave).

```
void Send_Data_Values(string value,int word_size)
{
   io.lineset(SPI_CLK,LOW); //Coloca o pino do relógio em "LOW"
   ssi.value(value,word_size); //Envia dados pelo canal SSI
   io.lineset(SPI_CLK,HIGH); //Coloca o pino do relógio em "HIGH"
}
```

# Anexo B

## Excertos do código do painel DCS e do Módulo TIBBO EM1206

### B.1 Código do módulo TIBBO EM1206

```
//Variáveis globais
string info;
unsigned long data_0[3], data_1[7];
```

A função seguinte discrimina para qual chip escrever e o que escrever no chip pretendido.

```
//Função para escrever para os vários chips
void work_function(string split)
{
   int choice=left(split,1);
   switch(choice)
   {
   case 0:
      data_0[0]=mid(split,2,2); //Guarda o endereço do chip
      data_0[1]=mid(split,4,2); //Guarda o endereço do registo do chip
      data_0[2]=right(split,3); //Guarda o valor escrito no registo do chip
      mcp23s17_function(data_0[0],data_0[1],data_0[2]); //Escreve para o chip
      break;

   case 1:
      data_1[0]=mid(split,2,2); //Guarda o endereço do chip
      data_1[1]=mid(split,4,2); //Guarda o endereço do registo do chip
      data_1[2]=mid(split,6,3); //Guarda o valor escrito no registo do chip
      data_1[3]=mid(split,9,2); //Guarda os bits de controlo e os prefixos
      data_1[4]=mid(split,11,3); //Guarda os bits de endereço os primeiros quatro bits
de informação
      data_1[5]=mid(split,14,3); //Guarda os restantes oito bits de informação
      data_1[6]=mid(split,17,4); //Guarda o resto da informação
      //Escreve para o expansor e posteriormente para o DAC

dac7568_function(data_1[0],data_1[1],data_1[2],data_1[3],data_1[4],data_1[5],data
_1[6]);
      break;

   default:
      //Por defeito envia notificação de um endereço errado
      sock.setdata("Wrong choice!\n");
      sock.send();
   }
}
```

As funções seguintes permitem escrever para o expansor (MCP23S17), sendo que no primeiro caso (mcp23s17_function(…)) é uma definição geral da escrita em si para o expansor, e no segundo caso mcp23s17_setup(…)) é configuração do chip para ser usado como Input/Output.

```
void mcp23s17_function(unsigned long chip_address,unsigned long register_address,unsigned long value)
{
  void chip_select(chip_address); //Chip select
  ssi.str(chr(chip_address)+chr(register_address)+chr(value),1); //Envia informação
  void chip_deselect(chip_address); //Chip deselect
}

//Configuração dos expansores
void mcp23s17_setup(unsigned long chip_address_stp,unsigned long register_address_stp,unsigned long value_stp)
{
  mcp23s17_function(chip_address_stp,register_address_stp,value_stp);
}
```

A função seguinte serve para escrever para o DAC e tem por base a função do expansor (mcp23s17_function(…)) acrescido da forma de escrita para o DAC.

```
//Função para escrever no DAC
void dac7568_function(unsigned long dac_chip_address,unsigned long dac_address,unsigned long dac_value,unsigned long prefix_control,unsigned long address_partdata,unsigned long remain_data,unsigned long extra_feature)
{
  mcp23s17_function(dac_chip_address,dac_address,dac_value);

ssi.str(chr(prefix_control)+chr(address_partdata)+chr(remain_data)+chr(extra_feature),1);
}
```

# B.2 Código para o DCS

Função que controla a comunicação SPI utilizada entre o DCS e o módulo.

```
int hvRem_initTCP()
{
  dyn_errClass err;
  int timeout = 10;
  Socket=tcpOpen("10.101.162.201", "1000", timeout);
  DebugTN(" hv_Rem_initTCP >> tpc open at..." +Socket);
  err=getLastError();
  if (dynlen(err)!=0)
  {
    tcpClose(lastOpendSocket);
    DebugTN("Error: "+ err);
    return -1;
  }
  return Socket;
}
```

Função que controla a troca de informação entre a placa e o DCS.

```
int hvRem_sendReceiveTCP(string command, string &messageReceived)
{
  hvRem_initTCP();
  dyn_errClass err; //error class
  int ret = tcpWrite(Socket, command);
  err=getLastError();
  if (dynlen(err)!=0) // in case of error prints error and returns -1
  {
    DebugTN("Error: "+ err);
    return -1;
  }
  delay(1);
  int read = tcpRead(Socket, messageReceived, 1);
  tcpClose(Socket);
  return 0;
}
```

Função que controla os botões “enable all” e “disable all”.

```
int hvRem_enableDisableAll(bool enableDisable)
{
  int command;
  string messageReceived;
  dyn_string TIBBO_Commands;
  if (enableDisable)
  {
    TIBBO_Commands = makeDynString ("06418255", "06419255", "07818255");
  }
  else
  {
    TIBBO_Commands = makeDynString ("06418000","06419000","07818000");
  }
  for (int i=1 ; i<= dynlen(TIBBO_Commands); i++)
  {
```

```
   int iResult= hvRem_sendReceiveTCP(TIBBO_Commands[i], messageReceived);
   if (iResult < 0)
  {
    DebugTN("hvRem_enableDisable >> sendReceive failled");
    return -1;
   }
  }
  return 1;
}
```

Função que controla os valores de tensão enviados pelo utilizador.

```
int setVoltage(int channel,float hv)
{
  int firstCommand;
  string messageReceived;
  string hvSetCommand;
  int maxHV=950;
  int conv=ceil(hv*4095/maxHV);
  channel = channel-1;
  if (channel <=7)
  {
   firstCommand=7718224;
  }
  else if (channel >=7 && channel <=15)
  {
   firstCommand=7718007;
  }
  else if (channel >=15 && channel <=23)
  {
   firstCommand=7719224;
  }
  else
  {
   DebugTN ("Channel does not exist...exiting");
   return -1;
  }
  DebugTN("1" +(string)firstCommand);
  hvRem_sendReceiveTCP(firstCommand,messageReceived);
  hvSetCommand=dacWrite(channel,conv);
  int setHV=hvRem_sendReceiveTCP(hvSetCommand,messageReceived);
  return setHV;
}
```

Função que controla a escrita no DAC.

```
int dacWrite(int channel,int conv)
{
  int dacAddress;
  string finalCommand;
  channel=channel-1;
  if (channel <=7)
  {
   dacAddress=channel;
   finalCommand="00"+(string)dacAddress+(string)conv+"00";
  }
  else if (channel >=7 && channel <=15)
```

```
  {
    dacAddress=channel-7;
    finalCommand="00"+(string)dacAddress+(string)conv+"00";
  }
  else if (channel >15 && channel <=23)
  {
    dacAddress=channel-16;
    finalCommand="00"+(string)dacAddress+(string)conv+"00";
  }
  else
  {
    DebugTN ("Channel does not exist...exiting");
    return -1;
  }
  DebugTN("1"+finalCommand);
  int hvValue=hvRem_sendReceiveTCP(finalCommand,messageReceived);
  return finalCommand;
}
```

Função que controla a leitura dos valores de tensão

```
int readVoltage(int channel)
{
  int readCommand;
  string messageReceived;
  channel = channel-1;
  if (channel <=15)
  {
    readCommand=7918133+4*channel;
  }
  else if (channel >15 && channel <=23)
  {
    readCommand=7918134+4*(channel-16);
  }
  else
  {
    DebugTN ("Channel does not exist...exiting");
    return -1;
  }
  DebugTN("0" +readCommand);
  hvRem_sendReceiveTCP(readCommand,messageReceived);
  return messageReceived;
}
```

Função que controla o D_Test.

```
int D_Test()
{
   int dtestCommand=7819162;
   string messageReceived;
   hvRem_sendReceiveTCP(dtestCommand,messageReceived);
   return messageReceived;
}
```

Função que controla os valores de temperatura lidos

```
int Temp(int tempDif)
{
   int temp=7819000;
   string messageReceived;
   if(tempDif==1)
   {
      temp=temp+166;
   }
   else if(tempDif==2)
   {
      temp=temp+170;
   }
   else
   {
      DebugTN ("Wrong temperature reference");
      return -1;
   }
   hvRem_sendReceiveTCP(temp,messageReceived);
   return messageReceived;
}
```

Função que controla os valores de referência da tensão do ADC.

```
int adc_ref()
{
   int adcrefCommand=7819174;
   string messageReceived;
   hvRem_sendReceiveTCP(adcrefCommand,messageReceived);
   return messageReceived;
}
```

Função que controla os valores de referência da tensão dos DACs.

```
int dac_ref(int difRef)
{
   int dacrefCommand=7819000;
   string messageReceived;
   if(difRef==1)
   {
      dacrefCommand=dacrefCommand+178;
   }
   else if(difRef==2)
   {
      dacrefCommand=dacrefCommand+182;
   }
   else if(difRef==3)
   {
      dacrefCommand=dacrefCommand+186;
   }
   else
   {
      DebugTN ("Wrong dac reference");
      return -1;
   }
   DebugTN("1" +(string)dacrefCommand);
```

```
    hvRem_sendReceiveTCP(dacrefCommand,messageReceived);
    return messageReceived;
}
```

Função que controla os botões “enable” e “disable” individuais.

**int hvRem_chanEnableDisable(int drawerNumber, int channel, bool enableDisable, string &messageReceived)**

```
{
   dyn_int TIBBO_command= makeDynString(6418000, 6419000, 7818000);
   bit32 bitArray_enableDisable;
   if (channel >=24 && channel<1 )
    {
       DebugTN ("Wrong channel");

       return -1;
    }
    channel = channel-1;
    int chanGroup=1+channel/8;
    hvRem_getEnableStatus( drawerNumber, chanGroup, bitArray_enableDisable );
    int bitPos = channel-(chanGroup-1)*8;
    setBit(bitArray_enableDisable, bitPos, enableDisable);
    int command = TIBBO_command[chanGroup] +bitArray_enableDisable;
    DebugTN("setBit(bitArray_enableDisable, bitPos, enableDisable);",
setBit(bitArray_enableDisable, bitPos, enableDisable));
    int iResult= hvRem_sendReceiveTCP( "0" +command, messageReceived);
    DebugTN("command: ",command);
    if (iResult < 0)
    {
       DebugTN("hvRem_enable >> sendReceive failled");
       return -1;
    }
    channel++;
DebugN(dpSetWait(getSystemName()+"hvRemDrawer"+drawerNumber+".pmt"+channel
+".enabled",enableDisable));
    return 1;
}
```

Função que verifica o estado de cada canal (ativo ou desativo).

**void hvRem_getEnableStatus(int drawerNUmber, int chanGroup, bit32 &chanStatus)**

```
{
    chanGroup=chanGroup-1;
    for (int i=1;i<=8;i++)
    {
       bool stat;

dpGet(getSystemName()+"hvRemDrawer"+drawerNUmber+".pmt"+(i+chanGroup*8)+".
enabled",stat); //control dpe used ot know the enable status
       setBit(chanStatus,(i-1),stat);
  }
  DebugTN("chanStatus", chanStatus);
}
```

# Anexo C

## Prémio de melhor poster na conferência CETC 2016