Francisco Menezes Rodrigues Pinto

Desenvolvimento de uma gateway Ethernet para os robôs CAMBADA

Ethernet gateway design for the CAMBADA robots

Universidade de Aveiro
2020

Departamento de Eletrónica,
Telecomunicações e Informática

Francisco Menezes Rodrigues Pinto

Desenvolvimento de uma gateway Ethernet para os robôs CAMBADA

Ethernet gateway design for the CAMBADA robots

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor José Luís Costa Pinto de Azevedo, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Manuel Bernardo Salvador Cunha, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais e à minha irmã.

o júri / the jury

presidente / president

Prof. Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor António Paulo Gomes Mendes Moreira
Professor Associado, Universidade do Porto - Faculdade de Engenharia

Prof. Doutor Manuel Bernardo Salvador Cunha
Professor Auxiliar, Universidade de Aveiro (Co-Orientador)

agradecimentos /
acknowledgements

A escrita desta dissertação foi possível derivado a toda a ajuda de familiares, amigos e colegas. Por isso quero agradecer a todas as pessoas que constituem o laboratório IRIS-LAB com especial agradecimento aos meus orientadores, Professor José Luís Azevedo e Professor Bernardo Cunha e aos elementos da equipa CAMBADA, Ricardo Dias, João Silva e Filipe Amaral.
Quero agradecer também ao meus colegas e amigos pelas horas dispensadas em todo o apoio dado no decorrer do curso.
Por fim, um especial agradecimento à Mariana Oliveira por todo o apoio dado no percorrer desta etapa e à Francisca Ferreira e ao David Costa por toda a ajuda e apoio que me deram.

Palavras Chave

arquitetura, equipa CAMBADA *gateway*, Ethernet, robótica, microcontroladores, sistemas de tempo real

Resumo

A equipa CAMBADA é uma equipa de futebol robótico constituída por quatro robôs jogadores e um guarda-redes que participa anualmente no campeonato mundial, RoboCup, na liga MSL. Nesta liga, a construção dos robôs é da total responsabilidade das equipas, levando estas a evoluírem constantemente. O robô CAMBADA é constituído por 3 camadas: visão, processamento e sensorização/locomoção. A interligação entre as duas últimas camadas é, atualmente, realizada por um módulo de *gateway* que reencaminha, bidirecionalmente, as mensagens trocadas entre elas. A *gateway* apresenta-se como mais um módulo CAN existente na base do robô, enquanto a sua ligação ao computador portátil é realizada através de uma ligação USB que emula uma porta de comunicação série. Esta solução apresenta algumas fragilidades a nível electro-mecânico e ao nível da comunicação. Como resultado da identificação destes problemas, a presente dissertação descreve o projeto e desenvolvimento de uma nova *gateway* baseada num microcontrolador com a capacidade de ligação às redes CAN e Ethernet. Desenvolveu-se a nova *gateway*. Este desenvolvimento começou por uma pesquisa de quais os microcontroladores com essa capacidade, seguindo-se uma abordagem ao *software*, *firmware* e *hardware*. Inicialmente desenvolveu-se o *firmware* otimizado para ter a capacidade de receber uma mensagem contendo várias tramas com o comportamento do robô. De seguida, procedeu-se à adaptação do software ao nível central de processamento com vista a assegurar a capacidade de comunicação com a nova *gateway* mas mantendo a retrocompatibilidade com as versões existentes. Por fim, na última fase, foi necessário desenhar e instanciar uma placa cumprindo os requisitos que permitam a sua integração direta no robô. Uma vez cumprido o desenvolvimento, foi necessário testar todo o sistema para validar a solução encontrada. Para isso, foram realizados vários ensaios intensivos para caraterizar a nova placa. Verificou-se que o tempo de comunicação de uma mensagem diminuiu substancialmente e a capacidade de transportar mais informação aumentou. Esta melhoria diminuiu o tempo por mensagem em, aproximadamente, 20 vezes e a capacidade de transportar uma mensagem de 400 bytes, num único pacote UDP. Por outro lado, estas alterações trazem uma desvantagem que passa pela quadruplicação do consumo da nova *gateway*, passando a ser de aproximadamente de 200 mA. Concluiu-se que, apesar do maior consumo, as melhorias obtidas favorecem o robô CAMBADA, pelo que esta nova versão passará a ser instalada em todos os robôs.

Keywords

architecture, gateway, ethernet, robotic, microcontroller, CAMBADA team, embeded system

Abstract

The CAMBADA team is a robotic football team made up of four robot players and a goalkeeper who annually participates in the world championship, RoboCup, in the MSL league. In this league, the construction of the robots is the total responsibility of the teams, leading them to constantly evolve. The CAMBADA robot consists of 3 layers: vision, processing and sensing/locomotion. The interconnection between the last two is currently carried out by a gateway module that forwards, bi-directionally, the messages exchanged between them. The gateway presents itself as another CAN module on the base of the robot, while its connection to the laptop is made through a USB connection that emulates a serial communication port. This solution presents some weaknesses at the electro-mechanical level and at the communication level. Based on this problem, and using a microcontroller with the ability to have a CAN and Ethernet connection, a new gateway was developed. This development began by researching which microcontrollers have this capability, followed by an approach to software, firmware and hardware. Initially, the optimized firmware was developed to be able to receive one message with all information from robot's behavior. Then, the adaptation of the software with the ability to communicate with the new gateway while ensuring backward compatibility of versions was developed. Finally, in the last phase it was necessary to design a PCB board with the requirements to allow it to be seamlessly integrated into the robots. After all the development, it was necessary to test the entire system to validate the solution found. For this, several intensive tests were carried out to characterize the new gateway. It has been verified that for a periodic single message the communication time decreased substantially and the capacity to carry more information increased. This improvement decreased the time by approximately 20 times and the ability to carry a 400 byte message, in a single packet. On the other hand this new solution brings a disadvantage that includes the quadruplication of consumption of the new gateway, reaching approximately 200 mA. It was concluded that, the advantage of the new gateway has more benefits and the consumption for now is not an issue for CAMBADA robot, with the same being installed in all robots.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

ASCII	*American Standard Code for Information Interchange*
ACK	*Acknowledge*
ARP	*Address Resolution Protocol*
CAMBADA	*Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*
CAN	*Controller Area Network*
CRC	*Cyclic redundancy check*
DLE	*Data Link Escape*
ETH	Ethernet
ETX	Fim do texto
FCS	*Frame Check Sequence*
FIFA	Federação Internacional de Futebol
GPU	Unidade de Processamento Gráfico
ID	Identificador
IMU	*Inertial Measurement Unit*
IP	Protocolo Internet
ISO	Organização Internacional de Normalização
LED	Díodo Emissor de Luz
MAC	*Medium Access Control*
MBN	Módulo de Baixo Nível
MSL	*Middle Size League*
OSI	*Open System Interconnection*
PCB	Placa de Circuito Impresso
RGB	*Red Green Blue*
RMII	*Reduced Media Independent Interface*
RTDB	*Real Time Data Base*
STX	Início do texto
TCP	*Transmission Control Protocol*
UART	*Universal Asynchronous Receiver/Transmitter*
UCD	Unidade Central de Decisão
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*

CAPÍTULO 1

Introdução

1.1 Motivação

A introdução crescente na indústria, ao longo das últimas décadas, de soluções baseadas em tecnologias de controlo eletrónico e gestão de informação levou ao inevitável problema derivado da intercomunicação entre sistemas, resultante do facto de existir uma grande variedade de protocolos de comunicação, cada um com diferentes especificações que vão desde a parte física da ligação até à parte protocolar. Daí resultou a necessidade de desenvolver formas de transferir a informação, presente em barramentos com características específicas, para outros com características completamente diferentes, procedendo às necessárias conversões de formato, quer elétrico quer de encapsulamento. Atualmente as indústrias têm começado a utilizar o protocolo com base em Ethernet para universalizar as comunicações entre sistemas e assim diminuir o recurso à utilização de protocolos proprietários ou com características específicas e menos universais [1].

Na versão mais recente da equipa de robôs CAMBADA, a infra-estrutura sensorial e de controlo presente no Módulo de Baixo Nível (MBN) é composta por vários sub-módulos baseados em microcontroladores que comunicam entre si através de uma rede CAN. Por outro lado, o principal elemento da Unidade Central de Decisão (UCD) dos robôs é baseado num computador portátil convencional que não disponibiliza interfaces standard para interligação com este tipo de rede. Consequentemente, a comunicação entre estes dois níveis da arquitetura CAMBADA é assegurada por um módulo dedicado que funciona como *gateway* entre eles. Na versão atual dos robôs, esta *gateway* usa um barramento USB que virtualiza uma ligação série entre o computador e o hardware de mais baixo nível.

Ora, a prática tem demonstrado que esta abordagem resulta em problemas de dois tipos. O primeiro ocorre ao nível da velocidade de transporte da informação. Efetivamente, embora a *gateway* recorra a uma ligação USB, esta é adaptada, para

efeitos de ligação à *gateway*, por um módulo específico que emula uma ligação série do tipo RS-232. Para além das limitações ao nível do *baud-rate* permitido, esta solução determina, como se verá mais tarde, a existência de um *jitter* significativo nas comunicações entre os dois níveis da arquitetura. O segundo problema resulta do facto de a UCD apenas disponibilizar ligações mecânico/elétricas adequadas a sistemas que se destinam a funcionar em ambientes não sujeitos a acelerações significativos. Com a evolução natural dos requisitos e da complexidade do jogo efetuado por estes robôs, estes são sujeitos, cada vez mais, a acelerações lineares e/ou rotacionais crescentes, a que se juntam as brutais desacelerações que, ocasionalmente, resultam do choque entre robôs a circular a velocidades de vários metros por segundo. A observação factual (em competição), tem permitido verificar que, devido a essas acelerações/desacelerações, a vibração desencadeada nos contactos elétricos da ficha USB determina, pontualmente, a perda de informação na comunicação entre os dois níveis da arquitetura do robô, podendo mesmo, no caso limite, levar a que essa mesma ficha se liberte fisicamente, cortando definitivamente as comunicações. Tem como consequência a impossibilidade de manter o adequado comportamento do ou dos robôs em que tal aconteça.

1.2 Enquadramento

O projeto CAMBADA começou em 2003 no DETI (Departamento de Eletrónica, Telecomunicações e Informática) e no IEETA (Instituto de Engenharia Eletrónica e Informática de Aveiro) da Universidade de Aveiro. Este projeto tem como objetivo criar, desenvolver e fazer evoluir de forma continuada uma equipa de robôs colaborativos, com capacidade de jogar futebol de acordo com o modelo e regras da *Middle Size League* (MSL) do ROBOCUP [1].

Desde 2003, a esta parte, a equipa CAMBADA tem sofrido um conjunto de alterações acompanhando e contribuindo para a evolução da Liga MSL. As alterações têm vindo a ocorrer a todos os níveis, desde os aspetos mais estruturais, a nível físico e mecânico, até ao desenvolvimento de elevados níveis de abstração em *software* que permitem, por exemplo, a modelação dinâmica de comportamentos táticos da equipa. Isto é possível pela natureza multidisciplinar dos sistemas robóticos em geral, e os da robótica móvel, autónoma e cooperativa em particular. O projeto CAMBADA é assim um projeto em permanente evolução em áreas tão diversificadas como a mecânica, os sistemas embutidos, a eletrónica, a fusão sensorial, o controlo distribuído em diferentes níveis do modelo arquitetural, a visão por computador, as comunicações, a simulação, o registo e análise de dados, a coordenação multi-agente ou ainda, nos níveis mais altos de abstração, a estratégia ou a aplicação de métodos de aprendizagem. Esta evolução é

[1] http://www.robocup.org

permanentemente testada em condições de *benchmarking* reais, que se consubstanciam na realização de competições internacionais nas quais participam outras equipas de investigação com objetivos similares. Assim, e em função da evolução e da avaliação dos resultados obtidos, o projeto tem, em permanência, que responder a desafios a diferentes níveis, e à necessidade de melhorar e fortalecer a resiliência e fiabilidade das soluções desenvolvidas. Foi assim, neste enquadramento, que surgiu a necessidade de assegurar uma solução que visa o aumento e melhoria do desempenho e da fiabilidade da comunicação entre os dois principais níveis arquiteturais de cada um dos robôs.

1.3 Objetivos

Usando como base o modelo da atual *gateway* existente nos robôs, o objetivo do trabalho desenvolvido, e que serve de base a esta dissertação, consistiu no desenvolvimento de uma nova *gateway* entre a rede CAN que interliga os sub-módulos *hardware* do MBN e a UCD, mas agora abdicando da solução baseada em USB (linha série emulada) e adotando, do lado do computador, uma comunicação Ethernet. Com este objetivo em vista, o trabalho desenvolvido foi organizado na seguinte sequência de fases:

- Projetar, desenvolver e instanciar um novo módulo *gateway* capaz de efetuar a interface bidirecional CAN/Ethernet;
- Desenvolver o *firmware* para este módulo que assegure a capacidade de receber uma mensagem a partir de um protocolo, reenviando-a através do outro, o mais rapidamente possível e sem perda de pacotes;
- Adaptar o software existente ao nível do computador central, por forma a que o mesmo possa funcionar, de forma transparente, com qualquer uma das *gateways* (USB/CAN ou Ethernet/CAN);
- Realizar um conjunto exaustivo de testes para validação do novo sistema;
- Proceder à análise dos resultados da solução proposta, comparando-os com os da solução já existente;
- Fundamentar e justificar adequadamente a solução encontrada para o problema.

1.4 Estrutura

Para além do capítulo 1, no qual se discutem brevemente a motivação, enquadramento e objetivos do trabalho, a presente dissertação encontra-se dividida em mais cinco capítulos que abordam, respetivamente, as seguintes temáticas:

- **Capítulo 2:** Apresentação da equipa CAMBADA e das tecnologias usadas no desenvolvimento da nova *gateway*;

- **Capítulo 3:** Análise dos problemas encontrados e apresentação da arquitetura proposta para a sua resolução;
- **Capítulo 4:** Descrição exaustiva do processo de desenvolvimento, bem como a justificação das escolhas feitas durante o mesmo;
- **Capítulo 5:** Descrição dos testes realizados, com a apresentação dos resultados e validação da proposta;
- **Capítulo 6:** Exposição das conclusões que permitem validar a nova *gateway*, bem como a apresentação de potenciais linhas de continuação de trabalho desenvolvido tendo em vista a exploração de potenciais benefícios suplementares para o projeto CAMBADA.

1.5 Contribuição

Ao longo dos últimos 16 anos a equipa CAMBADA tem levado a cabo um trabalho de investigação e desenvolvimento continuado, em variadas áreas das ciências da engenharia, que se tem reafirmado pela capacidade de se manter ao nível dos resultados técnico-científicos das melhores equipas de I&D que regularmente participam nas competições MSL. Pela diversidade de temáticas multidisciplinares que tal esforço representa, este projeto tem vindo, ao longo do tempo, a contar com a participação, envolvimento e dedicação de um alargado número de docentes, investigadores e alunos que, no âmbito do Laboratório de Robótica e Sistemas Inteligentes do IEETA (IRIS) têm contribuído para o seu sucesso.

O trabalho aqui apresentado e defendido ao longo desta dissertação pretendeu colmatar um dos problemas e fragilidades identificados ao nível da comunicação entre a camada de mais baixo nível da arquitetura do robô, MBN e o seu principal elemento de processamento, UCD. Com isso pretendeu-se contribuir quer para o reforço da fiabilidade dos agentes e da equipa como para a melhoria da performance e previsibilidade dessa comunicação. Desta forma contribui-se, entre outras coisas, para assegurar a qualidade do controlo de mais alto nível, através da diminuição do *jitter* nessa comunicação, e para o reforço da escalabilidade desejável do modelo adotado na camada de mais baixo nível da arquitetura dos robôs CAMBADA.

CAPÍTULO 2

Tecnologias

O presente capítulo apresenta o cenário para o qual é proposto o desenvolvimento de uma nova *gateway*, bem como o conceito de comunicações entre sistemas e a parte protocolar para esse efeito. Serão descritas as tecnologias atualmente utilizadas para a ligação entre o computador do robô e a eletrónica de controlo do mesmo. Por fim, serão discutidas as tecnologias que serão utilizadas na nova *gateway* dos robôs.

2.1 Equipa CAMBADA

A *RoboCup Federation*, em 1998, propôs a vários grupos de investigação, que desenvolvessem robôs capazes de, em 2050, defrontar e vencer o primeiro jogo entre as equipas campeãs do mundo de robôs e humanos, desse ano. Neste contexto, o futebol surgiu como área de investigação e desenvolvimento fundamental por duas origens de razões: por um lado representa um desafio de elevada dificuldade cientifica e tecnológica, em particular, no domínio da robótica móvel, autónoma, e de cooperação multiagente; por outro lado, sendo o futebol um desporto amplamente conhecido pela população em geral, este desafio constitui-se assim igualmente numa excelente forma de divulgação de ciência e tecnologia [2].

Com o avançar dos anos, e ainda no âmbito da *RoboCup Federation*, começaram a aparecer, ainda sob o tema do futebol, um número crescente de ligas como sejam o futebol simulado, o futebol com robôs humanóides para além do futebol baseado em robôs com rodas e completamente autónomos, como é o caso da liga MSL. A liga MSL, utiliza, como base, as regras da Federação Internacional de Futebol (FIFA), adaptando-as para o futebol com robôs, em função das suas características e do aumento crescente da dificuldade do desafio. A utilização de robôs com rodas permite, quando comparado com outras ligas como futebol com robôs humanóides, ganhos na estabilidade e na agilidade a nível da locomoção. Esta abordagem permite, por outro lado, mais espaço

para a evolução na comunicação, visão, cooperação entre agentes (robôs) e a evolução dinâmica de táticas e estratégia de jogo.

A equipa CAMBADA nasceu em 2003, na Universidade de Aveiro, com o objetivo de participar na competição mundial de robótica móvel inteligente, *RoboCup*, na liga MSL, que iria decorrer em 2004, em Portugal [3]. A constante participação em competições levou à evolução da equipa, adaptando-se aos novos desafios que todos os anos aparecem [4] [5]. Atualmente, a equipa CAMBADA é constituída por 5 robôs jogadores de campo e 1 robô guarda-redes. A diferença entre os dois tipos de robôs é que o guarda-redes tem uma *Kinect* e a possibilidade de ter uma estrutura que pode estender-se 10 centímetros. Esta, cada vez que é ativada, pode permanecer aberta durante 1 segundo.

Figura 2.1: Robô CAMBADA.

O robô CAMBADA, representado na figura 2.1, tem como dimensões máximas 50 centímetros de comprimento, 50 centímetros de largura e 80 centímetros de altura e um peso máximo de 40 kg [6]. Para além destas dimensões, a sua construção depende dos objetivos e desejos de cada equipa. A construção do robô da equipa CAMBADA é pensado para ser modular e de fácil manutenção. Assim, o robô é dividido em três módulos principais:

- Módulo de Visão: Câmara e espelho, situado na parte superior do robô;

- Unidade Central de Decisão: computador portátil, situado na parte central do robô, UCD;
- Módulo de Baixo Nível: responsável pela locomoção, eletrónica de controlo, módulos sensoriais, sistema de chuto e sistema de controlo de bola, tudo isto situado na parte inferior do robô, MBN.

No topo existe uma câmara com comunicação Ethernet apontada para um espelho, oferecendo a capacidade ao robô de ter uma visão 360º do campo, denominado de *Catadioptric system*. O sistema de visão é usado para a sua localização no campo, a identificação da posição dos obstáculos, dos jogadores e adversários. Para isso, o robô usa as linhas do campo e o mapa de distância, calculado pela relação entre a equação do espelho e o pixel da imagem.

Ao centro do robô encontra-se o computador, onde corre todo o *software* de alto nível do robô, ou seja, o *software* de comunicação entre robôs, comunicação com o microcontrolador, visão e decisão. O sincronismo destes processos é dado pelo momento em que é adquirida uma nova imagem na câmara do robô, processo que ocorre a 50Hz.

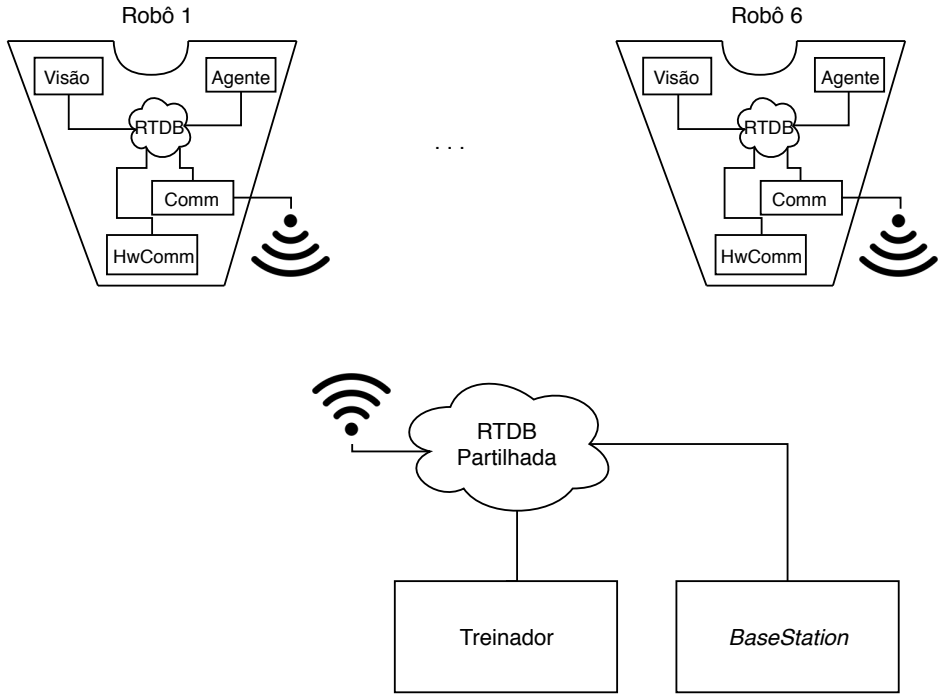


Figura 2.2: Ilustração da comunicação entre processos do robô e o treinador.

A partilha da informação entre os processos do robô e entre os robôs utiliza a *Real Time Data Base* (RTDB), tal como ilustrado na figura 2.2. A RTDB adota o modelo de *black board* a partir do qual todos os processos vão buscar nova informação e onde os mesmo publicam resultados após processamento [7]. Os processos que utilizam a RTDB são:

- **Agente**: processo onde, a partir de informação existente na RTDB, relativa ao treinador, à posição da bola e à dos outros robôs em campo, é calculado o próximo comportamento e atualizados os novos valores na RTDB;
- **Visão**: Recebe uma imagem da câmara e extrai a informação relevante para retirar, posteriormente, a posição e orientação do robô no campo, localização de obstáculos e dos outros robôs, bem como a localização da bola. Toda esta informação é, para cada nova imagem igualmente colocada na RTDB;
- **Comm**: É responsável por extrair a partir da RTDB um subconjunto da informação ali armazenada (aquela que é mais relevante para os restantes agentes da equipa) e proceder periodicamente à sua difusão para os restantes robôs e treinador por forma a que todos os elementos da equipa disponham de informação global sobre o estado atual do jogo;
- **HWcomm**: Processo de comunicação periódico entre Unidade Central de Decisão(UCD) e o Módulo de Baixo nível (MBN). Este processo é iniciado a cada novo ciclo e envia para o MBN a informação mais recente da RTDB. Simultaneamente, existe um subprocesso adicional que está à escuta de novas mensagens produzidas pelos subsistemas do MBN e que por sua vez atualiza a respetiva informação na RTDB;

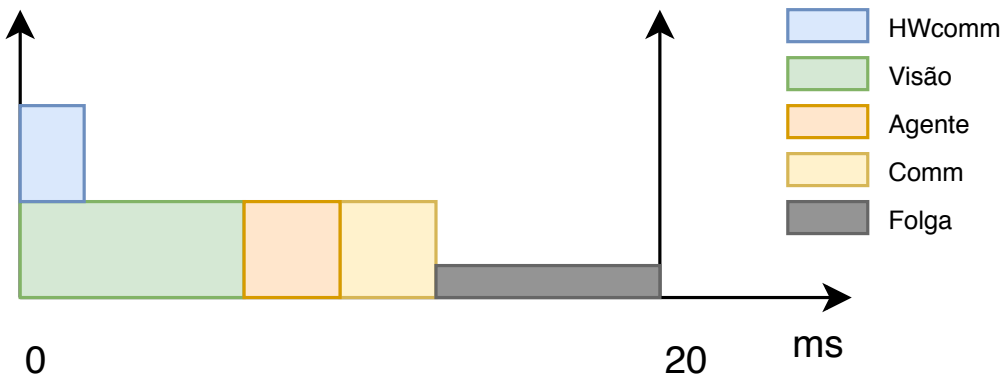


Figura 2.3: Ilustração do escalonamento dos processos a correr em cada ciclo no robô CAMBADA [8].

A sincronização entre processos, representada no diagrama da figura 2.3, mostra a evolução dos mesmos ao longo de um ciclo. No início de cada ciclo, iniciado pela receção de uma nova imagem, o *HWcomm* e a Visão arrancam ao mesmo tempo. No processo Visão, com base na imagem recebida, são retiradas as informações das posições do próprio robô, bem como dos obstáculos, da bola e dos demais robôs em campo, atualizando essa informação na RTDB. Este ciclo é terminado com o envio de um sinal para o processo Agente, de forma a este iniciar. O processo HWcomm divide-se em 2

subprocessos: o de envio e o de receção. O subprocesso do envio da mensagem para o MBN é periódico. Isto acontece no início de cada novo ciclo. A mensagem contém os *setpoints* dos motores, a ordem de chuto, bem como a sua potência e a velocidade dos *grabbers* (motores responsáveis pela manipulação da bola). Já o subprocesso de receção é assíncrono, de forma a permitir atualizar os dados na RTDB, assim que estes são recebidos. O último processo do ciclo é a atualização da informação partilhada pela RTDB. No diagrama 2.3 verifica-se um tempo de folga, uma vez que o processo de visão e do agente podem variar consoante a necessidade do algoritmo. Para as atuais condições de funcionamento do robô, cada ciclo de processamento não pode ultrapassar os 20 ms.

O Módulo de Baixo nível (MBN), representado na figura 2.4, é composto por uma arquitetura distribuída, interligada por uma rede CAN. Este tipo de arquitetura possibilita que o sistema seja modular, simplificando os processos, tanto de adicionar/retirar *hardware*, bem como o processo de manutenção do *hardware* existente. Como a UCD não possui uma ligação CAN, mas sim uma ligação USB, a ponte entre esta e o *hardware* é feita através de uma *gateway*, que tem como função a conversão da informação entre estes dois meios.

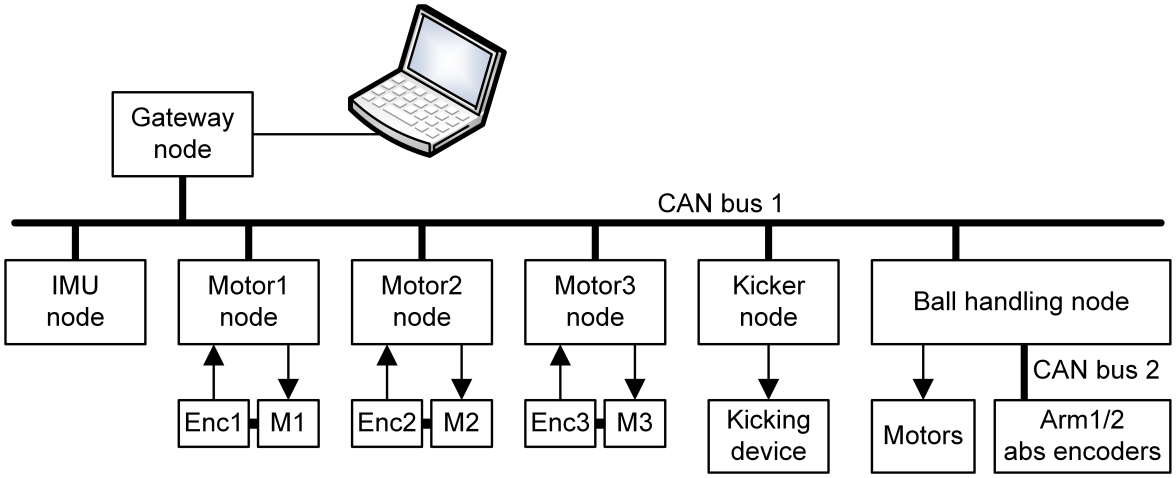


Figura 2.4: Arquitetura do Módulo de Baixo Nível [9].

2.2 Estrutura de rede

No início da comunicação entre computadores, cada fabricante tinha as suas normas, que variavam desde a conectividade, ou seja, modo como é feita a transmissão dos sinais a nível físico e elétrico, até aos protocolos utilizados. A diversidade existente criava um problema na comunicação entre sistemas de várias marcas.

Em 1979, a Organização Internacional de Normalização (ISO) oficializou a criação de um *standard* para a comunicação ponto a ponto [10]. Para isso, foi criado o modelo OSI, facilitando a comunicação entre por vários dispositivos, de vários fabricantes, na mesma rede de comunicação. Este modelo está dividido em dois grandes blocos: o bloco de transporte e o bloco de aplicação, sendo estas divididas em várias camadas, como mostra a figura 2.5 [11]. Uma vantagem desta separação por camadas é que cada uma tem uma função específica, não havendo problemas de sobreposição de funções entre elas [12].

Relativamente ao bloco de transporte, este é responsável pela receção/transmissão do sinal elétrico, independentemente do meio em que a informação viaje, por exemplo, meio elétrico (cabo de rede), meio eletromagnético (*wi-fi*) ou ótico (cabo de fibra ótica). Para além do meio físico, neste bloco é verificado o endereçamento dos pacotes e a existência de erros. Já o bloco de aplicação usa a informação devolvida pelo bloco de transporte e processa-a de acordo com os objetivos das aplicações às quais esta se destina[12].

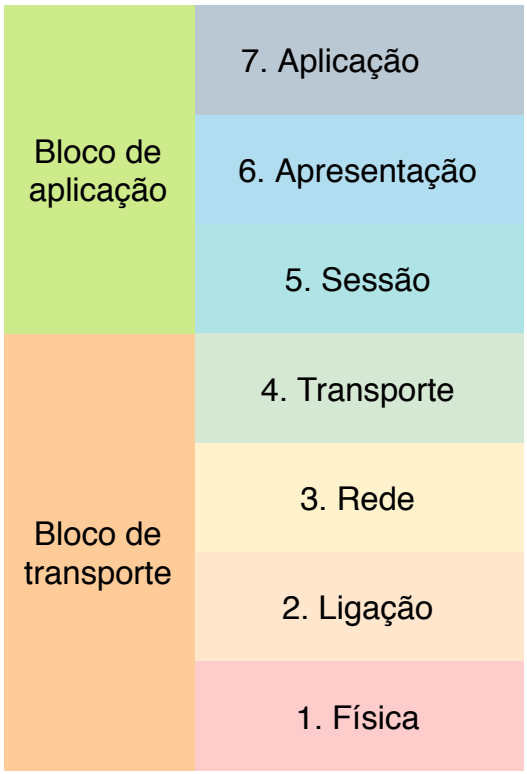


Figura 2.5: Modelo OSI.

Em relação às camadas presentes no modelo OSI, estas são sete, sendo a finalidade de cada uma brevemente descrita abaixo [13]:

1. **Física**: Caraterização do funcionamento elétrico e físico da ligação, ou seja, tensões de alimentação, especificações do sinal (normal ou diferencial), ficha de ligação e seu mapa de pinos, meio de comunicação (cabo ou radiofrequência) e o tipo de cabo (fibra, cobre...). Na perspetiva de um porto de entrada, o sinal que é recebido é por sua vez filtrado, eventualmente desmodulado e finalmente convertido num versão digital do mesmo. Na perspetiva de um porto de saída, o sinal digital a ser transmitido é convertido de acordo com a especificação elétrica desta camada, o que pode envolver ações como adaptação de tensão, geração de sinais diferenciais ou eventual modulação do sinal sobre uma portadora.

2. **Ligação**: Depois do sinal digitalizado, na camada de ligação é verificado se o destino do pacote coincide com o seu endereço físico (*Medium Access Control* (MAC)) e a existência de erros na trama. Para isso, é calculado o *Cyclic redundancy check* (CRC) da trama e comparado com o do pacote recebido. Caso seja igual, considera-se que o pacote recebido não contém erros. Contudo, no caso do pacote ter algum erro, com o uso de algoritmos de recuperação de erros (e.g. método de paridade ou código de *hamming*) tenta-se corrigir o mesmo, caso seja possível.

   Ao pacote de saída é adicionado o endereço físico (MAC) do destino e da origem da trama e é calculado o *checksum*, utilizando os mesmos métodos de verificação dos pacotes de entrada. Para além disso, nesta camada controla-se igualmente o fluxo de dados, tanto da transmissão como da receção, de modo a que este não exceda a capacidade da rede.

3. **Rede**: Com o fluxo de dados tratados, a camada 3 é responsável pelos protocolos de roteamento (assegurado por um equipamento específico normalmente designado por *router*), ou seja, por adicionar ao pacote Ethernet as normas IP (v4 ou v6), ICMP, ou outros. Por exemplo, quando um pacote chega a um *router*, este olha para a zona da camada 3 de modo a saber para onde é necessário redirecionar o mais rapidamente o pacote, uma vez que conhece todos os endereços existentes na rede. No caso de o pacote a enviar ter um tamanho maior que a capacidade de dados máxima da trama, esta camada procede à segmentação do mesmo, bem como à compressão dos vários pacotes fragmentados num só. No caso de haver um erro na criação da trama é criado um relatório de erro de entrega.

4. **Transporte**: A camada de transporte é responsável pela transferência de pacotes entre aplicações(UDP e TCP), controlo de congestionamento de informação (DCCP), entre outros. Os dados a serem transferidos são ainda encapsulados num dos protocolos existentes. Por exemplo, em TCP, um protocolo com garantia de

entrega ou em UDP, o qual não assegura essa mesma garantia.

5. **Sessão**: Esta camada é responsável pela gestão da comunicação entre dispositivos, definindo como vai ser feita a comunicação e quais as caraterísticas da mesma. Por exemplo, caso um dispositivo saia da rede por instantes, é esta camada que reinicia a comunicação.

6. **Apresentação**: A camada de apresentação é responsável pela conversão de diferentes códigos utilizados na comunicação. Por exemplo, no caso de uma aplicação que recebe o pacote codificado em UTF-8 e que necessita de usar essa informação utilizando a sua representação em ASCII, é nesta camada que essa conversão é assegurada. Outra responsabilidade desta camada é a encriptação/desencriptação do pacote, caso este mecanismo seja necessário.

7. **Aplicação**: É a camada de abstração que faz o interface entre o utilizador e todas as restantes camadas do protocolo OSI. Por exemplo, na utilização de uma página web, o utilizador só necessita de interagir com esta camada, não precisando de conhecer todos os processos e detalhes de mais baixo nível implementados no protocolo.

2.3 PROTOCOLOS DE COMUNICAÇÃO

A capacidade de comunicação entre dispositivos, placas controladoras ou computadores tem por base um conjunto de normas, com é o caso das anteriormente referidas. Estas normas vão desde o nível físico (tensões dos sinais) até à caraterização das propriedades da trama, ou seja, velocidade de comunicação, capacidade de deteção de erros e possibilidade de correção.

2.3.1 Ethernet

Nas redes Ethernet, a construção da trama é feita na camada 2 do modelo OSI. Essa construção varia consoante o tipo de protocolo usado nas camadas superiores deste modelo. Existem dois tipos de tramas diferentes, a Ethernet II (usada nos protocolos IP, *Address Resolution Protocol* (ARP), e *Wake-on-LAN*, entre outros) e a Ethernet 802.3 (usada em protocolos IEEE 802.1Q Virtual LANs e IEEE 802.1D Spanning Tree, entre outros). As duas tramas coabitam na mesma rede, sendo função do controlador da rede detetar qual o tipo de trama [13] [14].

A trama da Ethernet II, representada na figura 2.6, mostra a trama utilizada para comunicações TCP/IP ou UDP/IP. A função de cada campo da trama é:

- **Início de trama**: Indicador de início de uma nova trama. Este campo tem a dimensão de 8 *bytes*. Os primeiros 7 *bytes* são o preâmbulo da trama e 1 *byte* que indica o início da trama. A sua função é acordar todos os recetores, bem como assegurar a sincronização do relógio do emissor e dos recetores.
- **Endereço destino**: Endereço MAC do recetor, ao qual a trama se destina. Este campo tem a dimensão de 6 *bytes*.
- **Endereço origem**: Endereço MAC do emissor, do qual a trama foi enviada. Este campo também tem a dimensão de 6 *bytes*.
- **Tipo**: Este campo utiliza 2 *bytes* que permitem identificar qual o tipo de dados que vão ser enviados no campo Data. Por exemplo, uma mensagem ARP é identificada com o valor 0x0806. Já uma mensagem do tipo IP é identificada com o valor 0x0800.
- **Data**: Campo onde são transportados os dados, tendo um tamanho variável entre 46 a 1500 *bytes*.
- ***Frame Check Sequence* (FCS)**:Este campo é usado para verificar se a trama tem algum erro, no momento da sua receção. O método utilizado para a deteção de erros é o CRC-32.

Figura 2.6: Trama EthernetII.

2.3.2 Protocolo Internet

A trama IP pertence à camada de rede, camada 3 do modelo OSI. Este protocolo já existe há mais de 30 anos, havendo já 2 versões: IPv4 e IPv6. A diferença entre estas é o maior número de clientes suportados no IPv6.

O transporte da trama IP é feito no campo de dados da trama MAC, como representado na figura 2.7. No início da trama IP verifica-se que esta é dividida entre o cabeçalho (com 12 *bytes*), os IP de origem e o de destino (cada um de 4 *bytes*), um campo para opções (com 4 *bytes*) e, por fim, o campo de dados. O cabeçalho inclui várias informações: versão da trama IP utilizada, o seu tamanho, um identificador e se esta é uma trama repartida, ou seja, se ela faz parte de um pacote maior que teve de ser dividido. Também no cabeçalho há um campo que indica qual o protocolo do campo de dados. O campo de opção só é utilizado para *routing* opcional e questões relacionadas com o tempo de comunicação da trama [14].

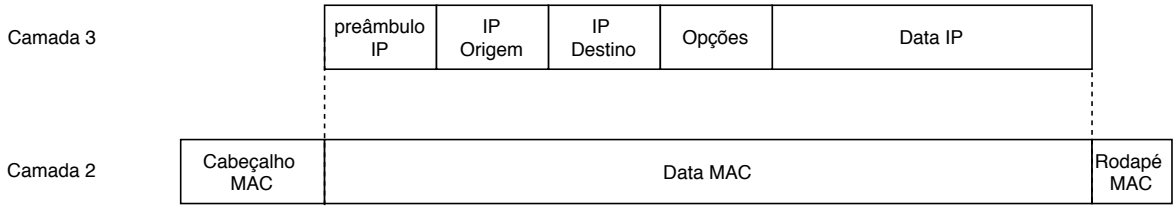


Figura 2.7: Trama IP.

2.3.3 UDP

O protocolo UDP, encapsulado no campo de data da trama IP, pertence à camada de transporte, camada 4 do modelo OSI. Este protocolo é usado para a comunicação ponto a ponto sem garantia de entrega. Por isso, as vantagens da utilização do mesmo são o seu diminuto tempo de transmissão (o que é ideal para aplicações de tempo real) e a não necessidade de ser criada uma conexão entre o emissor e o recetor. Contudo, uma desvantagem deste protocolo é o facto de, no caso da perda de um pacote, a aplicação que o enviou não tomar conhecimento que o recetor não o recebeu [11]. Estas situações de perda de pacotes poderão ser devidas, por exemplo a uma sobrecarga na rede.

A figura 2.8, mostra a trama UDP, tal como é encapsulada nas camadas subsequentes antes de ser transmitida. Na trama UDP, o seu cabeçalho ocupa 8 bytes, que incluem a informação da porta de origem e de destino (2 bytes para cada), o tamanho dos dados e o campo de *checksum*. O campo de dados no máximo tem capacidade de armazenar 1472 bytes.

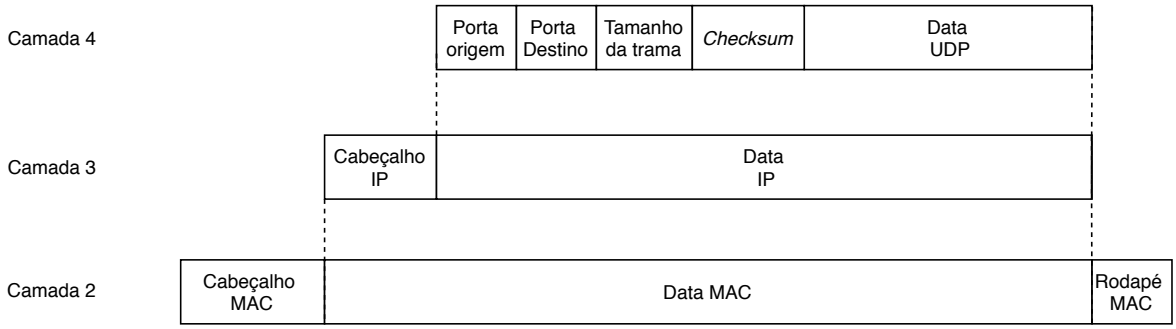


Figura 2.8: Trama UDP.

2.3.4 TCP

O protocolo TCP, tal como o protocolo UDP, encontra-se encapsulado no campo data do modelo IP e pertence à camada de transporte do modelo OSI. Este protocolo tem garantia de entrega do pacote enviado, ficando à espera da receção de um *Acknowledge* (ACK), o que implica que a transferência do pacote seja mais demorada. O recibo de entrega é verificado usando um contador temporal, que começa a contar até chegar a

mensagem de ACK. Caso o contador temporal chegue ao limite e verifique que não recebeu o recibo de entrega, volta a enviar o pacote.

Posição	**Função**
0	Porta origem
2	Porta destino
4	Número de sequência
8	ACK
12	Flags
14	*Window size*
16	*Checksum*
18	*Urgent pointer*
20	Opcões
40	Data

Tabela 2.1: Estrutura da trama TCP

Na tabela 2.1 é apresentada a estrutura da trama sendo a posição o índice do *byte* na trama e a respetiva função. A trama TCP é similar em tamanho à trama UDP, apenas mudando a informação existente no cabeçalho, sendo que o primeiro pode ter entre 20 a 60 *bytes*, sempre alinhados em múltiplos de quatro *bytes*. Desta forma, existem menos bytes disponíveis para dados. No cabeçalho do TCP está presente a informação da porta de destino e de origem, o identificador de pacote, um número de ACK, o tamanho da trama, entre outros [14].

2.3.5 CAN

O protocolo CAN é um protocolo desenvolvido pela Bosch, que se destina à comunicação entre dispositivos, para ambientes industriais e aplicações de tempo real. O sistema CAN é baseado no envio de mensagens em *broadcast*, ou seja, quando um controlador envia uma mensagem para a rede, esta será recebida por todos os nós da rede. Posteriormente, cada microcontrolador lê o identificador da mensagem e verifica se é do seu interesse ou não. Quando um controlador recebe uma mensagem, envia o *acknowledge* informando o nó emissor da receção da mesma. Caso não seja recebida, após um certo tempo, o nó repete o envio [15]. Ao nível físico, a rede CAN, é constituída por um par diferencial, ao qual se encontram ligados em paralelo todos os nós da rede. Este tipo de rede é tipicamente terminado nas extremidades com uma carga passiva, mantendo assim a impedância da linha. A velocidade de transferência de dados varia entre 1kbps e 1Mbps. Esta variação tem implicações na distância máxima do barramento, ou seja, com o aumento da velocidade, o tamanho da linha diminuí. Um exemplo da arquitetura de uma rede CAN pode ser observado na figura 2.9.

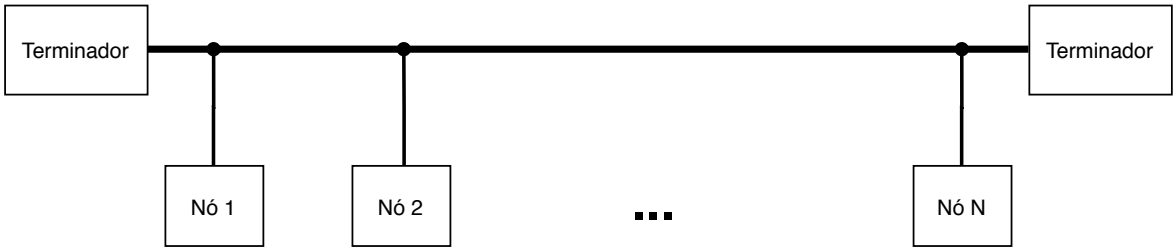


Figura 2.9: Arquitetura da rede CAN.

O início da trama é sinalizado por uma alteração na tensão diferencial na linha, o que indica o começo de uma nova transmissão. O Identificador (ID) é usado para identificar o tipo de mensagem e a sua prioridade, consoante a aplicação em causa. Na versão 2.0B em relação à 2.0A do CAN aumenta o número máximo de ID diferentes possíveis num barramento. Caso o CAN seja do tipo 2.0A tem 11 bits de identificação e se for do tipo 2.0B, o identificador tem 29 bits.

O CAN tem a capacidade de transportar até 8 bytes de dados por cada trama envidada, sendo este tamanho definido no campo do tamanho de dados. O CRC e o ACK são usados para deteção de erros na transmissão, por parte do recetor, e envio do *acknowledge* de receção, respetivamente.

Antes de enviar uma nova mensagem, o controlador CAN verifica a existência de comunicação no barramento. Caso haja atividade na linha, espera até poder tentar o envio.

2.4 Conversor série

Um dos modos de comunicação entre os microcontroladores e o computador é por porta RS-232/RS-485/RS-422, utilizando um protocolo série. Atualmente, os computadores já não vêm equipados com uma porta deste tipo, o que leva a ser necessário virtualizar uma. Para isso, no mercado existem alguns conversores, sendo um dos mais comuns o da marca FTDI, em que converte de *Universal Asynchronous Receiver/Transmitter* (UART) para USB. Um dos problemas existentes é a velocidade máxima de 12Mbps [16].

2.5 Conversor CAN para Ethernet

As soluções de comunicação industrial tendem atualmente a migrar para a adoção de protocolos com base em Ethernet. Segundo um estudo da HMS-Networks, referente ao biénio 2018/2019, a tendência de utilização dos protocolos com base em Ethernet tem vindo a aumentar, com um crescimento de 20% ao ano, figura 2.10. Os protocolos com base em barramento próprio, tendem a ter uma diminuição nos próximos anos.

Isso gera no mercado a necessidade de criar soluções que interliguem protocolos de barramento próprio com os de base em Ethernet, a partir de *gateways*.

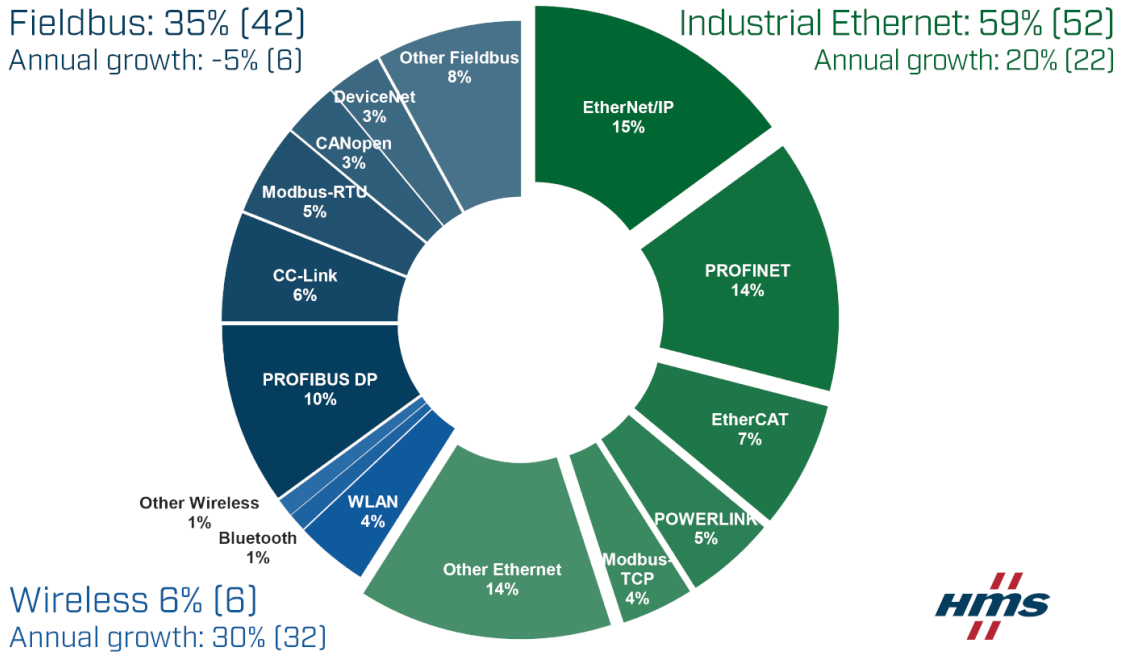


Figura 2.10: Estudo do marcado sobre a tendência em 2019 do uso de protocolos industriais. Fonte: HMS-Networks [1].

No mercado existem algumas marcas com estes conversores, como por exemplo a AxioMatic, VScom, proconX. Estes conversores têm a possibilidade de comunicarem por UDP ou TCP/IP, sendo configurados por uma página *web*. O seu modo de configuração obedece a certas normas definidas pelo fabricante. Por exemplo, a *gateway* da VScom ao converter a informação vinda na rede CAN está feita para só aceitar caracteres em codificação ASCII [17]. Já a proconX só tem a possibilidade de filtrar até 8 ID na parte de receção [18]. Embora estes conversores tenham a possibilidade de adaptar a comunicação entre uma rede Ethernet e um barramento CAN, apresentam limitações a nível da configuração que limitam a sua utilização no domínio da aplicação apresentada nesta dissertação.

CAPÍTULO 3

Arquitetura do Sistema

O aumento da velocidade de deslocação dos robôs, a maior capacidade de drible e táticas com mudanças repentinas da posição do robô levam a que a robustez das ligações mecânico/elétricas tenham de ser um fator a ter em conta, bem como a velocidade de transferência de informação, que difere substancialmente com esta mudança de paradigma. Este trabalho centra-se numa tentativa de solucionar estes novos desafios tecnológicos, com o desenvolvimento de uma nova *gateway*. A alteração para Ethernet, na nova *gateway*, herda todo o conhecimento adquirido da arquitetura anterior, sendo este o ponto de partida para o desenvolvimento da *gateway* Ethernet. Ao longo deste capítulo serão apresentados o funcionamento e a arquitetura da antiga *gateway* USB, bem como os requisitos mínimos e a arquitetura prevista para a nova *gateway* Ethernet.

3.1 *Gateway* USB

Na secção 2.1 foi apresentada uma descrição geral da construção de um robô CAMBADA, desde o nível mecânico até à estrutura do *software*. Nesta secção será explicada a comunicação entre a UCD e o MBN. Esta comunicação é possível devido à existência de uma *gateway*, cuja função é exclusivamente a conversão protocolar de USB para mensagens suportadas na rede CAN e vice-versa. A sua construção desde o *hardware* até ao *software* foi, à data, baseada num modelo que visou a sua otimização por forma a tornar a comunicação o mais rápida possível. A comunicação baseia-se num conversor USB/série, da marca FTDI. Este conversor envia e recebe o sinal diferencial, vindo no barramento USB, convertendo-o, posteriormente, para RS-232. Quando a informação chega ao microcontrolador, este verifica a existência de erros e reenvia-a para a rede CAN.

3.1.1 Arquitetura do *hardware* da *Gateway*

A arquitetura da *gateway* USB, representada na figura 3.1, é constituída por um porto CAN, um porto USB e uma zona de *Debug*, baseada em dispositivos Díodo Emissor de Luz (LED). A função de cada bloco será descrita seguidamente:

- **LED *debug*:** É constituído por 3 LEDs vermelhos, que servem para sinalizar mensagens do CAN, mensagens recebidas do USB e estado de funcionamento. Os LEDs da comunicação CAN e USB apresentam dois modos de funcionamento distintos, em que um modo indica a não existência de mensagem (piscando a uma frequência de 2Hz) e o outro modo indica a existência de uma nova mensagem no último segundo (piscando a uma frequência de 0.4Hz).
- **CAN:** Constituído por um conversor da *Maxim integrated*, MAX5031, uma ficha para ligação ao barramento e o terminador do barramento, caso este esteja num dos extremos da linha CAN.
- **USB:** A zona de USB é constituída por um conversor FT232R e uma porta USB-B.

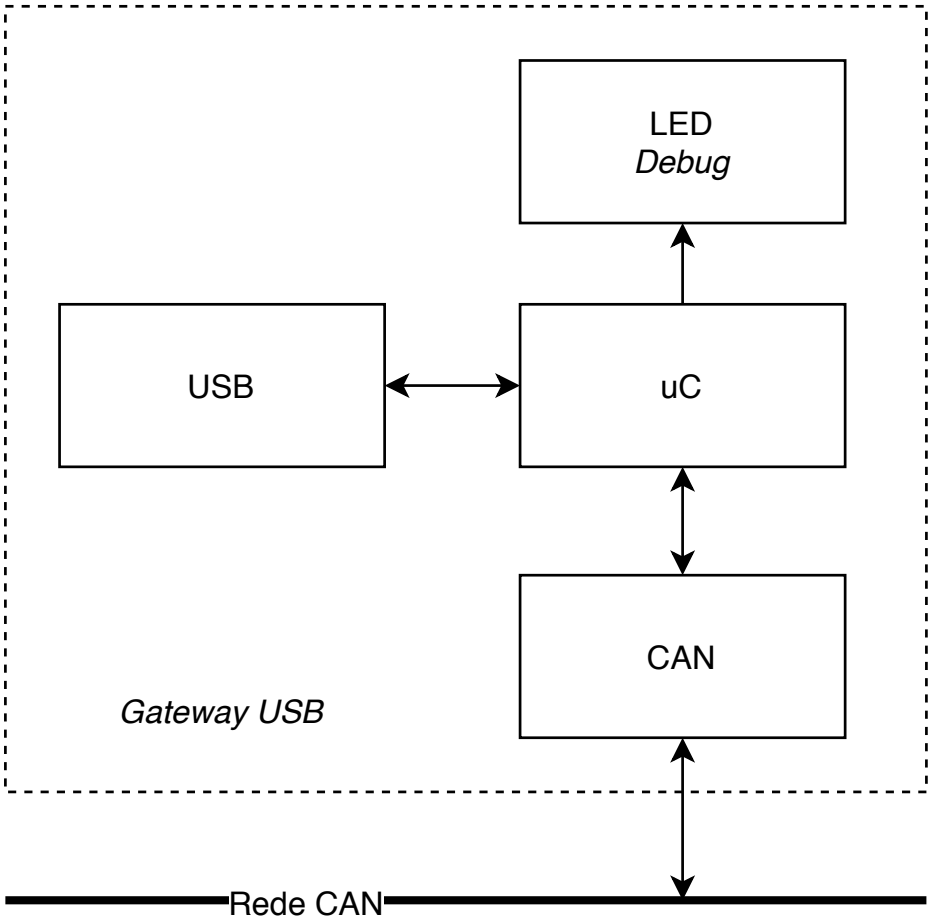


Figura 3.1: Arquitetura da Gateway USB do robô CAMBADA.

3.1.2 Arquitetura de *software*

O *software* que corre na UCD, HWcomm, é responsável pela comunicação entre o MBN e a UCD e é um processo independente constituído por duas *threads*. Estas servem para que a receção e a transmissão possam decorrer independentemente uma da outra, sendo no entanto geridas pelo mesmo processo. A figura 3.2 mostra quais as funcionalidades de cada *thread* [19]. No início do processo HWcomm são configuradas as *threads* e a comunicação USB. Depois da configuração, as *threads* são ativadas até que recebam ordens para desligar.

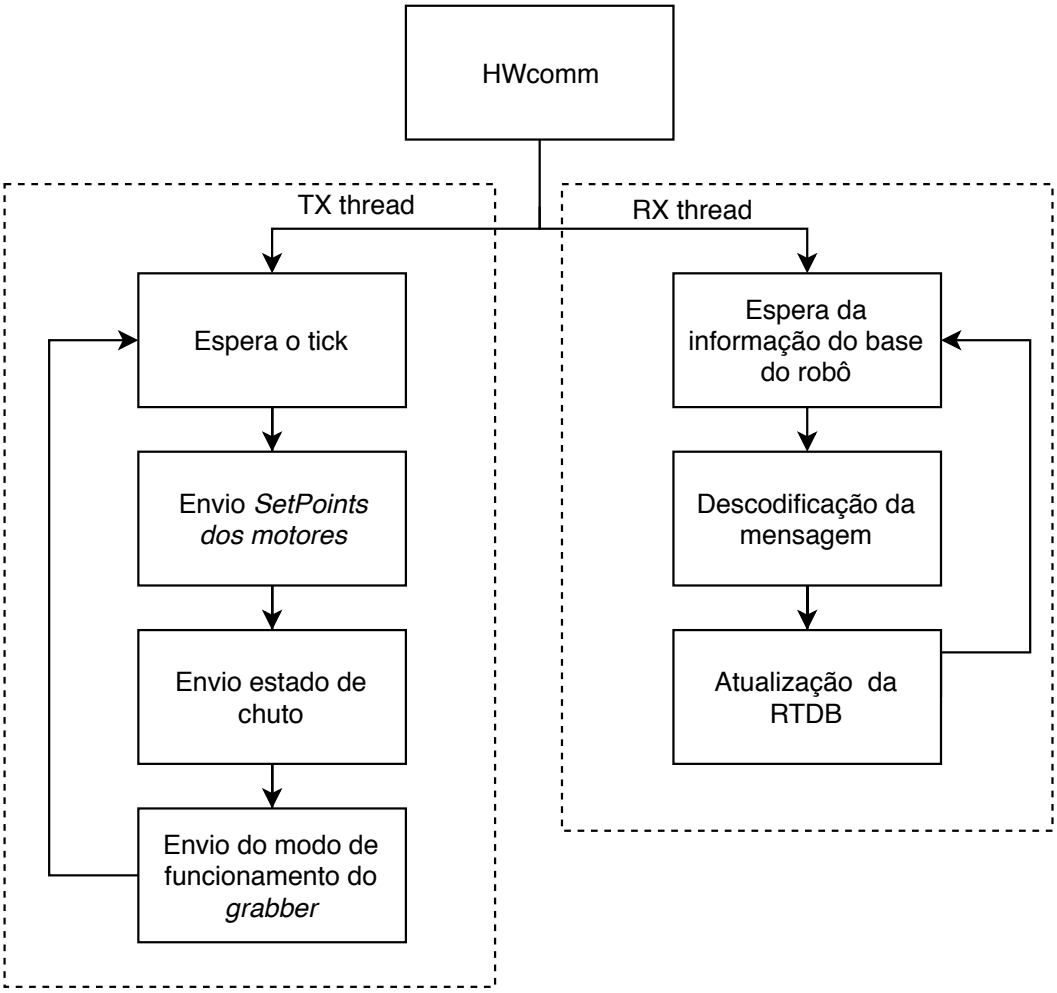


Figura 3.2: Arquitetura do *software* do processo HWcomm.

Na transmissão, em cada ciclo, aguarda-se pelo sinal de sincronização, conforme apresentado na secção 2.1. Após o sinal, quando começa um novo ciclo de transmissão de informação, é necessário ir buscar à RTDB a informação dos *set points* dos motores. Com esta informação é criada uma trama que é posteriormente enviada para a porta USB. Este processo acontece também para o estado chuto e o funcionamento dos

grabbers. Após o envio de todas as mensagens a *thread* fica em espera até ser iniciado um novo ciclo.

A receção do lado da UCD trabalha assincronamente, encontrando-se sempre à escuta por nova informação. Cada vez que chega uma nova mensagem do MBN, essa informação passa por um processo de descodificação e validação. Caso a mensagem seja válida os dados dai extraídos são atualizados na RTDB, para serem usados por outros processos no ciclo seguinte. É de referir que a mensagem pode conter novas informações sobre o *Inertial Measurement Unit* (IMU), a posição do *grabber*, os valores dos *encoders* dos motores ou a tensão do condensador do sistema de chuto.

3.1.3 Estrutura de *firmware*

O *firmware* que está a correr num microcontrolador da *gateway* (um PIC32 da Microchip®), é responsável por receber os caracteres vindos da porta série e convertê-los numa mensagem CAN, de acordo com o que é visível na figura 3.3 [20]. No arranque do *firmware* são configurados os *timers*, as interrupções, a UART e o CAN.

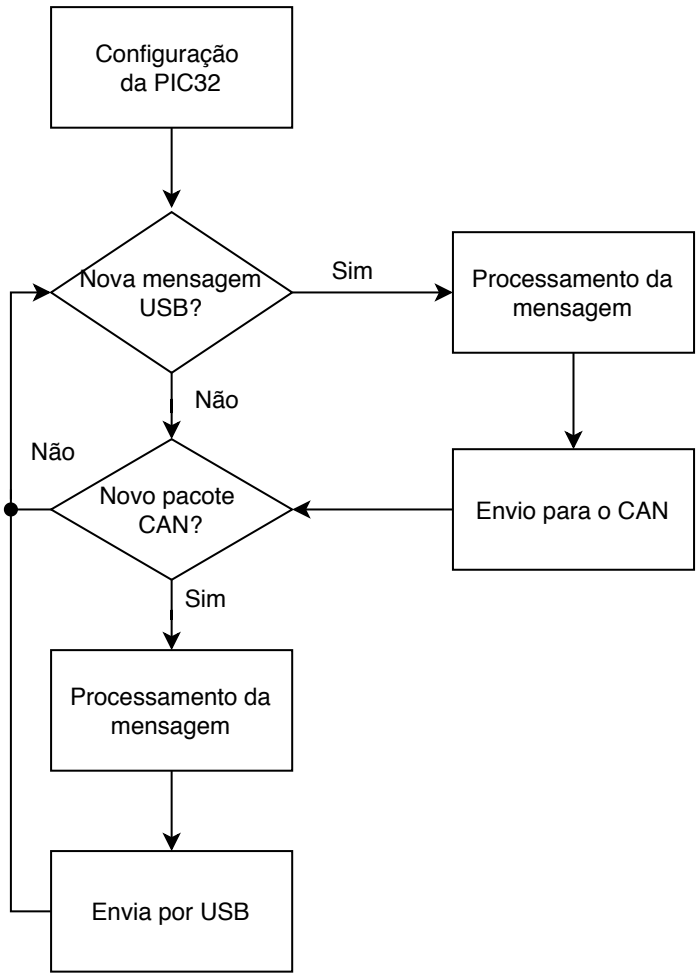


Figura 3.3: Arquitetura do *firmware* da *Gateway* USB.

No fim da configuração o sistema começa a execução do seu ciclo principal. No início de um ciclo começa-se por verificar a existência de uma nova mensagem vinda do USB. Caso exista uma nova mensagem, esta será validada e enviada para a rede CAN. Posteriormente, ou caso não exista uma nova mensagem vinda do USB, é verificada a existência de uma mensagem na rede CAN. Quando esta mensagem existe, o *firmware* procede à sua compactação numa mensagem destinada a ser enviada através da porta série. Esta será convertida para USB pelo modulo FTDI.

A estrutura da mensagem que o microcontrolador recebe por USB encontra-se representada na figura 3.4. A mensagem recebida inclui um caracter de início, Início do texto (STX) e a partir deste caracter, o microcontrolador começa a guardar, em *buffer*, o remanescente da mensagem até receber o caracter de fim de mensagem, Fim do texto (ETX). Para a mensagem em *buffer* ser validada, verifica-se se o *checksum* recebido é igual ao *checksum* calculado. Caso seja diferente, então o sistema descarta a mensagem. Caso, seja válida, este cria a mensagem CAN e envia-a para a rede.

A transmissão de informação do MBN para a UCD é feita sempre que a *gateway* recebe uma nova mensagem vinda do CAN. Sempre que isto acontece, o microcontrolador cria uma nova mensagem de acordo o formado indicado da figura 3.4, e envia-a para a UCD. Este processo acontece sempre que uma placa da rede CAN envia uma nova mensagem.

STX	ID	DATA	CS	ETX
1 *Byte*	1 *Byte*	0-8 *Byte*	1 *Byte*	*1Byte*

Figura 3.4: Estrutura da mensagem usada na comunicação entre a UCD e a *gateway* USB.

3.2 Requisitos do projeto

A construção da nova *gateway* serve para colmatar falhas e necessidades existentes na anterior, sem que sejam necessárias grandes alterações no robô. Assim, os requisitos necessários no desenvolvimento desta nova *gateway* passam, não só por especificações de *hardware* e *software*/*firmware*, como por especificações a nível mecânico dos próprios robôs.

A nível mecânico, é necessário que a placa mantenha as mesmas dimensões da placa anterior e que as suas ligações estejam em zonas acessíveis, promovendo uma fácil manutenção do robô. A escolha dos componentes deve manter a marca de microcontroladores já existentes nas restantes placas - Microchip®. O *software* e *firmware* devem manter a estrutura e serem criadas funções em que possa ser escolhida a *gateway* Ethernet ou USB.

A nível de comunicações com a *gateway* é necessário garantir a receção de mensagens a cada 20 ms vindas da UCD e enviá-las para a rede CAN. Ao mesmo tempo, a *gateway* tem de assegurar o envio para a UCD das mensagens recebidas da rede CAN. Por outro lado, a nova *gateway* tem de conseguir coabitar numa rede Ethernet que, no pior dos casos, terá que suportar tráfego com origem na câmara de vídeo do sistema de visão e dados sobre a posição da bola gerados por um Unidade de Processamento Gráfico (GPU) NVIDIA Jetson. A transmissão de toda esta informação, incluindo a da *gateway* poderá ocorrer simultaneamente, sendo necessário assegurar que não existe perda de pacotes.

A nova *gateway* deve poder ter pelo menos duas entradas para botões de pressão, sendo que um deve ser configurado a partir do alto nível e o outro ser responsável pelo envio de um pacote *wake on lan*, por forma a que seja possível ligar a UCD a partir da *gateway*.

Deste modo, esta arquitetura herda todo o conceito já adquirido anteriormente, sobre a forma como a placa comunica com o alto nível. Para além disto, deseja-se que esta alteração de paradigma traga uma maior largura de banda e mais robustez nas ligações.

3.3 Arquitetura proposta

Da análise de requisitos e do conhecimento já adquirido com a *gateway* anteriormente utilizada, foi estruturada uma estratégia de desenvolvimento, dividida em 4 partes. O estudo de cada uma destas partes (descritas abaixo) será fundamental para a viabilidade da implementação desta nova *gateway*.

- **Validação:** A primeira fase serviu para analisar a viabilidade da nova *gateway* numa placa de demonstração. Com isto, foi possível verificar a possibilidade de continuar a usar a mesma família de microcontroladores da Microchip®. Esta possibilidade resulta na prática, num benefício por permitir a reutilização de alguns módulos da versão anterior;
- **Adaptação de *firmware*:** A segunda fase serviu para o desenvolvimento do *firmware*. Tal como no modelo anterior este tem como objetivo a receção da informação vinda da rede, recorrendo ao protocolo UDP e reencaminhando-a para a rede CAN. Complementarmente o *firmware* integrou também o suporte à interação com os botões de pressão;
- **Adaptação de *software*:** Nesta fase visou a adaptação do software que é executado na UCD. O principal objetivo consistiu em adicionar ao código existente no HWcomm a possibilidade de enviar informação para o MBN, assegurando simultaneamente que este envio pode ser feito tanto por USB como por Ethernet

(neste caso específico por UDP). A escolha do protocolo a utilizar é feita aquando do lançamento do processo HWcomm;

- **Desenho de *hardware*:** Por último, e após toda a validação e adaptação a nível do *software* e *firmware*, procedeu-se ao desenvolvimento uma Placa de Circuito Impresso (PCB) conforme as especificações do robô.

Após a parte de desenvolvimento, produção e assemblagem de um protótipo foi finalmente necessário validar todo o sistema. Esta validação foi feita recorrendo a uma bateria de testes por forma a certificar que os respetivos resultados cumprem os requisitos para assegurar que a nova *gateway* pode ser montada nos robôs CAMBADA. Com a realização destes testes pretendeu-se verificar qual o tempo mínimo de comunicação entre sistemas, se há perda de pacotes em ambiente de funcionamento normal do robô e qual o consumo energético deste novo sistema.

3.3.1 Arquitetura do *software*

O novo *software*, utilizado no robô ao nível da UCD, deve coabitar com a estrutura já existente de código, podendo o utilizador escolher, ao lançar o processo, qual o tipo de comunicação que irá ser executada.

O HWcomm, ao iniciar, verifica qual o modo de comunicação que executará, a partir de um argumento que lhe é passado no seu lançamento. Ao iniciar o processo, o modo de comunicação é configurado sendo posteriormente lançadas as respetivas *threads*.

Uma vantagem da Ethernet é a capacidade de transportar um maior volume de dados numa só mensagem. Por isso, antes do envio da informação, é construída uma nova estrutura onde são agrupadas, numa só mensagem, todas as tramas. Esta mensagem é por sua vez enviada para o MBN no final de cada ciclo. Assim, no início de cada ciclo, começa a ser criada uma mensagem com a estrutura que pode ser observada na figura 3.5. Esta mensagem inclui as tramas originalmente enviadas por USB, ou seja, informação para o IMU, velocidade dos *grabbers*, *setpoints* de velocidade dos motores e a ordem de chuto (ativa ou não ativa).

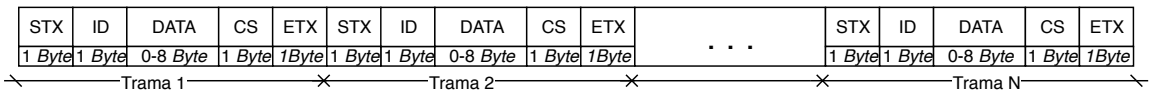


Figura 3.5: Nova mensagem, com a concatenação de todas as tramas.

Já a receção funciona de maneira diferente. Pelo facto de as mensagens serem produzidas, por cada um dos sub-módulos do MBN, de forma assíncrona, será mantida a mesma estrutura da versão anterior da *gateway*. A mensagem que é recebida virá com a mesma estrutura da figura 3.4, sendo a única diferença o protocolo de comunicação usado.

3.3.2 Arquitetura do *firmware*

Ao nível do *firmware* foi necessário efetuar duas alterações. A primeira diz respeito à conexão do PIC32 com a interface de Ethernet, enquanto a segunda consistiu na adaptação da *gateway*, por forma a que esta estivesse em conformidade com as alterações feitas no *software*, secção 3.3.1.

O PIC32 disponibiliza já, internamente, um controlador Ethernet, sendo apenas necessário proceder à sua configuração e respetivos portos. Para além disso também é preciso criar todas as camadas de rede referentes ao modelo OSI que se mostrem necessárias, sendo estas otimizadas para o microcontrolador.

O *firmware* funciona de acordo com o fluxograma da figura 3.6. No início de cada ciclo é verificada a existência de uma nova mensagem vinda por Ethernet. Cada mensagem é analisada na sua totalidade de modo a identificar o início e o fim de cada trama, a partir dos caracteres STX e ETX, respetivamente. No fim de cada mensagem, caso a mesma seja validada, é enviada para a rede CAN.

Após ser processada a mensagem vinda por Ethernet, é verificada a eventual existência de uma mensagem proveniente pela rede CAN. Caso exista, esta mensagem deve ser compactada e enviada por Ethernet para a UCD.

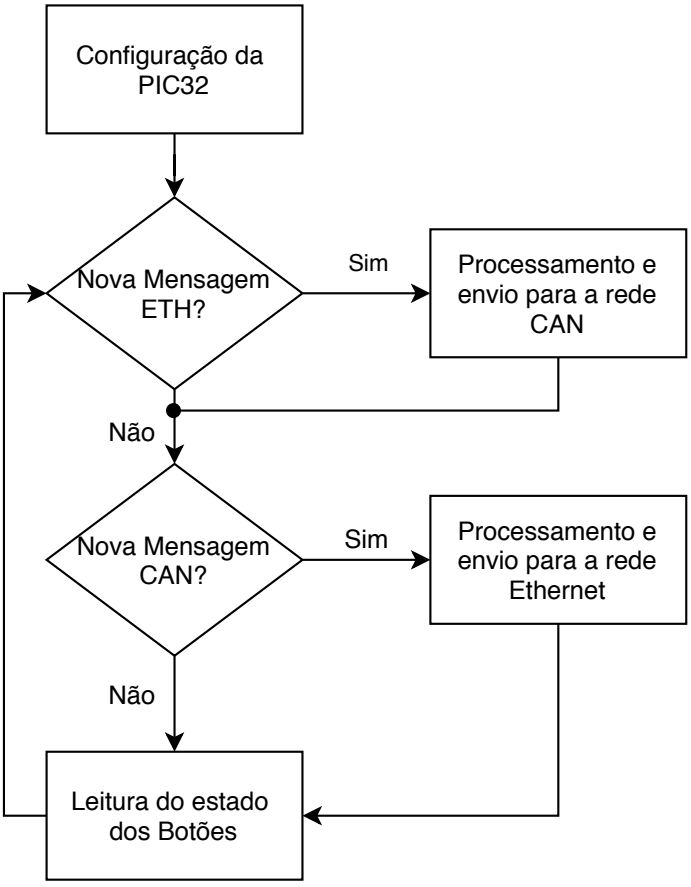


Figura 3.6: Estrutura do novo de *firmware*.

Uma vez que os requisitos identificados, previam a adição de pelo menos dois botões de pressão na *gateway* foi necessário ao nível *firmware* configurar as respetivas funcionalidades. Um dos botões serve para enviar o *magicpacket* do *wake on lan* para ligar o computador do robô (UCD). O segundo botão tem uma finalidade mais genérica, sendo utilizado para enviar para a UCD informação de quando foi pressionado. A responsabilidade sobre a utilização desta informação caberá neste caso ao HWcomm de acordo com as necessidades.

3.3.3 *Hardware*

Da análise dos requisitos e das caraterísticas da *gateway* USB, apresentados na secção 3.2 e 3.1, respetivamente, foi elaborado um diagrama de blocos para uma primeira representação da arquitetura da nova *gateway*, como demonstra a figura 3.7.

O elemento central da placa é então um microcontrolador da Microchip® da família PIC32. Esta opção tem a vantagem suplementar, para além das que foram referidas, de constituírem a única família da Microchip® com possibilidade de suportar um controlador Ethernet e um controlador CAN integrada num só microcontrolador [21]. Embora estes controladores se encontrem integrados no PIC32, é necessário para implementar a totalidade da camada física e adicionar os respetivos *transceivers*, quer para o Ethernet, quer para o CAN. Finalmente são ainda mantidos quer os LED, quer o USB, neste caso exclusivamente para efeito de *debug* e adicionado um porto onde serão ligados os botões.

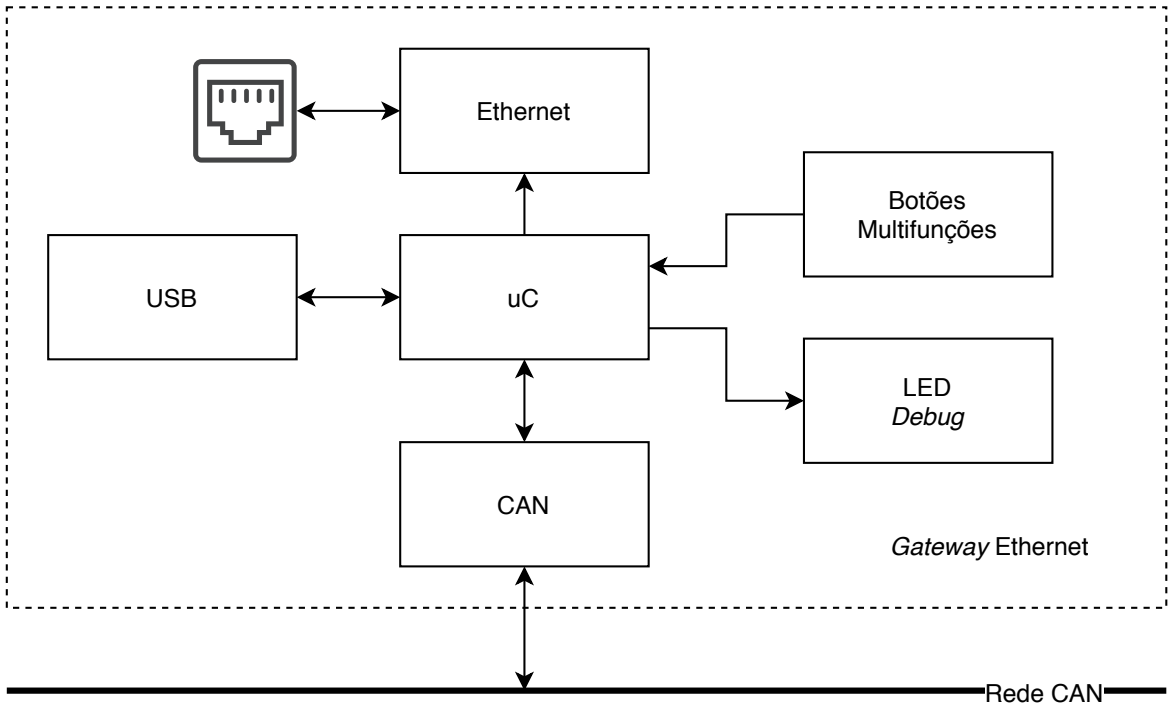


Figura 3.7: Nova arquitetura da *gateway* Ethernet.

A fase final consistiu então no projeto de uma PCB que respeitasse o mais possível as dimensões e o layout da anterior *gateway*. Estes requisitos visam simplificar a integração da nova *gateway* na estrutura do robô, permitindo uma substituição direta da anterior *gateway* pela nova versão.

CAPÍTULO 4

Implementação

No capítulo 3 foi exposta a atual arquitetura existente no CAMBADA e a problemática na comunicação entre a eletrónica existente no MBN e a UCD. Para além disso, foi apresentada uma proposta de desenvolvimento da nova *gateway* com o objetivo de mudar o panorama da comunicação, mas mantendo toda a estrutura entre a UCD e o MBN.

O capítulo atual, servirá exclusivamente para apresentar o desenvolvimento feito na nova *gateway* e as adaptações na arquitetura já existente nos robôs CAMBADA, a nível de *software* e de *firmware*, bem como todos os procedimentos seguidos para desenhar, desenvolver e testar a nova placa. Para isso, é necessário ter em conta os requisitos existentes, ou seja, a nova *gateway* deve manter as funcionalidades da anterior mudando apenas o modo de comunicação e adicionando novas funcionalidades.

A abordagem ao longo do capítulo começará por uma avaliação do microcontrolador e a justificação do motivo da utilização do mesmo. Após isso, serão explicados os passos feitos na alteração do *software*, *firmware* e desenho da nova PCB para a *gateway*. No desenrolar do capítulo serão detalhadamente justificados os procedimento seguidos.

4.1 Validação do hardware proposto

A validação a nível de *hardware* é um ponto essencial no processo de desenvolvimento da *gateway* com base em microcontroladores. O primeiro passo da validação recorreu ao uso de uma placa de desenvolvimento da Microchip®, na qual é já disponibilizada a interface Ethernet ao nível físico. Para além disso, esta placa também apresentava a vantagem de ter alguns pinos do microcontrolador disponíveis, o que permitiu desde logo o desenvolvimento dos módulos complementares da *gateway* como sejam, a interface CAN ou uma interface UART para efeitos de *debug*.

A placa de desenvolvimento é uma *"Microchip Ethernet Starter Kit"*, devido ao facto de ela ser composta por um PIC32MX, uma interface de rede e um módulo de programação [22]. Para além disto, e fazendo uso dos pinos de expansão disponíveis na placa de desenvolvimento, foi adicionada uma outra placa construída com um módulo CAN e uma interface UART para *debug* durante o desenvolvimento. A figura 4.1 apresenta a placa utilizada com as seguintes caraterísticas:

Módulo Ethernet: *Transceiver* Ethernet da *Texas Instruments* do modelo DP83848. Responsável pela camada física da rede Ethernet;

Módulo I/O: Porto de interface com LED e botões. Esta interface ajudou no desenvolvimento, permitindo selecionar vários modos de *debug*;

Módulo CAN: Com a colocação em funcionamento do módulo Ethernet foi possível começar a executar testes com uma rede idêntica à existente num robô. Para esse efeito foi naturalmente necessário integrar neste módulo de desenvolvimento uma interface CAN;

Módulo UART: Porto UART para *debug* em desenvolvimento. Este módulo serviu exclusivamente durante o processo de desenvolvimento, por isso não foi integrado na placa final.

Microcontrolador: Microcontrolador Microchip®, modelo PIC32MX795F512L, responsável pelo processamento e reencaminhamento das mensagens.

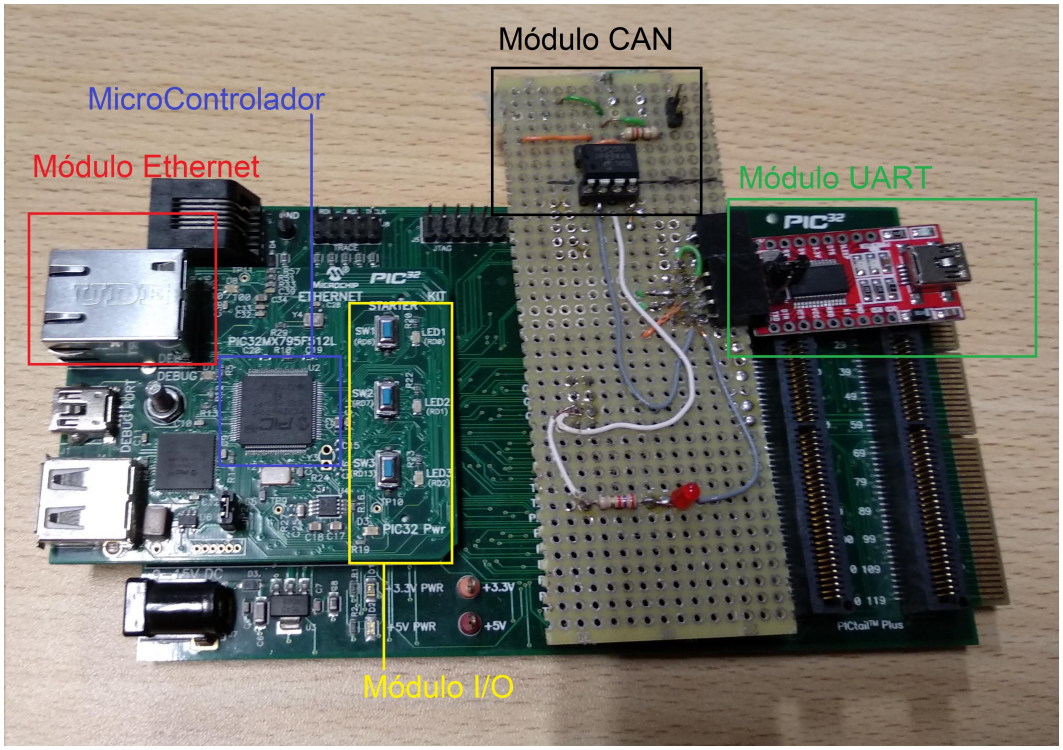


Figura 4.1: Placa de desenvolvimento inicial.

O uso desta placa teve dois objetivos. O primeiro foi validar a utilização do módulo de Ethernet existente e a interação com o PIC32. O segundo foi permitir que, durante o tempo de produção da placa e assemblagem da mesma, fosse possível a continuação do trabalho de desenvolvimento do *firmware*, *software* e respetivos testes preliminares.

4.2 *Firmware*

O microcontrolador é responsável, como foi já referido, pela tradução entre protocolos, devendo funcionar de forma eficiente para rapidamente processar as mensagens que lhe chegam. Para isso, a estrutura do código, para a nova *gateway*, teve como base a versão anterior, mudando o modo de comunicação e o processamento das mensagens recebidas da UCD.

A figura 4.2 mostra o diagrama de fluxo do funcionamento da nova *gateway*. Após um *power up reset* a *gateway* começa por configurar os portos de saída e de entrada de todos os pinos usados, bem como o porto CAN, o porto Ethernet e as interrupções de sistema. Após esta configuração inicial, e para assegurar o normal funcionamento do PIC32, é igualmente necessário configurar a rede à qual a placa irá ser ligada, ou seja, o MAC, o IP, a máscara de rede e o *socket* usado para comunicação UDP.

Ao utilizar UDP, o PIC32 tem a possibilidade de ser configurada para abrir até oito *sockets* ao mesmo tempo. Em condições de funcionamento normal no robô são utilizados apenas 2 *sockets*, respetivamente para o envio e receção de mensagens para e do UCD, com informações comportamentais do robô. Os restantes *sockets* são por agora reservados para futuras *upgrades* e aplicações.

No início de cada ciclo é verificada a existência de uma nova mensagem UDP. A existência de uma nova mensagem começa um ciclo que irá percorrer todos os caracteres para descompactar as tramas que serão enviadas pela rede CAN. Conforme representado na figura 3.5, cada nova trama contida na mensagem inicia-se com o carácter STX e acaba com o carácter ETX. A informação existente entre estes caracteres é guardada para posteriormente ser retirado o identificador da trama, os dados da trama e o *checksum*, o qual é calculado e verificado se é igual ao valor vindo na trama. É importante referir que os dados da trama estão codificados em binário. Logo é perfeitamente possível que entre os dados recebidos possam existir um ou mais bytes que, em *American Standard Code for Information Interchange* (ASCII), correspondam aos códigos de STX, ETX e *Data Link Escape* (DLE). Quando tal acontece, o caráter em causa é precedido de um código DLE por forma a que o software que está a extrair os dados da trama seja informado de que o byte subsequente é um byte de dados e não um carácter de controlo. Quando se chega ao fim de uma nova trama os dados recebidos são validados, após se proceder à construção de uma mensagem CAN que será em seguida enviada.

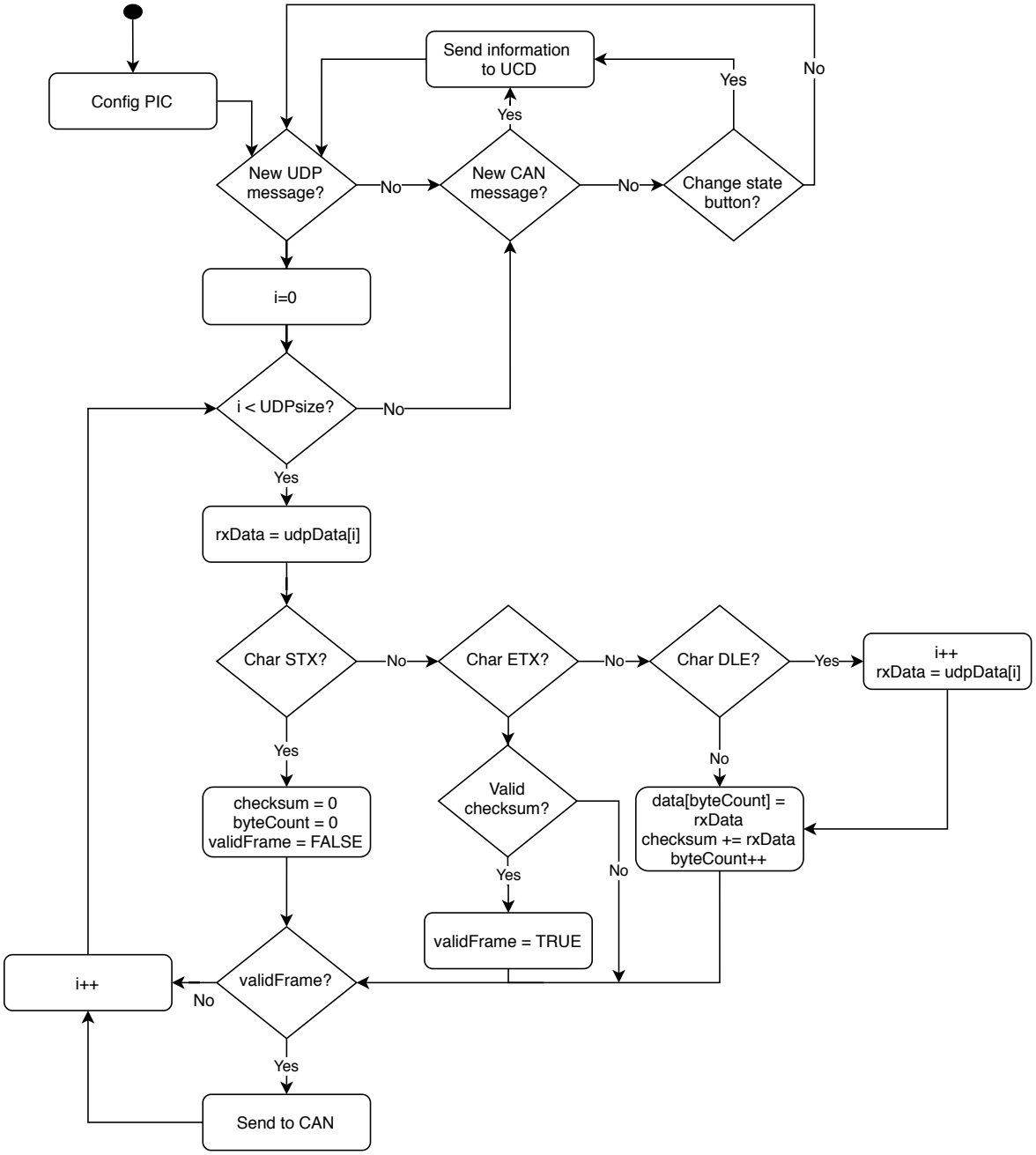


Figura 4.2: Arquitetura do *firmware*.

A mensagem recebida por UDP tem a capacidade de transportar várias tramas, conforme mostra a figura 3.5. Por isso, o processo descrito da conversão de UDP para CAN percorrerá a mensagem até ao último carácter recebido, separando cada nova trama com a informação para o MBN.

Terminada a receção e processamento de uma mensagem vinda da rede Ethernet, procede-se então à verificação da existência de uma mensagem na rede CAN para ser enviada para a UCD. No caso de não haver, verifica-se se houve mudança do estado dos botões e, em caso afirmativo essa informação é envida para a UCD. No caso de existir

uma mensagem na rede CAN, esta é compactada numa trama, conforme a figura 3.4, e enviada para a UCD. Terminado este procedimento recomeça-se a partir do ponto inicial verificando se existe uma nova mensagem UDP.

A figura 4.2, mostra o fluxograma de funcionamento de todo o sistema a correr na nova *gateway*. A eficiência e rapidez, sem perda informação, é uma das mais valias que se pretende obter com a nova *gateway*, por forma a que esta possa ser efetivamente aplicada e usadas no robôs, ultrapassado as limitações existentes na anterior.

4.3 Software do HWcomm

A adaptação do *Software* executado na UCD de modo a suportar a nova *gateway* requer que este seja compatível com os dois modos de comunicação (USB e Ethernet (ETH)). Conforme descrito no capítulo 3, o *software* encontra-se separado em duas *threads*: uma responsável pela transmissão de dados da UCD para o MBN, de forma síncrona com os processos restantes do robô. A outra na receção de mensagens vindas do MBN, que como anteriormente referido, são recebidas de forma assíncrona.

A estrutura de funcionamento do processo HWcomm continua a assegurar a mesma funcionalidade, sendo agora apenas necessário adicionar a capacidade de comunicar por Ethernet. Esta alteração deve ter em conta o código original mantendo, tanto quanto possível, uma estrutura do código semelhante ao já existente. Para isso o módulo de comunicação por Ethernet utilizará a capacidade de herança existente no C++. Esta classe de comunicação inclui funções virtuais específicas para receção, envio, codificação e descodificação de mensagens. As classes filho são específicas para cada modo de comunicação (USB e ETH) e herdam as funções de uma classe pai de comunicação.

A utilização de herança a nível do *software* releva-se uma mais valia no processo principal do HWcomm. Com os métodos virtuais na classe principal é possível criar uma interface qualquer que seja o modo de comunicação. As classe que herdam as funcionalidades só tem de obedecer a estes métodos podendo desta forma funcionar de modo independente. A escolha da classe a utilizar no HWcomm é realizada no momento em que este é iniciado.

Para além do modo de comunicação, como já foi brevemente referido, existem também diferenças no interface da mensagem enviada para o MBN. Na comunicação por USB será enviada uma mensagem por cada informação retirada da RTDB. Por outro lado, a Ethernet como dispõe de maior largura de banda permite compactar numa só mensagem as várias tramas construídas a partir da informação retirada da RTDB. A receção irá continuar a funcionar da mesma forma da versão anterior, sendo só adicionada a capacidade de utilização do modo de comunicação por Ethernet. A figura 4.3 demonstra o novo funcionamento da HWcomm.

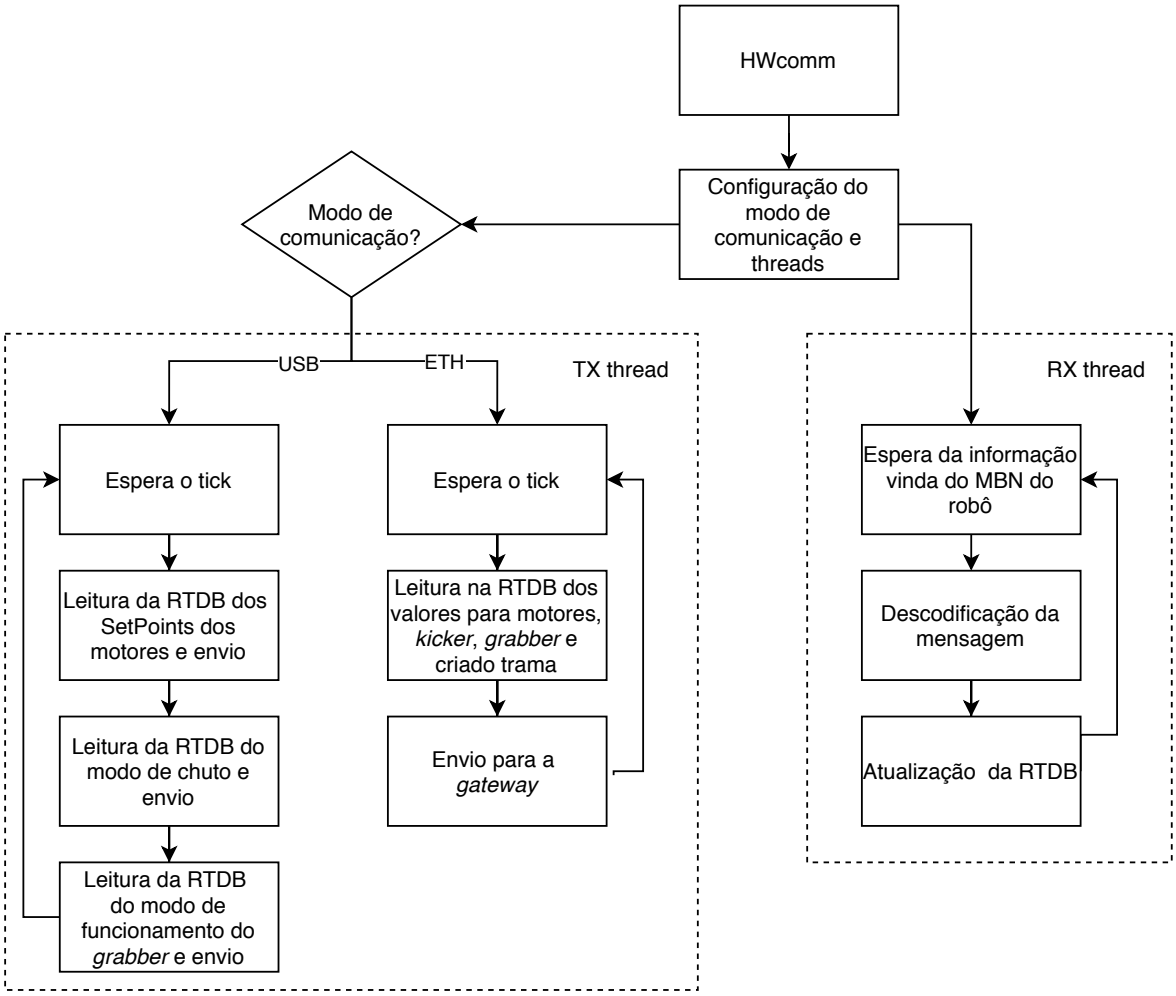


Figura 4.3: Arquitetura do software do HWcomm.

O benefício da utilização do mecanismo da herança é que o código da HWcomm fica transparente para qualquer modo de comunicação, uma vez que este utiliza sempre as mesmas funções de envio e receção. Por outro lado, o código terá maior capacidade de escalabilidade, caso seja futuramente usado outro modo de comunicação.

4.4 Estrutura do hardware

A placa de desenvolvimento utilizada na secção anterior, ajudou a perceber metodologias e processos a executar para o desenho da nova *gateway*. A continuação do desenvolvimento da mesma obriga a que seja desenhada uma placa, para substituir a atual existente no robô.

O desenho da nova arquitetura começou por levar em consideração a anterior *gateway* e a placa de desenvolvimento utilizada. Na figura 4.4 é possível observar a arquitetura concebida para a nova *gateway* com Ethernet.

A nova *gateway* será alimentada, tal como os restantes módulos do MBN, a partir

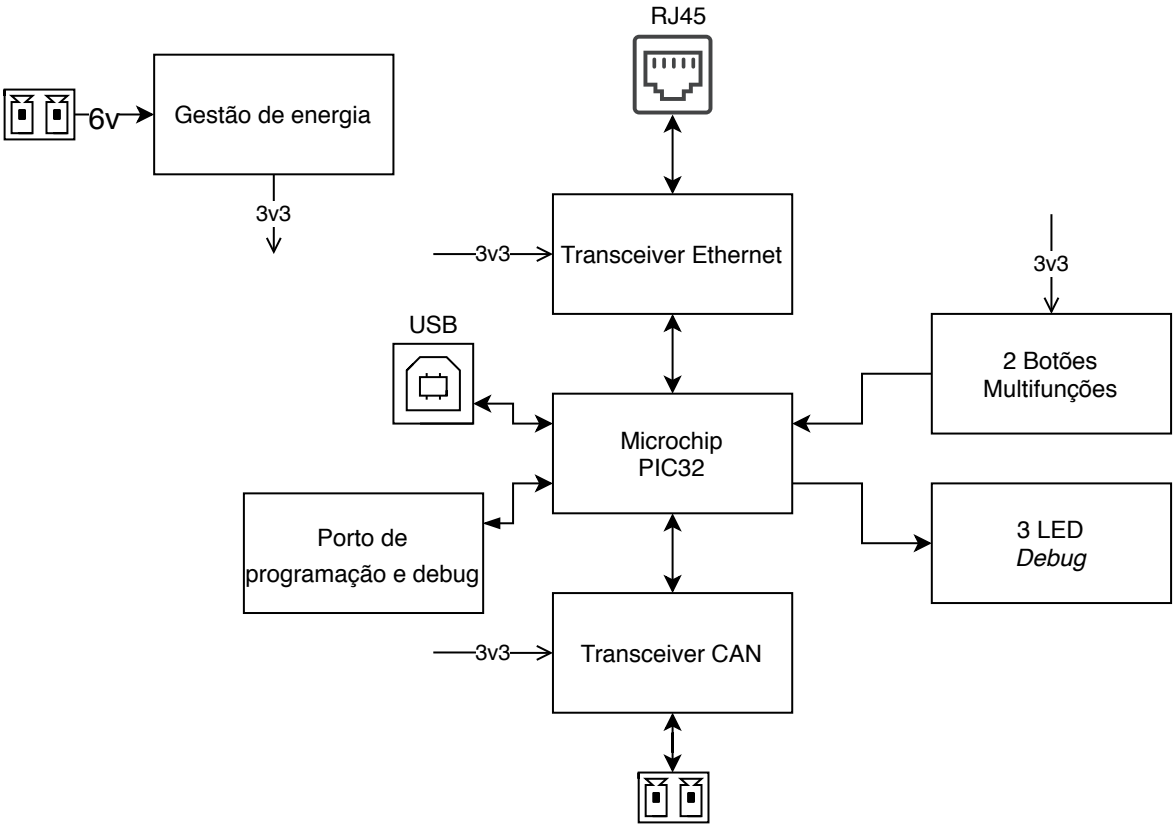


Figura 4.4: Arquitetura detalhada da nova *gateway*.

de uma tensão de 6V. Esta tensão é posteriormente reduzida para 3.3V por recurso a um regulador de tensão linear, e irá alimentar o microcontrolador e os *transceivers* CAN e Ethernet. O microcontrolador será um PIC32 que controlará um porto Ethernet, um porto CAN e um porto USB nativo. Para além disso, existirão um conjunto de LEDs e de botões de interface. É ainda disponibilizado uma ficha com os pinos para programação do PIC32, um pino I/O e uma UART para efeitos de *debug*. Em anexo mostra o esquemático completo. Nas subsecções apresentadas em seguida proceder-se-á à justificação de algumas destas opções.

4.4.1 Microcontrolador

Desde há vários anos a esta parte, os robôs CAMBADA usam microcontroladores da marca Microchip®. Este encontram-se espalhados pelos vários módulos do MBN, como sejam o do IMU, o do *kicker* e o dos controladores dos motores que são baseados num dsPIC. A placa do controladores do Grabber e na versão anterior da *gateway* utilizam PIC32. Consequentemente a adoção de um microcontrolador deste fabricante parece ser a opção lógica e consistente com a experiência existente.

Na nova versão foi necessário escolher um microcontrolador que disponibilize internamente pelo menos um controlador CAN e um controlador Ethernet [23] [24]. Através

da análise da tabela comparativa da Microchip® foi possível verificar que, para estes requisitos, existem disponíveis os modelos PIC32MX7 e o PIC32MZ,[21].

O modelo PIC32MZ é um microcontrolador com uma maior frequência de relógio e, consequentemente, um maior consumo de energia, quando comparado com o PIC32MX7. Este primeiro modelo também é, por sua vez, otimizado para processamento gráfico, o que não é necessário para esta aplicação [25]. O modelo PIC32MX7 é um microcontrolador com uma frequência de relógio mais baixa, mas para a aplicação em causa é o ideal, sendo o seu mais reduzido consumo uma mais-valia para uma aplicação dependente de baterias. Por outro lado, a placa de desenvolvimento utilizada nos primeiros passos tem incorporado também um PIC32MX795F512L.

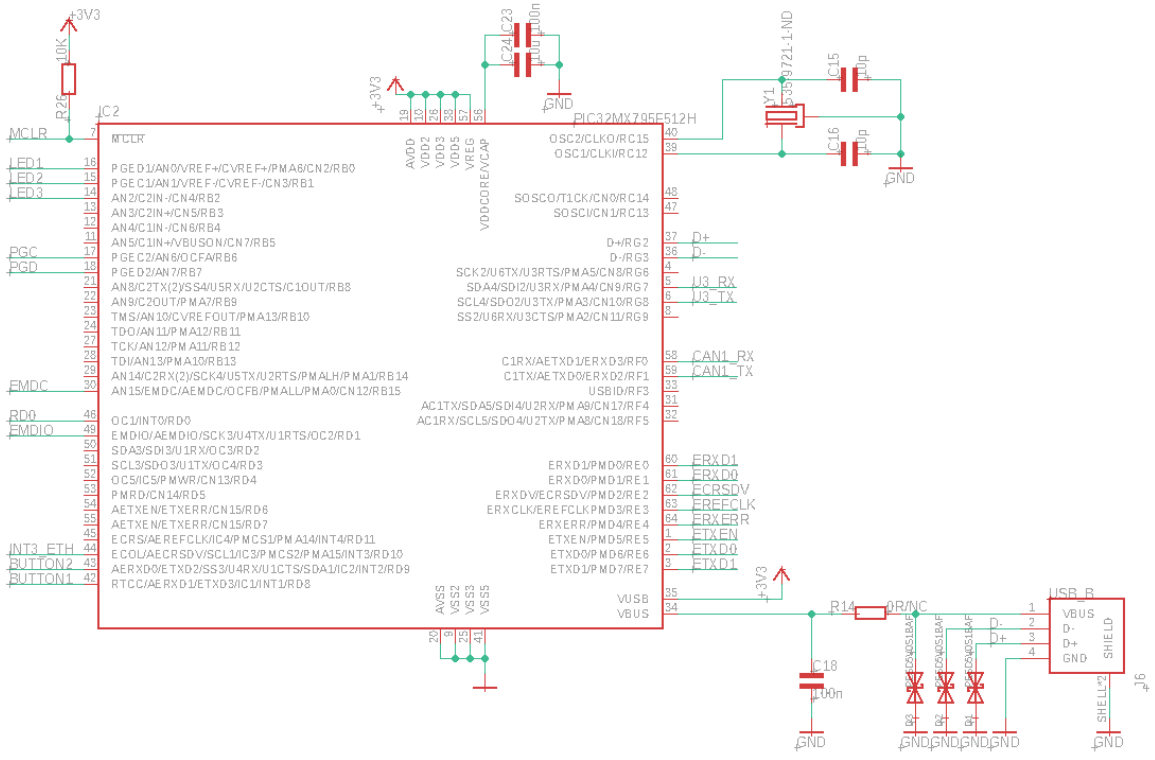


Figura 4.5: Saídas e entradas do PIC32MX795F512H.

Após a avaliação dos microcontroladores possíveis para a *gateway* e a avaliação de funcionamento da comunicação por Ethernet no PIC32, descrito na secção 4.1, verificou-se que o PIC32MX795F512L tem disponíveis um maior número de pinos do que o necessário. Para a nova *gateway* são necessários 2 pinos para o CAN, 10 pinos para Ethernet, 3 pinos para os LEDs, 2 pinos para botões, os pinos de programação e 3 pinos para USB. Tendo em conta os pinos necessários, o PIC32MX795F512H é a opção mais indicada. O custo dos dois modelos é semelhante, por isso não é um fator a ter em conta. Esta escolha justifica-se assim porque embora a versão PIC32MX795F512H disponibilize um menor número de pinos não há do ponto de vista da arquitetura interna

e do seu funcionamento qualquer outra diferença entre os dois modelos. A figura 4.5, mostra o modelo selecionado, bem como a disposição dos pinos que foram utilizados.

4.4.2 Ethernet

O PIC32 escolhido é dotada de um controlador Ethernet, mas o mesmo necessita de uma interface que interprete a camada 1 do modelo OSI. Essa interface é um circuito cuja função é codificar e descodificar os sinais digitais ou analógicos, respetivamente entre a rede e o microcontrolador.

A *Texas Instruments* ® tem uma gama de circuitos com esta capacidade, tendo sido escolhido para integrar a placa, o modelo DP43848, por ser o mesmo utilizado na placa de demonstração. O protocolo *Reduced Media Independent Interface* (RMII) liga o controlador Ethernet do PIC32 ao *transceiver* que interpreta a camada física do modelo OSI.

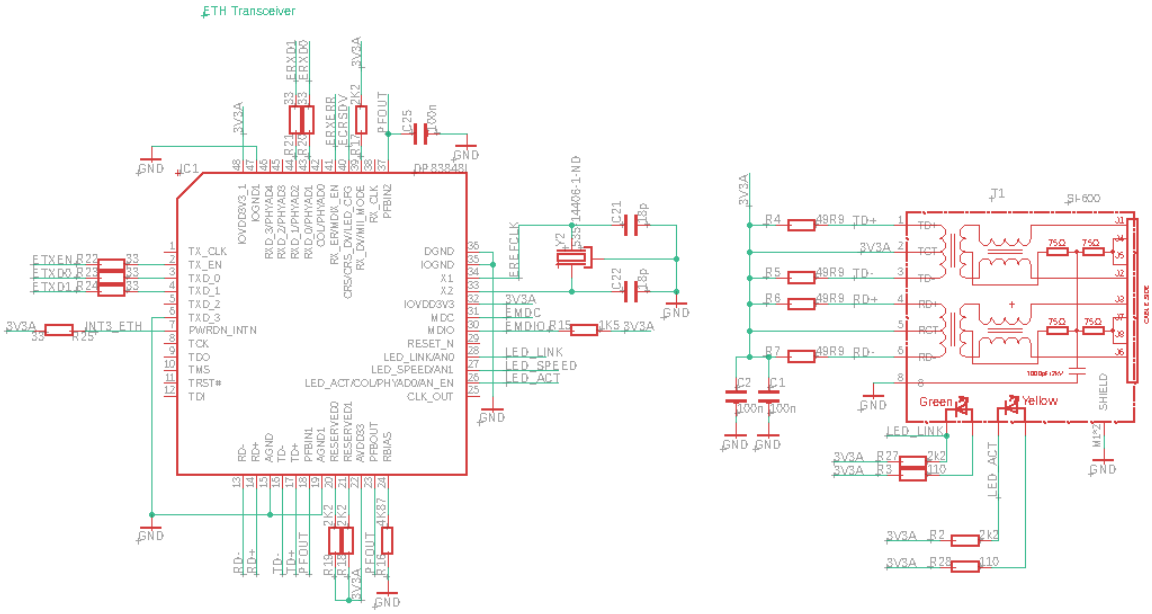


Figura 4.6: *Transceiver* Ethernet e circuito complementar.

A figura 4.6 representa o módulo de interface de rede. Os sinais entre a ficha RJ45 e o *transceiver* são sinais diferenciais pelo que é necessário tomar em consideração o comprimento das linhas assegurando que o mesmo é idêntico em ambas. Na fase de desenvolvimento foi decidido utilizar uma ficha de rede com os transformadores de linha já embutidos, pela simplicidade na implementação, bem como pelo menor espaço necessário em relação à utilização de transformadores externos.

4.4.3 Desenvolvimento da placa

O processo de desenho da PCB tomou em consideração todas as escolhas discutidas nas secções anteriores.

O desenho do esquemático teve por base a figura 4.4, o que ajudou a separar modularmente a eletrónica da placa. O desenho do circuito teve como base o circuito da placa de demonstração da Microchip® e todas as especificações técnicas dadas pelos fabricantes dos componentes. A tabela apresentada no apêndice A identifica todos os componentes necessários para a construção de uma placa.

No processo de desenho do circuito foi necessário ter em conta todo o tipo de sinais existente na placa e as especificações aconselhadas pelos fabricantes dos componentes. A escolha da posição dos componentes teve em conta a posição das fichas e o tamanho da placa, visto que esta teria que ser integrada no local da antiga *gateway*. Na face de cima do circuito encontram-se o microcontrolador, os *transceivers*, as fichas, o regulador de tensão, os LEDs e componentes essenciais para o funcionamento. A face de baixo ficou reservada para condensadores de desacoplamento e filtros na linha de alimentação. O *layout* da face superior e inferior podem ser visualizados no apêndice A.

No desenho do circuito impresso foi dada uma particular atenção aos elementos correspondentes ao circuito de rede uma vez que este se apoia em sinais diferenciais. Isto é verdade, quer para o caso do Ethernet, quer para o caso do barramento CAN. Para isso foram seguidas as recomendações dos fabricantes dos componentes descritas nos *datasheets* do *transceiver* CAN e Ethernet [26]. A nível do regulador de tensão, para evitar o sobreaquecimento foi criada, nas duas faces da placa, uma área destinada a dissipar calor. A figura 4.7 representa uma ilustração da placa desenhada e a figura 4.8 é uma fotografia da PCB pronta a ser montada num robô CAMBADA.

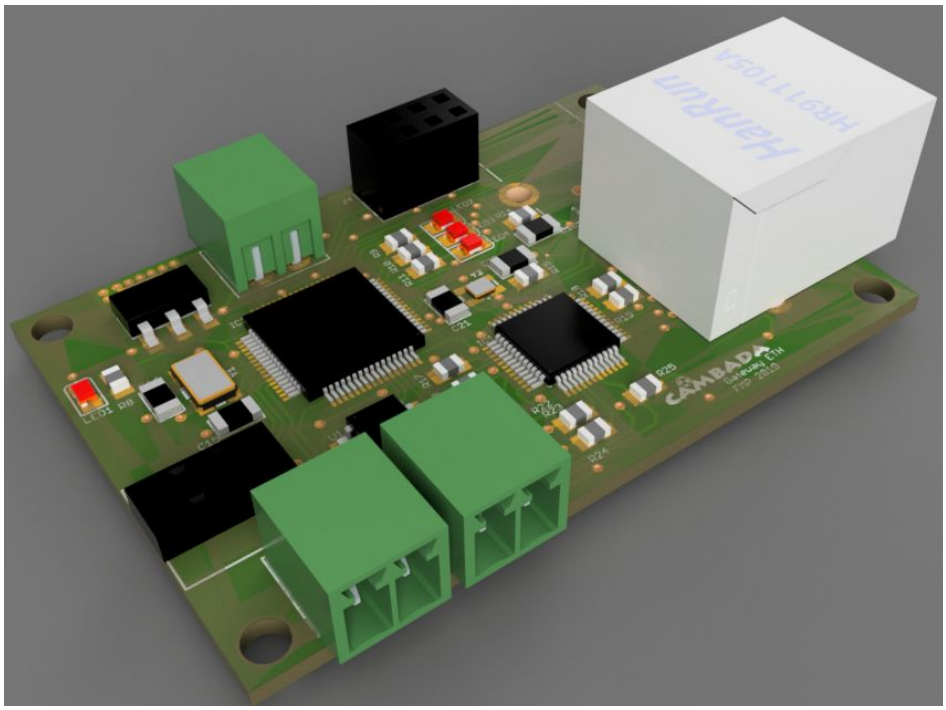


Figura 4.7: Modelo da placa final da *gateway*.

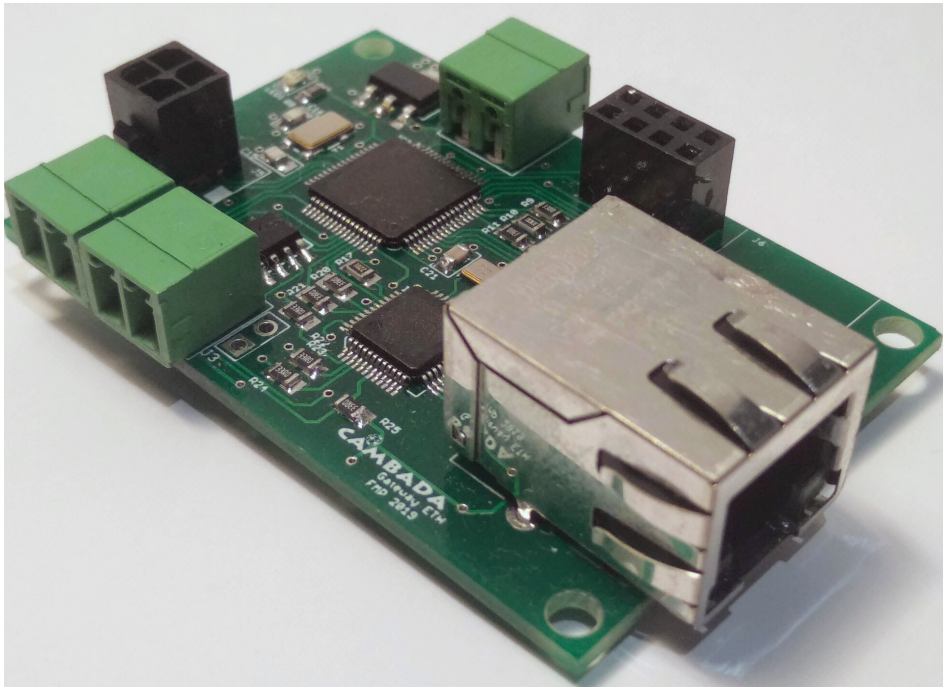


Figura 4.8: Fotografia do aspeto final da *gateway*.

CAPÍTULO 5

Ensaios e Análise de Resultados

Na sequência do desenvolvimento da nova arquitetura e posterior implementação da *gateway* apresentada no capítulo 4, foi necessário proceder a um conjunto de testes para a caraterização do desempenho da nova placa e validação da mesma face aos requisitos pré-estabelecidos. A caraterização do desempenho serviu para avaliar quais os tempos de processamento da mensagem, os tempos de envio e receção e o consumo energético da placa.

Ao longo deste capítulo são apresentados os resultados de um conjunto de testes comparativos entre a nova e a antiga *gateway* e a respetiva análise de resultados.

5.1 Tempos de processamento de mensagem

A introdução da nova *gateway* na arquitetura CAMBADA e a capacidade alargada do pacote UDP para transportar mais informação presume-se ser uma mais-valia uma vez que permite concatenar todas as tramas numa só mensagem. Para validar este pressuposto, é necessário avaliar se esta alteração é benéfica, verificando a diferença temporal entre os dois procedimentos.

Na análise das diferenças temporais entre o envio de todas as tramas numa só mensagem ou o envio de uma trama por mensagem com a informação comportamental do robô, foram usados os pinos livres do microcontrolador. Com a ajuda de um analisador lógico mediram-se os tempos de processamento, desde que é recebida uma mensagem na placa, vinda do UCD, até que esta é despachada na rede CAN, e vice-versa. O importante são as mensagens no sentido da UCD - *gateway*, pois estas são as que poderão ser compactadas numa só mensagem. O tamanho das tramas, incluindo o cabeçalho e o rodapé, enviadas desde a UCD até ao MBN do robô, tem a seguinte dimensão:

- *SetPoints* dos Motores: 10 *bytes*;
- Informação de Chuto: 7 *bytes*;
- Modo de Chuto: 5 *bytes*;
- Modo dos *grabbers*: 6 *bytes*;
- Sincronização do IMU: 6 *bytes*.

Adicionando à mensagem o DLE, conforme descrito na secção 3.1, em média são enviados 41 bytes por cada ciclo. Caso seja pedido para levantar a estrutura do guarda-redes, a mesma mensagem passa a ser constituída por 50 bytes. O principal objetivo do teste feito serviu para verificar se compensa enviar uma mensagem com cada trama ou todas as tramas numa única mensagem. Para isso, foi só considerada uma comunicação normal, ou seja, 5 tramas numa mensagem a enviar para o MBN com um número médio de 41 bytes. A figura 5.1 mostra o tempo que o microcontrolador demora a processar os dados, desde que recebe uma mensagem do UDP até que envia a mesma para a rede CAN (RX). Na mesma figura está igualmente representado o tempo a processar os dados no sentido oposto da comunicação, ou seja, da rede CAN para a UCD (TX) .

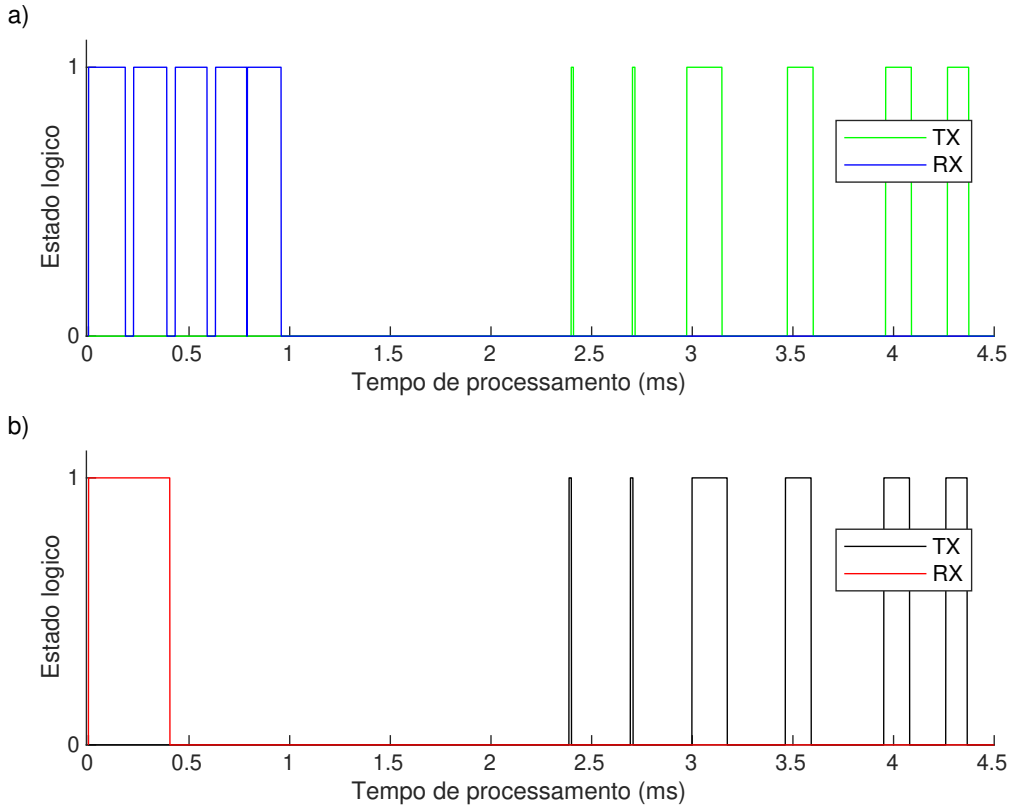


Figura 5.1: Tempo de processamento de uma mensagem. a) envio de uma trama por mensagem; b) envio de todas as tramas numa só mensagem.

Na alínea a) da figura 5.1 são apresentados os tempos de transferência de uma trama por mensagem. O tempo necessário para enviar todas as cinco mensagens está, em

95.2% dos casos, no intervalo de aproximadamente $951\mu s$ a $960\mu s$. Por outro lado, no caso alínea b), a transferência de todas as tramas numa só mensagem está, em 93.1% dos casos, no intervalo de aproximadamente $404\mu s$ a $407\mu s$. Comparando as duas abordagens verifica-se que no caso a) a transferência do total das tramas no pior dos caos é mais demorada cerca de $556\mu s$. Esta diferença deve-se ao facto de, ao nível do HWcomm, por cada nova mensagem ser necessário aceder a RTDB encapsular a nova informação e proceder ao seu envio por UDP para a *gateway*. Desta forma e com este teste pode-se concluir que é vantajoso enviar todas as tramas numa só mensagem. Não só o tempo total necessário para o efeito é mais curto como a ocupação da rede Ethernet é mais eficiente e com menor impacto ao nível de problemas de arbitragem. Acresce ainda que, com a segunda abordagem, a diferença no tempo de chegada das tramas cada um dos módulos do MBN ser significativamente reduzida. No caso da transferência de informação no sentido MBN para UCD, e com pode observar-se, os tempos envolvidos nos 2 casos são semelhantes, pois cada placa envia uma mensagem sempre que tem novos dados e, por isso, a única alteração é simplesmente a mudança do modo de comunicação.

5.2 Variação do tempo de espera

A avaliação do tempo que uma mensagem demora desde que sai da HWcomm até à *gateway* e voltar, oferece uma perspetiva do tempo necessário desde que a mensagem é enviada da UCD até que é recebida na *gateway* para que seja analisada e enviada para a rede CAN.

No barramento Ethernet existe a capacidade de ter vários dispositivos ligados ao mesmo tempo, a partir de um *swicth* de rede. O teste serviu para caraterizar as diferenças no tempo de comunicação, a partir de diferentes tempos de envio de uma nova mensagem. Para além disso, é testado em dois cenários diferentes: um com a câmara a funcionar, com as configurações normais usadas no robô CAMBADA (uma imagem a cada 20 ms) e outro sem a câmara ligada. Inicialmente é necessário saber se o *swicth* de rede tem a capacidade para processar toda a informação que irá passar na rede. A nova *gateway* irá necessitar de uma largura de banda de 2.5 kB/s, conforme a equação a baixo:

$$\textit{Tamanho Da Mensagem * Mensagens Enviados Por Segundo} = 50 * 50 = 2500 bytes/s = 2.5 kB/s$$

A câmara, ao transferir uma imagem a cada 20ms, necessita de uma largura de banda de 50 MB/s, como se pode confirmar pela equação seguinte[1]:

[1]Nas câmaras utilizadas a informação que é enviada para UCD corresponde à imagem *Bayer*. A

$Dimensão\ Da\ Imagem = 1024 * 1024 \text{ pixeis} = 1MB$
$Dimensão\ Da\ Imagem\ *\ fps = 1MB * 50 = 50MB/s$

Deste modo, os dispositivos envolvidos, *swicth*, porta de rede do computador e câmara de vídeo, suportam o modo GigaBit, encontra-se confortavelmente dentro dos limites de transferência de informação dos mesmos.

Para isso, foi criado um teste com vista a medir o tempo de viagem de uma mensagem desde que sai da UCD até que regressa. A figura 5.2 mostra a relação entre o tempo de viagem da informação na rede, enviada da HWcomm, e o tempo de espera entre comunicações sucessivas. Em cada ponto do gráfico é repetido o mesmo teste 10000 vezes e feita uma média. No mesmo gráfico estamos perante a comparação quando a câmara está conectada à rede ou não.

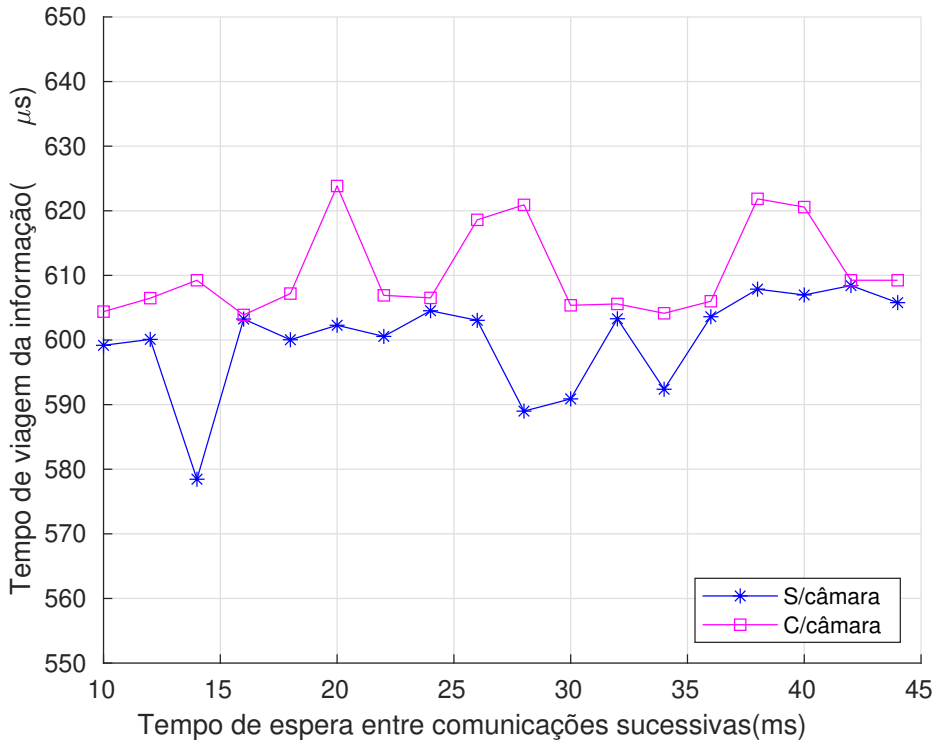


Figura 5.2: Relação entre o tempo de viagem da informação na rede e o tempo de espera entre comunicações sucessivas.

A análise estatística dos resultados na figura 5.2, permite calcular a informação compilada na tabela 5.1. Verifica-se que numa comunicação Ethernet, o tempo de comunicação não varia substancialmente com os diferentes tempos entre mensagens nem

geração da imagem *Red Green Blue* (RGB) é realizada ao nível do *device driver*. Logo, por cada pixel é apenas enviado um *byte*

com a adição da câmara na rede, conquanto neste último caso se verifique um ligeiro aumento do atraso temporal. Saliente-se, no entanto, que o teste realizado representa o pior caso uma vez que não é realizada qualquer sincronização temporal entre o processo de Visão e o HWcomm. Em condições normais de funcionamento no robô, e uma vez que o HWcomm é executada, juntamente com a visão, sempre após a receção de uma nova imagem é expectável que o aumento do atraso temporal verificado com a câmara ligada possa na realidade vir a desaparecer.

	C/ câmara (μs)	S/ câmara (μs)
Média	610.63	601.27
Variância	052.09	032.58
Máximo	623.84	608.42
Mínimo	603.88	588.99

Tabela 5.1: Estatística da relação entre o tempo de viagem da informação na rede e o tempo de espera entre comunicações sucessivas.

5.3 Comparação entre tempos de Comunicação

Com vista a verificar se existe uma melhoria temporal em termos de comunicação com a construção da nova *gateway*, foram realizados um conjunto de testes de modo a comparar o desempenho de ambos os modelos (USB e ETH). Para isso e conforme o ensaio feito na secção anterior mediu-se o tempo de viagem de uma mensagem desde que sai da UCD até que regressa.

Este teste começa por enviar uma mensagem da HWcomm composta por 50 bytes (pior caso), conforme explicado na secção 5.1. A mensagem enviada contém um identificador único. A *gateway* ao receber a mensagem reenvia a mesma para a UCD. O processo a correr na UCD, por sua vez, ao receber a mensagem, valida o identificador e calcula a diferença de tempo entre o envio e a receção, o qual é armazenado. A cada 20ms é enviada uma nova mensagem, recriando assim uma comunicação similar, a nível do tamanho da mensagem e tempo entre mensagens, ao que acontece no normal funcionamento do robô. O teste foi repetido para cada caso em estudo um milhão de vezes, o que equivale a 10 jogos seguidos, aproximadamente cinco horas e meia. A existência do identificador único destina-se a verificar a eventual perda de mensagens.

5.3.1 USB

Ao contrario da ligação Ethernet a placa da *gateway* USB usa uma ligação ponto-ponto que não é influenciada pela comunicação da câmara de vídeo. No caso desta *gateway* o teste realizado teve como objetivo verificar qual a variação de tempo de

viagem dos dados na comunicação de acordo com o explicado na introdução desta secção.

Na figura 5.3 é possível observar que este tempo de viagem tem uma variação entre 4.61ms e 20.71ms com uma distribuição aproximadamente uniforme. Este facto resulta de dois fatores associado à comunicação USB: no sentido UCD para a *gateway* há um atraso igual ou superior a 1 ms desde que é solicitado o envio da mensagem ao *device driver*; no sentido contrario e de acordo com a documentação [27] o *driver* do FTDI só envia a mensagem que estiver no seu *buffer* a cada 16ms não sendo possível ao utilizador saber quando esta é enviada. Sendo os dois processos de envio assíncronos entre si o resultado é uma distribuição dos tempos de viagem marcadamente significativa que pode exceder, como se vê, os 20ms. Este dois fatores podem, por um lado ser responsável por um *jitter* não desprezável no envio das mensagens para o MBN e ainda, no sentido contrário, à não receção do lado UCD de informação por um período de tempo superior a um ciclo de processamento.

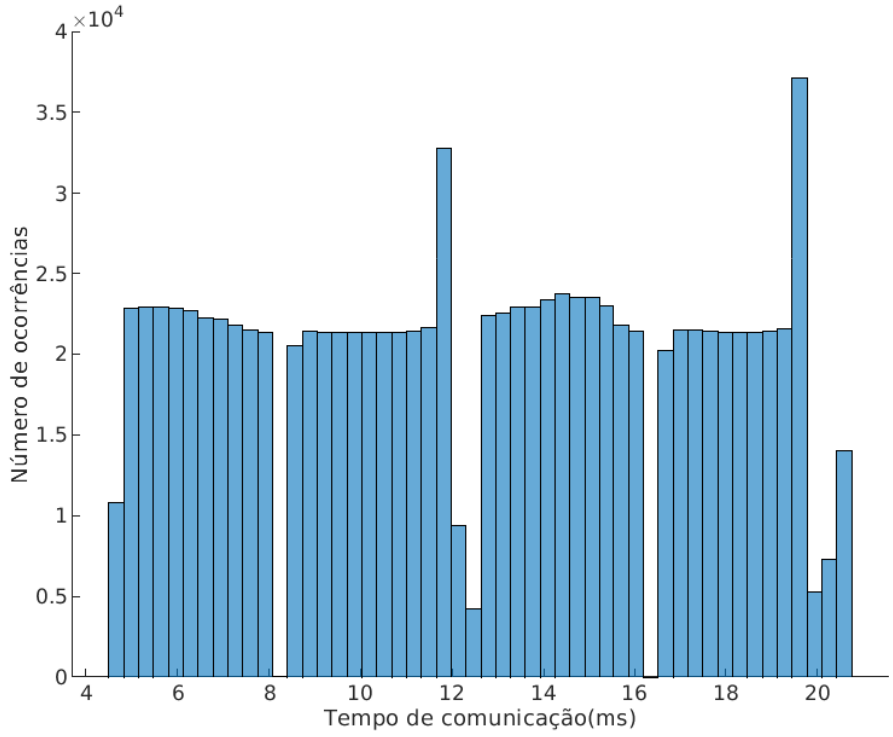


Figura 5.3: Envio de uma trama de 50 bytes 1 milhão de vezes por USB.

Ao analisar a comunicação por USB e os dados apresentados na figura 5.3, obteve-se a tabela 5.2 que contém a análise estatística dos mesmos. Em média a comunicação é de 12.52ms mas, como já foi discutido, a distribuição dos tempos é significativa e corresponde aproximadamente a mais ou menos 8ms em torno da média.

	Comunicação USB (ms)
Média	12.52
Máximo	20.71
Mínimo	4.61

Tabela 5.2: Estatística relativa ao tempo de comunicação de 1 milhão de amostras com a *gateway* USB.

5.3.2 Ethernet

A comunicação Ethernet foi também verificada com e sem a câmara ligada. Realizaram-se estes dois testes devido ao controlador do computador e o *swicth* de rede terem de suportar um elevado volume de dados a cada 20ms. De realçar ainda que no teste com a câmara ligada não é realizada qualquer sincronização entre a receção das imagens e o envio das mensagens para a *gateway*. Este é, consequentemente, o pior caso em termos de potencial para gerar colisões na rede. No histograma da figura 5.4 estão apresentados os dois casos. No teste sem câmara podemos observar um pico que corresponde a mais de 25% das amostras entre $639\mu s$ e $648\mu s$, sendo este o tempo mais provável de comunicação. Por outro lado, no teste em que temos a câmara ligada, a dispersão é maior, devido ao facto de o controlador do *switch* ter mais informação a gerir quando os dois dispositivos estão ligados em simultâneo. O funcionamento dos dois dispositivos em simultâneo adiciona um pouco de *jitter* na comunicação, mas substancialmente menor em relação ao que acontece na *gateway* USB.

A tabela 5.3 apresenta os resultados estatísticos das medições executadas. Verifica-se, em média, que ao adicionar a câmara ao sistema, o tempo de comunicação aumenta ligeiramente e há uma maior dispersão no tempo de comunicação.

	S/ câmara (μs)	C/ câmara (μs)
Média	639.33	597.43
Desvio Padrão	028.47	089.30
Máximo	942.00	997.00
Mínimo	471.00	401.00

Tabela 5.3: Estatística relativa ao tempo de comunicação de 1 milhão de amostras com a *gateway* Ethernet.

Assim face aos testes realizados verifica-se que a nova *gateway* apresenta, em relação à antiga, uma diminuição de cerca de 20 vezes no tempo de comunicação, o que beneficia a nível do comportamento do robô pois as mensagens são recebidas com um menor atraso.

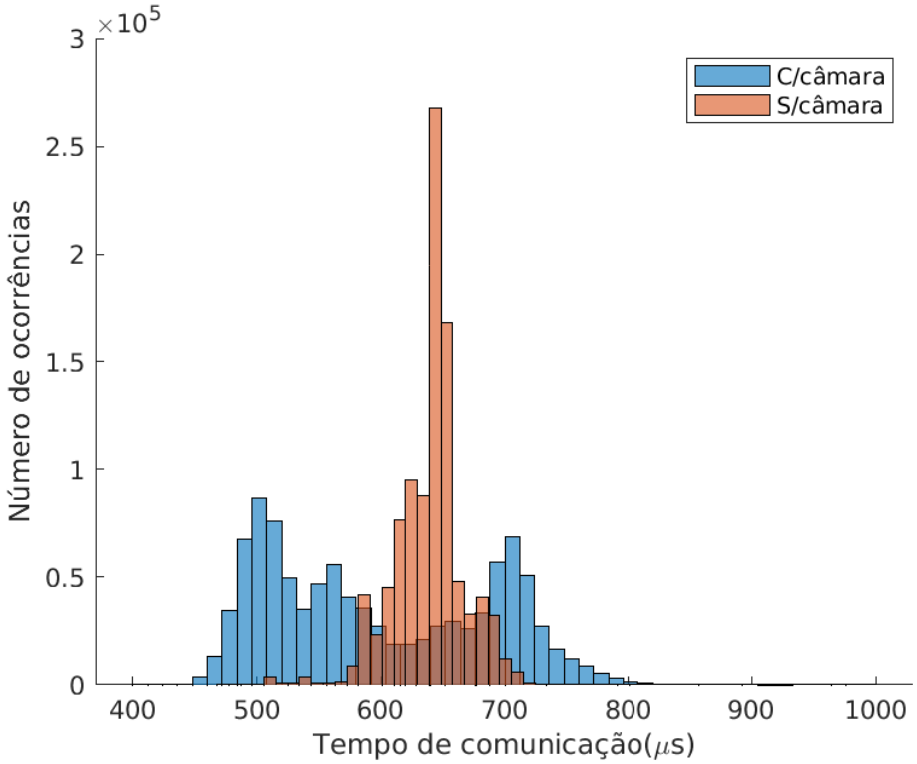


Figura 5.4: Envio de uma trama de 50 bytes 1 milhão de vezes por Ethernet.

5.4 Tempo de comunicação vs. carga na rede

Com o recurso a utilização do UDP sobre Ethernet passa a ser possível enviar mais informação numa única mensagem. Devido a isso, foi criado um teste para verificar qual a carga máxima da mensagem que é possível processar na nova *gateway* e, por sua vez, qual o tempo de comunicação envolvido.

Neste teste começou-se calcular o tempo de ida e volta de uma mensagem com o tamanho médio que se envia numa comunicação com o robô (50 bytes), aumentando gradualmente até à capacidade máxima que a *gateway* consegue processar sem perda de informação. De acordo com os testes efetuados a dimensão máxima suportada numa mensagem para as condições indicadas é de 400 bytes. Entre cada mensagem enviada, o sistema espera 20ms devido a este ser, em funcionamento normal, o tempo entre envio de mensagens da UCD para o MBN. O teste serviu para analisar qual a carga máxima na comunicação entre o UCD e o MBN em condições normais de funcionamento num robô. Para cada valor de carga o teste foi repetido 10.000 vezes, usando sempre as mesmas condições, tendo no fim sido obtida a média total para o tempo de comunicação. A figura 5.5 representa graficamente a média das medições realizadas.

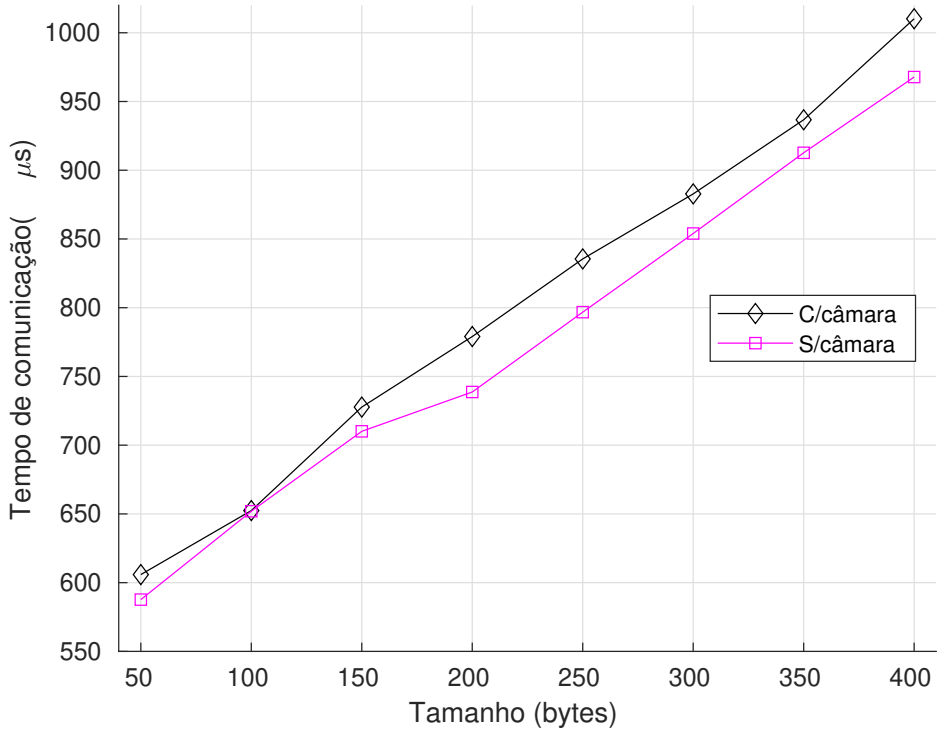


Figura 5.5: Relação de tempo com o tamanho da mensagem de dados.

Como se pode observar, o aumento do tempo de comunicação é aproximadamente proporcional ao aumento do tamanho da mensagem. Note-se, em particular, que uma mensagem com 400*bytes* (8 vezes superior ao normal) tem um tempo de comunicação que não chega a ser o dobro do tempo da comunicação da mensagem base.

No gráfico da figura 5.5 podem-se observar os resultados de dois ensaios, um com a câmara ligada e outro com a câmara desligada. Este teste confirma o analisado anteriormente, ou seja, que a introdução da câmara traz um ligeiro aumento no tempo de comunicação. No entanto, a variação ao longo da carga transferida numa mensagem é similar em ambos os casos.

5.5 Consumo de corrente

A mudança de protocolo de comunicação implica a alteração de componentes, que por sua vez aumentam o consumo energético. Por isso, durante o funcionamento da *gateway* USB e Ethernet, o consumo foi medido e caraterizado.

Analisando os *datasheets* dos componentes, verifica-se que o microcontrolador, nas configurações de funcionamento com um relógio de 80MHz, tem um consumo típico de 85mA, o *transceiver* Ethernet, em comunicação, consome 81mA e sem comunicação, 14mA e o *transceiver* CAN consome tipicamente 35mA [26] [28] [29].

No ensaio realizado foi necessário o auxílio de um multímetro com capacidade para gravar as medições durante um determinado tempo. Para isso, durante noventa segundos de comunicação foi gravada a corrente medida para posterior análise. Inicialmente as duas *gateways* estavam desligadas. Na *gateway* Ethernet, para verificar o consumo base, sem comunicação, só foi ligado o *swicth* após alguns segundos do início, o que não é necessário para a placa com USB. Ao analisar o consumo de corrente, representado na figura 5.6, verifica-se que, em média, a placa com USB consume 52mA enquanto que a placa com Ethernet à espera de começo da comunicação consome aproximadamente 142mA e em comunicação total à volta de 210mA. Com base nos *datasheets*, os valores medidos vão de encontro ao esperado.

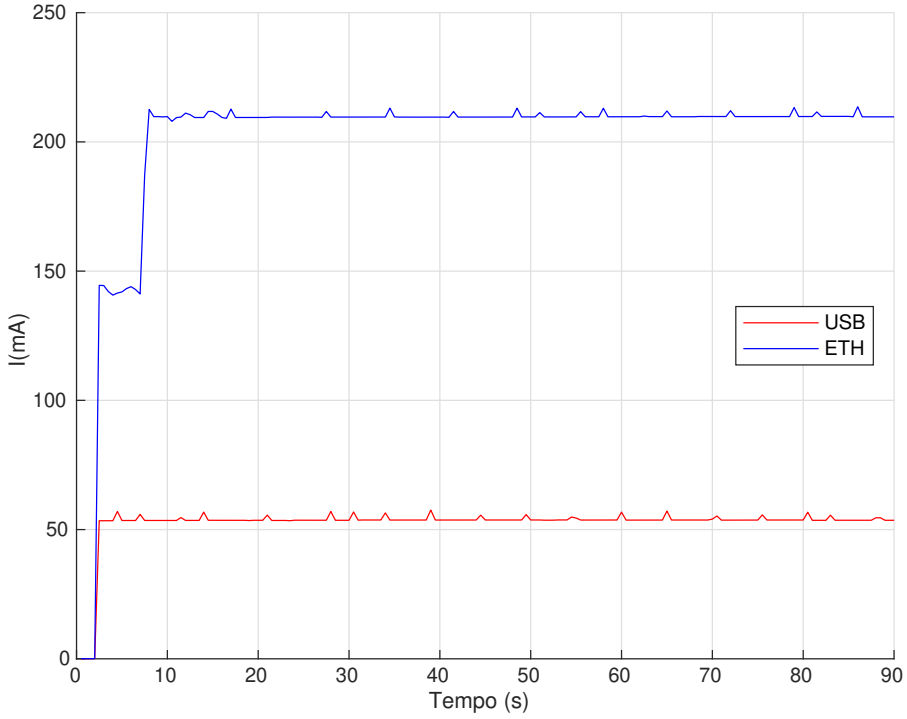


Figura 5.6: Consumo de corrente das 2 *gateways*.

Para alem das óbvias vantagens em termos de fiabilidade das ligações eletromecânicas, pode-se ainda deduzir a partir dos dados medidos, que a nova *gateway* traz um maior benefício em relação aos tempos de comunicação, e maior capacidade de transporte de informação por mensagem. No entanto isto implica um maior consumo a nível de baterias. Com a análise do consumo de corrente, verificou-se que a mudança de paradigma, a *gateway* vai consumir aproximadamente quatro vezes mais que a *gateway* anterior.

CAPÍTULO 6

Conclusões e Trabalhos Futuros

O desenvolvimento da nova *gateway* teve início com a análise do funcionamento da *gateway* já existente no robô CAMBADA. Com a análise executada delineou-se o trabalho para o desenvolvimento da nova *gateway*, com comunicação por Ethernet. Esta alteração espera vir a oferecer vantagens na melhoria do tempo de comunicação, diminuir os atrasos na chegada das mensagens e acabar com as perdas de conetividade derivada à fragilidade da ligação USB.

Com a escolha do microcontrolador foi necessário verificar a viabilidade de criar uma comunicação por Ethernet entre o microcontrolador e o computador, e para tal usou-se uma placa de desenvolvimento. Esta placa é composta por um microcontrolador, da mesma família do pensado para a *gateway*, e por um módulo Ethernet. Com este trabalho inicial foi possível desenvolver uma primeira comunicação via Ethernet e criar um conjunto de funções com o intuito de facilitar a utilização da comunicação por UDP.

Após a comunicação estar a funcionar, o desenvolvimento do *firmware* e *software* foram paralelizados, de modo a ser possível testar a comunicação. O *software* começou por adaptar o existente, garantindo a retrocompatibilidade com a versão antiga da *gateway*. Para além disso, no caso da comunicação Ethernet, o envio passou a concatenar todas as tramas numa só mensagem. A receção continua a funcionar de igual forma, mas com um diferente protocolo de comunicação. Por outro lado, o *firmware* começa por verificar se existem novas mensagem recebidas por Ethernet, descompacta-as e envia-as posteriormente para a rede CAN. Após isso, verifica se existe uma nova mensagem no CAN e caso exista, envia-a para o computador. Todo este processo deve ser o mais eficiente possível.

Finalizado o desenvolvimento do *software* e do *firmware* foi necessário passar para o desenho de uma nova placa da *gateway* para poder ser montada no robô. Para isso, e com base nos esquemáticos da placa de demonstração utilizada e nas especificações

existentes, procedeu-se ao desenho do esquemático e posterior *layout* da placa. Para o desenho do *layout* foi tido em conta o tipo de sinais existentes na placa e especificações técnicas de cada componente.

Passado o desenho da nova placa da *gateway* e respetivo código foi necessário executar um conjunto de testes para validar a *gateway* Ethernet. A validação foi um dos pontos mais importantes para se perceber a possibilidade de integrar a nova *gateway* nos robôs CAMBADA. Foi necessário efetuar a caraterização da placa para se perceber quais as suas limitações. Os testes passaram por verificar o tempo que demorava a processar cada mensagem nova, o tempo de comunicação entre a placa e o computador, a capacidade máxima de carga suportada pela placa e o consumo que a nova *gateway* irá ter. Alguns dos testes foram criados de modo a poderem ser realizados nos dois modos de comunicação, a fim de as comparar para perceber as suas diferenças.

A análise dos resultados obtidos revelou que para uma comunicação Ethernet normal, entre UCD e o MBN, o tempo diminuiu aproximadamente vinte vezes, com a possibilidade de transportar muito mais informação num só envio, isto derivado da mudança de comunicação. Porém, as melhorias têm um custo adicional, neste caso um consumo elevado de corrente da *gateway* Ethernet, aproximadamente quatro vezes mais que a anterior.

Após o desenvolvimento de *hardware*, *firmware* e *software* e posterior fase de testes de validação exaustivos, chegou-se a uma versão estável da nova *gateway*. O cumprimento de todos os requisitos desejados levou a que esta esteja atualmente integrada nos robôs CAMBADA, com sucesso. Desta forma, esta pode ser usada como ponto chave na comunicação do UCD para o MBN.

6.1 Trabalhos futuros

O desenvolvimento da nova *gateway* abriu um conjunto de possibilidades futuras, com o intuito de contínuo aperfeiçoamento da comunicação entre a UCD e o MBN. As melhorias que se propõem na continuação do trabalho são:

- Criar uma página *web* ou uma aplicação, que possibilite a configuração de parâmetros da rede Ethernet, como o IP, estático ou dinâmico, submáscara de rede, entre outros e os filtros da rede CAN, ou seja, os IDs a serem filtrados.
- Na aplicação para que se destina a *gateway*, a perda de pacotes e o tempo de comunicação entre dispositivos são relevantes. Será benéfico utilizar um protocolo de Ethernet com propriedades de tempo real, por exemplo o protocolo EtherNet/IP, usado por exemplo em ambiente industrial. Esta alteração permite, por exemplo, garantir tempos restritos de comunicação entre dispositivos.

- Visto as melhorias criadas com a comunicação por Ethernet e visto ser possível diminuir o tempo de comunicação existente entre o computador e as outras placas, uma hipótese a avaliar seria substituir a comunicação CAN de todas as placas do MBN por uma comunicação Ethernet.
- As placas desenhadas pela equipa têm um isolamento ótico entre a alimentação da lógica e a potência. Admitindo a hipótese de que a comunicação exclusivamente Ethernet possa ser uma solução vantajosa, então, por forma a diminuir a cablagem, seria também possível utilizar a capacidade existente na Ethernet de alimentar todos os sub-módulos do MBN, por *Power over Ethernet*, PoE.

Referências

[1] HMS-networks, *Industrial network market shares 2019 according to HMS*, 2019. URL: https://www.hms-networks.com/news-and-insights/news-from-hms/2019/05/07/industrial-network-market-shares-2019-according-to-hms.

[2] RoboCup, *Objective*. URL: https://www.robocup.org/objective (acedido em 03/12/2019).

[3] CAMBADA, *CAMBADA - RoboCup MSL Soccer Team - Home*, 2019. URL: http://robotica.ua.pt/CAMBADA/index.php?a=106a6c241b8797f52e1e77317b96a201 (acedido em 04/12/2019).

[4] R. Soetens, R. van de Molengraft e C. Bernardo, *RoboCup MSL - History, Accomplishments, Current Status and Challenges Ahead. RoboCup 2014*. 2014.

[5] R. A.C.R.A.d. C. Bianchi, H. L. Akin, S. Ramamoorthy e K. Sugiura, *RoboCup 2014 : Robot World Cup XVIII*. 2014, p. 719, ISBN: 3319186159.

[6] M. Asada, T. Balch, A. Bonarini, A. Bredenfeld, S. Gutmann, G. Kraetzschmar, P. Lima, E. Menegatti, P. Jonker, A. Fadaei, T. T. Nakamura, G. Steinbauer, M. Lauer, Y. Takemura, H. Lu, E. Pagello, F. Ribeiro, T. Schmitt, W.-M. Shen, H. Sprong, S. Suzuki, Y. Takahashi, P. G. Ploeger, F. Schreiber, J. Van, E. Akihiro, M. Saeed, S. Ghidary, R. Merry, B. Cunha, D. Lau, S. Ebrahimijam, O. Zweigle, A. J. R. Neves, J. Miguel, A. Hamed, R. Farad, R. S. Zhao, Y. Shota, C. Wu, J. Hao, M. Montazeri, J. Xiao, R. Dias, A. Witsch, Z. Yong, S. Ehsan, M. Wouter, H. Yifei, H. Junchong e M. E. Schreuder, «Middle Size Robot League Rules and Regulations for 2019», rel. téc., 2018. URL: https://msl.robocup.org/wp-content/uploads/2018/12/Rulebook_MSL2019_v20.pdf.

[7] D. Bastos Tavares da Silva, «RTDB2: A Flexible Distributed Blackboard», rel. téc., 2017, p. 121. URL: https://ria.ua.pt/handle/10773/23819.

[8] A. Neves, J. Azevedo, M. Cunha, J. Cunha, R. Dias, P. Fonseca, N. Lau, E. Pedrosa, A. Pereira e J. Silva, «CAMBADA'2016: Team Description Paper», rel. téc., 2016. URL: http://robotica.ua.pt/CAMBADA/docs/qualif2016/CAMBADA-tdp-2016.pdf.

[9] R Dias, F Amaral, I Angelico, J. L. Azevedo, B Cunha, P Dias, C Gomes, N Lau, P Martins, A. J. R. Neves, E Pedrosa, A Pereira, F Pinto, P Rodrigues, D Silva e J Silva, «CAMBADA'2019 : Team Description Paper», *Proceedings of RoboCup'2019*, pp. 1–8, 2019. URL: http://robotica.ua.pt/CAMBADA/docs/qualif2019/CAMBADA_TDP_2019.pdf.

[10] M. M. Alani, «Guide to OSI and TCP/IP Models», em, 8 de jul. de 2014, 2017, pp. 5–17, ISBN: 9783319051529.

[11] A. G. Blank, *TCP/IP Foundations*. 2006, p. 304, ISBN: 0782151132.

[12] Debbra Wetteroth, *OSI Reference Model for Telecommunications*, M.-H. Professional, ed. McGraw-Hill Professional, 2001, p. 320, ISBN: 978-0071380416.

[13] K. W. R. James F. Kurose, *Computer Networing A Top-Down Approach*. 2015, vol. 1, p. 862, ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[14] N. C. Objective, N. Installation, N. Media, T. Objective, N. M. Objective e N. S. Objective, *Comptia Network + Examination Objectives*, ISBN: 9781133608196.

[15] J. F. J.A. Cook, «Controller area network», *IEEE Potentials*, pp. 1–8, 2008, ISSN: 02786648. DOI: 10.1109/45.721726.

[16] Future Technology Devices International Ltd, «FT232R USB UART IC Datasheet Version 2.15», rel. téc., 2019, p. 40. URL: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

[17] VS Vision Systems GmbH, «VScom NetCAN Plus 110», VScom, rel. téc. 424, 2012, p. 6. URL: http://www.vscom.de/datasheet/424_data.pdf.

[18] proconX Pty Ltd, «CAN/Ethernet gateway», proconC, rel. téc., 2019, p. 2. URL: https://www.proconx.com/assets/files/products/caneth/DSCANETH-1901.pdf.

[19] R Dias, F Amaral, J. L. Azevedo, B Cunha, P Dias, N Lau, A. J. R. Neves, E Pedrosa, A Pereira, F Pinto, D Silva e J Silva, «CAMBADA software architecture», University of Aveiro, rel. téc., 2019, p. 7. URL: http://robotica.ua.pt/CAMBADA/docs/qualif2019/CAMBADA-software_structure-2019.pdf.

[20] ——, «CAMBADA, Hardware Description», University of Aveiro, rel. téc., 2019, p. 12. URL: http://robotica.ua.pt/CAMBADA/docs/qualif2019/CAMBADA-electrical_description-2019.pdf.

[21] Microchip Technology Inc., «32-bit PIC ® and SAM Microcontrollers Peripheral Integration», Microchip, rel. téc., 2018. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/60001455D.pdf.

[22] ——, «PIC32 Ethernet Starter Kit User's Guide», rel. téc., 2010. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/61166A.pdf.

[23] ——, «PIC32 - Section 34. Controller Area Network (CAN)», rel. téc., 2012, p. 100. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/61154C.pdf.

[24] ——, «PIC32 - Section 35. Ethernet Controller», rel. téc., 2017, p. 96. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/60001155D.pdf.

[25] ——, *32-Bit Microcontrollers (MCU) | Microchip Technology*. URL: https://www.microchip.com/design-centers/32-bit (acedido em 03/01/2020).

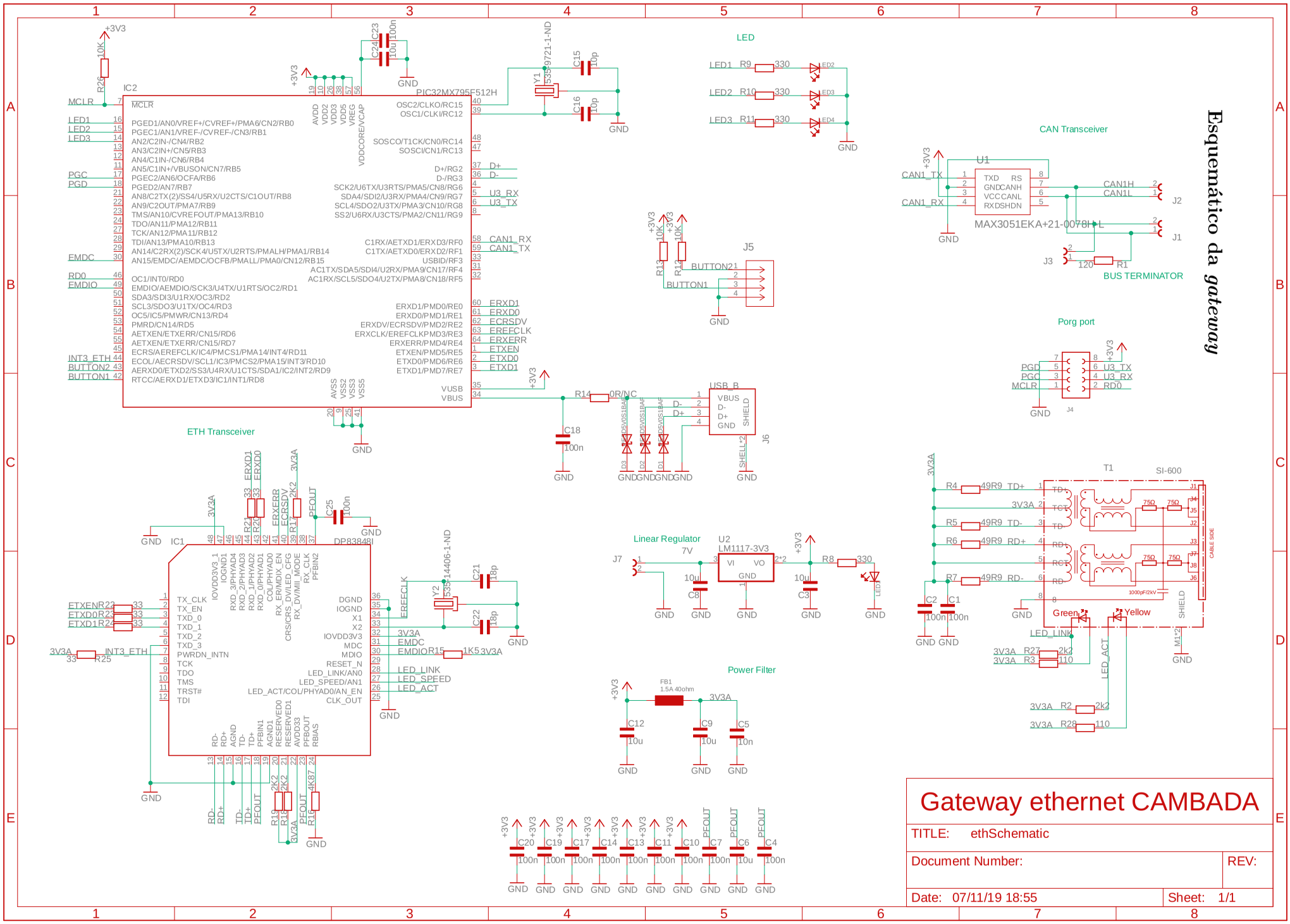
[26] Texas Instruments Inc, «DP83848-EP PHYTER™ Military Temperature Single Port 10/100 Mbps Ethernet Physical Layer Transceiver», rel. téc. 1, 2019, p. 101. URL: http://www.ti.com/lit/ds/symlink/dp83848-ep.pdf.

[27] Future Technology Devices International Ltd, «AN232B-04 Data Throughput, Latency and Handshaking», rel. téc., 2006, p. 18. URL: https://www.ftdichip.com/Support/Documents/AppNotes/AN232B-04_DataLatencyFlow.pdf.

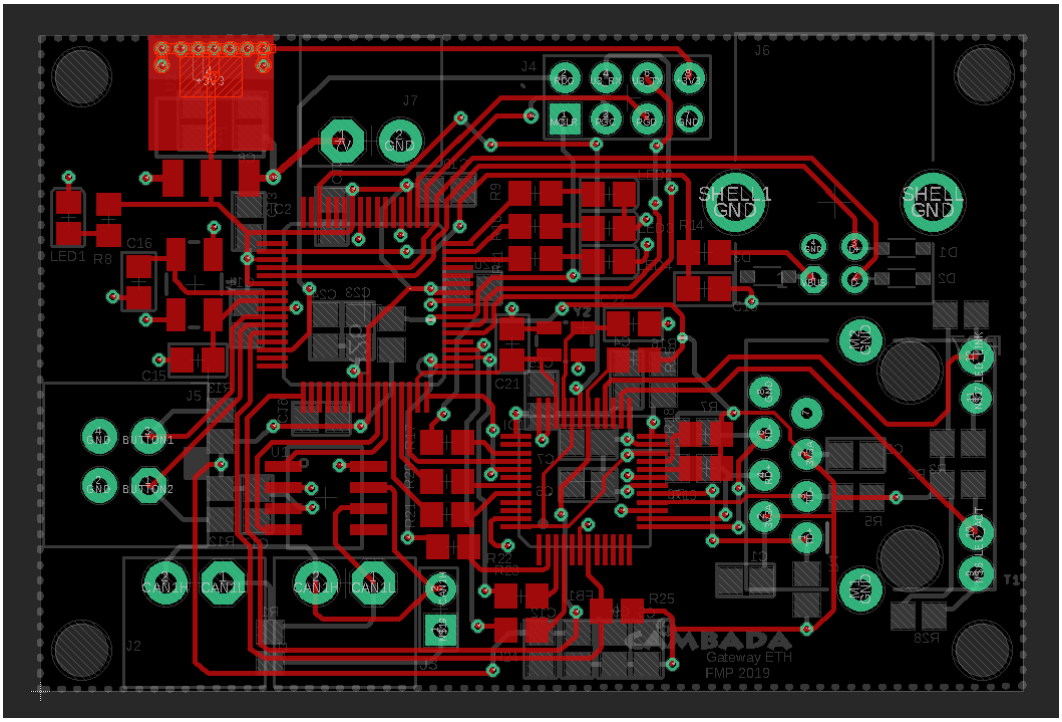
[28] Microchip Technology Inc., «PIC32MX5XX/6XX/7XX 32-bit Microcontrollers (up to 512 KB Flash and 128 KB SRAM) with Graphics Interface, USB, CAN, and Ethernet», rel. téc., 2019, p. 446. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX5XX6XX7XX_Family)Datasheet_DS60001156K.pdf.

[29] Maxim Integrated TM, «MAX3051 +3.3V, 1Mbps, Low-Supply-Current CAN Transceiver», rel. téc., 2018, pp. 1–13. URL: https://datasheets.maximintegrated.com/en/ds/MAX3051.pdf.

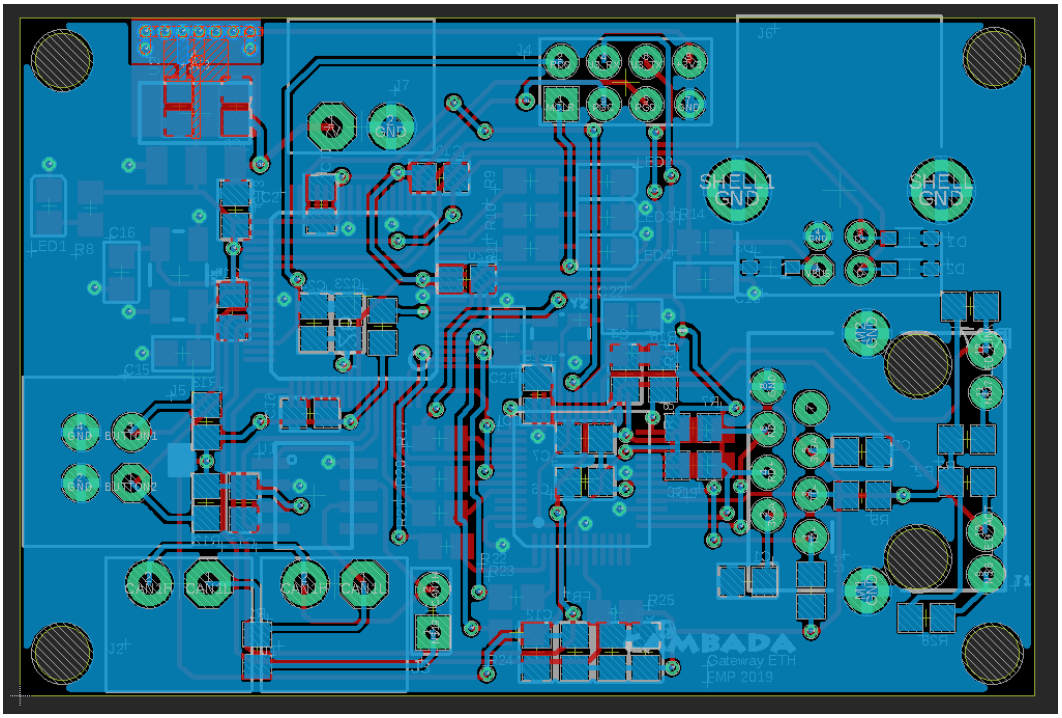
Apêndice A: Placa da *gateway*


Esquemático da gateway
IC2
PIC32MX795F512H
LED
CAN Transceiver
MAX3051EKA+21-0078+L
BUS TERMINATOR
Porg port
ETH Transceiver
DP83848I
USB_B
Linear Regulator
LM1117-3V3
Power Filter
T1
SI-600
Green
Yellow
Gateway ethernet CAMBADA
TITLE: ethSchematic
Document Number:
REV:
Date: 07/11/19 18:55
Sheet: 1/1

Face superior da PCB



Face inferior da PCB



Lista de componentes utilizado na *gateway*

Qt	Referência	Fabricante	Série	Componente
3	1803277	Phoenix contact	Micro MATE-N-LOK	J1, J2, J7
1	61300821821	Wurth elctronic	Bag	J4
1	61300211121	Wurth elctronic	Bag	J3
4	150080RS75000	Wurth elctronic	D0805	LED1, LED2, LED3, LED4
1	0R/NC	–	R0805	R14
1	1.5A 40ohm	–	FB0805	FB1
14	100n	–	C0805	C1, C2, C4, C7, C10, C11, C13, C14, C17, C18, C19, C20, C23, C25
3	10K	–	R0805	R12, R13, R26
1	10n	–	C0805	C5
2	10p	–	C0805	C15, C16
6	10u	–	C0805	C3, C6, C8, C9, C12, C24
2	110R	–	R0805	R3, R28
1	120R	–	R0805	R1
2	18p	–	C0805	C21, C22
1	1K5	–	R0805	R15
3	2K2	–	R0805	R17, R18, R19
2	2k2	–	R0805	R2, R27
6	33R	–	R0805	R20, R21, R22, R23, R24, R25
4	330R	–	R0805	R8, R9, R10, R11
4	49R9	–	R0805	R4, R5, R6, R7
1	4K87	–	R0805	R16
1	535-14406-1-ND	Abracon	CRYSTAL-SMD-3.2X2.5-4PAD	Y2
1	535-9721-1-ND	Abracon	CRYSTAL-SMD-5X3.2-4PAD	Y1
1	DP83848I	Texas Instrumentes	LQFP48	IC1
1	LM1117-3V3	Texas Instrumentes	SOT223	U2
1	MAX3051EKA	Maxim Integrated	SOP08	U1
3	PESD5V0S1BAF	Nexperia	SOD323R	D1, D2, D3
1	PIC32MX795F512H	Microchip	LQFP-ST-64-2	IC2
1	SI-600	MagJack	MAGJACK	T1
1	3-794632-4	TE Connectivity	Micro MATE-N-LOK	J5
1	USB-B	ON Shore Technology	US-BB	J6