

# MÓDULOS DE CONTROLO DE MOTORES BRUSHLESS PARA SISTEMA DE LOCOMOÇÃO DE ROBÔS

Flávio Vasconcelos



Laboratório de Sistemas Autónomos  
Departamento de Engenharia Electrotécnica  
Instituto Superior de Engenharia do Porto

2014

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Seminário/Estágio, do 3º ano, da Licenciatura em Engenharia Electrotécnica e de Computadores

Candidato: Flávio Vasconcelos, Nº 1100419 , 1100419 @isep.ipp.pt

Orientação científica: André Dias, apd@isep.ipp.pt

Empresa: Laboratório de Sistemas Autónomos

Supervisão: Guilherme Silva, gsilva@lsa.isep.ipp.pt



Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

21 de Julho de 2014

## ***Agradecimentos***

Desejo expressar os meus agradecimentos ao meu orientador André Dias-pela sua orientação e conselhos preciosos e pelas enriquecedoras discussões mantidas. Desejo agradecer sobretudo a sua disponibilidade e preocupação.

A todos os elementos do Laboratório de Sistemas Autónomos pelo descontraído ambiente de trabalho criado e apoio prestado.

Por fim, desejo expressar os meus maiores agradecimentos aos meus pais, pelo apoio e paciência que tiveram comigo. Agradeço sobretudo por me proporcionarem a oportunidade de estudar.

Flávio Lopes Vasconcelos

Esta página foi intencionalmente deixada em branco.

## Resumo

Este projeto tem como objetivo desenvolver módulos de controlo de motores *Brushless DC*, adicionando funcionalidades para disponibilizar medidas de velocidade, potência e de comunicar por barramento CAN.

Para isso realizou-se uma pesquisa sobre as várias plataformas *open-source* existentes com o objetivo de escolher a mais adequada à nossa aplicação. A escolha caiu sobre um controlador de motores *Brushless DC* personalizado.

Foi necessário realizar algumas alterações à *board* de modo a que seja possível realizar a comunicação através do protocolo de comunicação CAN. Após isso, criou-se o protocolo das mensagens a enviar para que a *interface* com este módulo de controlo fosse estabelecida de forma simples e eficaz.

O projeto *open-source* tem uma GUI, desenvolvida em QT, para que utilizador possa realizar alguns testes ao motor *Brushless DC*.

Foi adicionada a essa aplicação a possibilidade de selecionar os dados a enviar via CAN dando todo o controlo ao utilizador.

## Palavras-chave

*Brushless DC*, *Open-Source*, Linux, CAN

Esta página foi intencionalmente deixada em branco.

## Abstract

This project aims to develop control modules for brushless motors adding features to provide measures of speed and power and to communicate by CAN bus.

For this there was a research on various platforms open-source in order to choose the most appropriate for our application. The choice fell on a custom electronic speed control.

It was necessary to make some changes to the board so that it is possible to perform the communication via CAN bus. After that, we created the protocol of messages for the interface with this control module were established in a simple and effectively way.

The open-source project has a GUI, developed in QT, so that user can perform some tests to Brushless DC motor. Was added to this application the ability to select the data to be sent via CAN giving all control to the user.

## Keywords

*Brushless DC , Open-Source, Linux, CAN*

Esta página foi intencionalmente deixada em branco.

# Conteúdo

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introdução</b>                                    | <b>1</b>  |
| 1.1      | Enquadramento e Motivação . . . . .                  | 2         |
| 1.2      | Cenários de aplicação . . . . .                      | 3         |
| 1.3      | Objectivos . . . . .                                 | 3         |
| <b>2</b> | <b>Estado da Arte</b>                                | <b>5</b>  |
| 2.1      | Open-BLDC . . . . .                                  | 5         |
| 2.1.1    | Open-BLDC 0.3 . . . . .                              | 5         |
| 2.1.2    | Open-BLDC Strip 0.1 . . . . .                        | 6         |
| 2.1.3    | CLogic BLDC . . . . .                                | 7         |
| 2.2      | <i>Brushless DC Motor Controller Board</i> . . . . . | 8         |
| 2.3      | <i>A Custom BLDC Motor Controller</i> . . . . .      | 9         |
| 2.4      | Resumo . . . . .                                     | 10        |
| <b>3</b> | <b>Análise de Requisitos</b>                         | <b>11</b> |
| 3.1      | Microcontrolador . . . . .                           | 11        |
| 3.2      | Alimentação . . . . .                                | 12        |
| 3.3      | Comunicações . . . . .                               | 12        |
| 3.4      | Aplicações . . . . .                                 | 12        |
| <b>4</b> | <b>Arquitetura de Alto Nível</b>                     | <b>13</b> |
| <b>5</b> | <b>Projeto</b>                                       | <b>15</b> |
| 5.1      | Descrição de <i>Hardware</i> . . . . .               | 15        |

|          |   |           |
|----------|---|-----------|
| 5.1.1    | Controlo . . . . .                            | 15        |
| 5.1.2    | Potência . . . . .                            | 15        |
| 5.2      | Descrição do <i>Firmware</i> . . . . .        | 18        |
| 5.2.1    | BLDC . . . . .                                | 18        |
| 5.2.2    | CAN . . . . .                                 | 19        |
| 5.3      | Descrição da aplicação . . . . .              | 22        |
| <b>6</b> | <b>Resultados</b>                             | <b>27</b> |
| <b>7</b> | <b>Conclusão e Trabalho Futuro</b>            | <b>31</b> |
| <b>A</b> | <b>Motores <i>Brushless</i></b>               | <b>35</b> |
| A.1      | Tipos de motores BLDC . . . . .               | 36        |
| A.1.1    | Movimento . . . . .                           | 36        |
| A.1.2    | Ventilação . . . . .                          | 36        |
| A.1.3    | Desempenho . . . . .                          | 37        |
| <b>B</b> | <b>Protocolo CAN</b>                          | <b>39</b> |
| B.1      | Um pouco de história . . . . .                | 39        |
| B.2      | Características do CAN . . . . .              | 40        |
| B.3      | Método de Endereçamento . . . . .             | 40        |
| B.4      | Formato da mensagem . . . . .                 | 41        |
| B.5      | O Nível Físico . . . . .                      | 44        |
| B.6      | Transmissão diferencial . . . . .             | 44        |
| B.7      | Tempo de <i>bit</i> e Sincronização . . . . . | 44        |
| <b>C</b> | <b>Soldadura</b>                              | <b>47</b> |
| C.1      | Estação de Soldadura . . . . .                | 47        |
| C.2      | Fluxo e solda . . . . .                       | 47        |
| C.3      | Placa de Circuito Impresso . . . . .          | 48        |
| <b>D</b> | <b>Esquemas Elétrico</b>                      | <b>49</b> |

# Listas de Figuras

|     |  |    |
|-----|--|----|
| 1.1 | Drive de controlo de motor                 | 2  |
| 2.1 | <i>Open-BLDC</i> 0.3                       | 6  |
| 2.2 | <i>Open-BLDC Strip</i> 0.1                 | 7  |
| 2.3 | <i>CLogic BLDC</i>                         | 8  |
| 2.4 | <i>Brushless DC motor controller board</i> | 9  |
| 2.5 | <i>A Custom BLDC Motor Controller</i>      | 10 |
| 4.1 | Diagrama de blocos                         | 13 |
| 5.1 | Esquema elétrico da parte de Controlo      | 16 |
| 5.2 | Esquema elétrico da camada de Potência     | 17 |
| 5.3 | Modelo CAD da placa de circuito impresso   | 17 |
| 5.4 | Circuito Opto-isolado                      | 18 |
| 5.5 | Fluxograma do <i>firmware</i>              | 18 |
| 5.6 | <i>Nominal CAN Bit Time</i>                | 20 |
| 5.7 | Separador <i>main</i> da aplicação         | 22 |
| 5.8 | Aba CAN da GUI                             | 23 |
| 6.1 | Tensão no arranque                         | 27 |
| 6.2 | Corrente no arranque                       | 28 |
| 6.3 | <i>Software CANReal</i>                    | 29 |
| 6.4 | GUI BLDC                                   | 29 |

|  |    |
|--|----|
| A.1 Exemplo de motor brushless <i>inrunner</i> (esquerda) <i>outrunner</i> (direita) . . . . . | 36 |
| B.1 <i>Método de endereçamento</i> . . . . .   | 42 |
| B.2 <i>Método de endereçamento</i> . . . . .   | 43 |
| B.3 <i>Representação física de bit.</i> . . . . .  | 45 |
| B.4 <i>Divisão do tempo de bit.</i> . . . . .  | 45 |
| C.1 Placa de Circuito impresso durante o processo de soldadura . .                             | 48 |
| D.1 Esquema elétrico da parte de Controlo . . . . .  | 49 |
| D.2 Esquema elétrico da parte de Potência . . . . .  | 50 |

# Lista de Acrónimos

**ATV** *All Terrain Vehicle*

**AUX** *Auxiliary Port*

**BLDC** *Brushless Direct Current*

**CAN** *Controller Area Network*

**CAD** *Computer Aided Design*

**CPU** *Central Processing Unit*

**DSP** *Digital Signal Processor*

**DC** *Direct Current*

**ESC** *Electronic speed controller*

**FOC** *Field-Oriented Control*

**FPGA** *Field-Programmable Gate Array*

**GUI** *Graphical User Interface*

**I2C** *Inter-Integrated Circuit*

**JFET** *Junction Gate Field-Effect Transistor*

**KV** *RPM/Volt*

**LED** *Light Emitting Diode*

**LIP** *Lithium-Ion Polymer Battery*

**LSA** *Laboratório de Sistemas Autónomos*

**MAV** *Micro-Air Vehicle*

**PC** *Personal Computer*

**PPM** *Pulse Position Modulation*

**PWM** *Pulse Width Modulation*

**RPM** *Rotation Per Minute*

**TI** *Texas Instruments*

**UART** *Universal Asynchronous Receiver/Transmitter*

**UAV** *Unmanned Aerial Vehicle*

**USB** *Universal Serial Bus*

# Capítulo 1

## Introdução

Devido à observação de fenómenos naturais, tais como os relâmpagos durante uma trovoada, a humanidade já tem conhecimento da existência da eletricidade há muitos séculos. O estudo da eletricidade permaneceu uma curiosidade intelectual até ao século XVIII. Foi neste século que os fenómenos elétricos começaram a ser compreendidos e que se conseguiu produzir e armazenar energia com sucesso. Depois de dados estes primeiros passos, a evolução tecnológica evoluiu bastante. Começou uma nova era![4]

A eletricidade enraizou-se de tal forma, que dois séculos depois vivemos numa sociedade altamente tecnológica, onde a dependência de dispositivos elétricos e eletrónicos é bastante significativa. Como tal, os motores elétricos estão presentes numa vasta gama de aplicações.

O acionamento dos diferentes motores é feito através da associação dos motores com conversores, que contêm elementos de eletrónica de potência. Dá-se o nome de *driver* ao controlador do motor, como pode ser visualizado na Figura 1.1

O tipo de controlo implementado depende do motor em questão. *Drivers* que incorporem motores de indução necessitam de um controlo mais sofisticado, como por exemplo, orientação de campo, de modo a cumprir os requisitos. Já no caso de motores de passo, é necessário um controlo

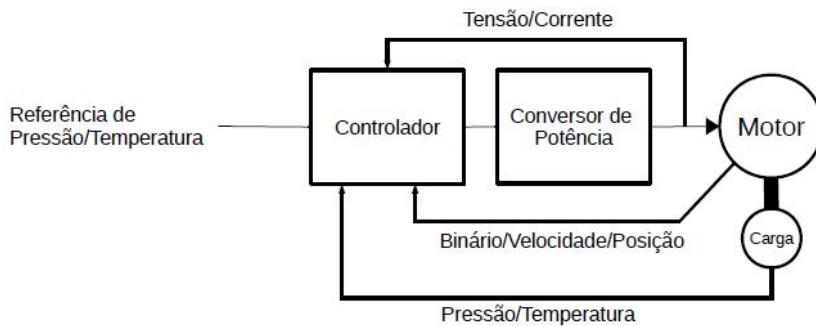


Figura 1.1: Drive de controlo de motor

de posição. Podem encontrar-se *drivers* de motores numa grande gama de potências, desde alguns Watt até milhares de kiloWatt. Em todos esses *drivers* de controlo de velocidade e posição, é necessário um conversor de potência de interface entre a potência de entrada e o motor. Diversos tipos de controlo foram desenvolvidos para os diversos *drivers* existentes.

## 1.1 Enquadramento e Motivação

A evolução dos dispositivos semicondutores de potência tornou possível o desenvolvimento de conversores eletrónicos de potência quer de muito reduzida dimensão e elevada eficiência quer de conversores industriais com aplicação em elevadas potências. As questões da miniaturização e da eficiência energética são cada vez mais importantes nos sistemas de engenharia tornando a integração de subsistemas uma necessidade. Num outro domínio, devido ao aumento da capacidade de cálculo, de armazenamento e de velocidade, de plataformas de controlo baseadas em microprocessadores, microcontroladores, DSP e FPGA, bem como a integração de periféricos, nomeadamente conversores A/D e D/A, de elevada resolução, e temporizadores com grande resolução temporal, tornou viável a implementação de algoritmos de controlo complexos, com processamento de grande volume de dados adquiridos.

Finalmente, o gradual aumento de aplicações de atuadores eletromecânicos

nos mais diversos domínios, em particular em veículos elétricos, faz com que a procura e o desenvolvimento da melhor solução para um acionamento capaz de satisfazer um determinado conjunto de critérios sejam, ainda, atividades relevantes neste domínio.[3]

## 1.2 Cenários de aplicação

Como já foi dito anteriormente, a tecnologia está a crescer a um ritmo alucinante de modo que nasce a necessidade de criar controladores de motores eficientes. No âmbito deste projeto, iremo-nos focar essencialmente nos motores de corrente contínua sem escovas (*Brushless DC*) devido à sua alta eficiência e à sua longa durabilidade.

Outra aplicação também bastante importante é o uso desta tecnologia nos *drones*. Estes precisam de um conjunto controlador + motor bastante eficiente e, por isso, é aconselhado o uso de motores *Brushless DC*. Com o avançar da tecnologia o objetivo é reduzir o tamanho dos *drones* tornando-os mais leves conferindo maior capacidade de carga para as baterias, o que significa um aumento da autonomia.

Uma equipa de pesquisadores da Universidade de Tecnologia de Delft na Holanda, desenvolveu um módulo de *autopilot* para MAV particularmente compacto. A *board* tem uma área de  $2\text{ cm}^2$  e pesa menos de 2 gramas. O *autopilot* contém um giroscópio, um acelerómetro, um magnetómetro, altímetro, GPS e um processador ARM Cortex M3 a funcionar a 72 MHz.[5]

## 1.3 Objectivos

Neste projeto temos como principal objetivo o desenvolvimento de um módulo que permita informar o *autopilot* do *robot* o estado do motor, como por exemplo, a velocidade, sentido ou outras informações pertinentes para o controlo.

No caso dos *drones* é importante certificar que os motores estão no seu prefeito funcionamento, a velocidade e a potência ativa são bastantes importantes, pois caso falhe um motor num *drone* poderá haver dois desfechos: ou cai subitamente ou, caso o *drone* esteja preparado para tal, este pode aplicar mais potência nos restantes motores de modo a minimizar os danos na queda ou, dependendo do seu porte, este pode simplesmente continuar a sua missão mesmo tendo uma falha num dos seus motores. Atualmente não é possível obter essa informação com os controladores dos motores *Brushless DC* existentes no mercado. Surge assim a necessidade de desenvolver um *driver* de motores *Brushless DC* que possa informar o ”cérebro” do *robot* acerca do estado do motor de modo a que este possa processar a informação. Essa informação será enviada do controlador até ao *autopilot* por barramento CAN pois esta é uma comunicação bastante robusta e eficaz que também é bastante usada no ramo da robótica e automóvel, mais informação sobre comunicação CAN no anexo B.

# Capítulo 2

## Estado da Arte

Neste capítulo é descrito o estado da arte e são apresentados alguns dos projetos mais relevantes sobre o controlo dos motores *Brushless DC*. Alguns deles são projetados apenas para fins educativos, outros são desenvolvidos para uma possível entrada no mercado.

### 2.1 Open-BLDC

O *Open-BLDC* é um projeto totalmente *Open-Source* que visa a criação de um controlador *Brushless DC*, ou também conhecido por, electronic speed controller (ESC). [1]

#### 2.1.1 Open-BLDC 0.3

Foram feitas algumas versões. A mais recente e validada é a 0.3 apresentada na Figura 2.1.

Algumas características neste projeto são:

- Sensores de tensão em 2 fases
- Sensores de corrente nas 3 fases.

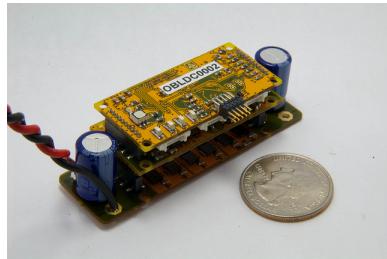


Figura 2.1: *Open-BLDC 0.3*

- *Interface* de comunicação UART, CAN e I2C.
- O controlador é composto por 3 *boards* de forma a que seja mais fácil fazer revisões ao *Hardware* e conferindo um elevado grau de modularidade.
- Corrente máxima admissível é cerca de 20A contínuos.
- Alimentação feita a 11.2V (3s) ou a 14.8V (4s).

Como já foi dito anteriormente, a grande vantagem deste projeto é a modularidade. É possível com a mesma *board* de sinais controlar diferentes módulos de potência substituindo a respetiva *board* relativa a esta camada, dependendo da aplicação. Este projeto apenas foi alvo de estudo pois não foi implementado em nenhum *drone* ou algo que pudesse mostrar que realmente era robusto ao ponto de ser colocado no mercado embora tenha sido testado em laboratório mostrando o seu funcionamento através de um vídeo no *YouTube*. [2]

### 2.1.2 Open-BLDC Strip 0.1

Este é um projeto desenvolvido exclusivamente para o uso em *Quadrotors*, como se pode visualizar na figura 2.2, devido a sua forma estreita e comprida a utilização deste projeto é apropriada para ser colocada nas hastas dos *drones*.[1]

Este projeto está ainda na fase de protótipo pois até agora apenas foi desenhada a *board* e as imagens que temos acesso são apenas de modelos CAD.

Tem características interessantes dependendo, é claro, da aplicação:

- Tamanho reduzido (55mm x 11.5 mm).
- Baixo custo de produção.
- *Interface PPM, Serial, I2C , CAN.*
- Corrente máxima admissível é cerca de 7A contínuos e 12A de pico o que é suficiente para a maioria dos *drones* até 40 cm.
- Alimentação feita exclusivamente a 11.2V (3s).

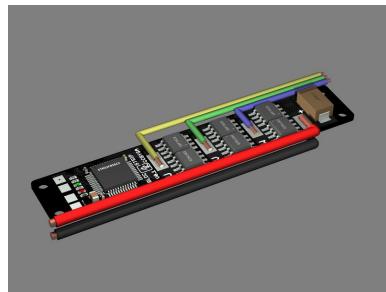


Figura 2.2: *Open-BLDC Strip 0.1*

### 2.1.3 CLogic BLDC

*CLogic* (Figura 2.3) nasceu no berço dos mesmos produtores do *Open-BLDC*. Esden foi um dos responsáveis pela criação do projeto *Open-BLDC*. Este teve a oportunidade de analisar o interior de um controlador de motores *Brushless DC* avariado, neste caso da marca *Castle Creations*, e apercebeu-se que a *Castle Creations* também constroem os seus controladores de forma modular. Então decidiu colocar a parte lógica do seu antigo projeto, *Open-BLDC 0.3* (2.1.1), a operar juntamente com a parte de potência do controlador da

*Castle Creations.* E assim nasceu o *CLogic*.[1] Com o decorrer do tempo começou a ver outros fabricantes de controladores de motores *Brushless DC* também faziam o mesmo nomeadamente as marcas *Tekin* e *Turnigy*.

A sua maior entrave neste projeto foi o tamanho pois tinha que garantir que a sua *board* lógica do projeto *Open-BLDC 0.3* (2.1.1) teria que ter dimensões reduzidas de forma a ficar dentro da estrutura o controlador de mercado.

O *CLogic* tem a maioria das funcionalidades do projeto *Open-BLDC 0.3* (2.1.1). Por causa da restrição do tamanho Esden teve que retirar os conectores da comunicação I2C e PPM e adicionou um conector AUX de modo a conectar facilmente codificadores ou sensores hall no caso de ser *sensored*.

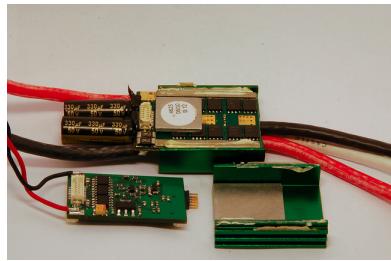


Figura 2.3: *CLogic BLDC*

É de salientar que o repositório de arquivos do *Open-BLDC* não é atualizado à cerca de 2 anos. Neste projeto o repositório de arquivos usado é o *GitHub*.

## 2.2 Brushless DC Motor Controller Board

Detentor deste projeto, Eng. Daniel Strother projetou em 2008 um controlador de motores *Brushless DC* Figura 2.4. [7]

Alguns pontos fortes:

- Sensor de corrente e tensão.

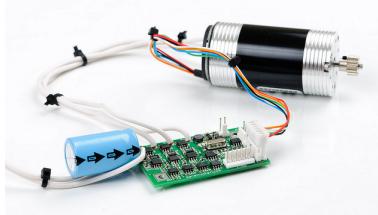


Figura 2.4: *Brushless DC motor controller board*

- *Interface CAN e RS-422.*
- Corrente máxima admissível de 30A.
- Alimentação feita na faixa de 15V a 30V

Daniel conseguiu colocar estas características todas numa placa de circuito impresso de 63 mm por 38 mm.

O propósito da criação deste projeto foi o controlo dos motores *Brushless DC* para aplicação num carro de rádio modelismo nomeadamente *TRAXXAS E-MAXX*. O que levou a substituir o controlador de motores *Brushless DC* que possuía por um personalizado foi essencialmente 3 pontos.

- Controlo do seu *TRAXXAS* em baixas velocidades.
- Odometria.
- Mecânica simples e confiável

### **2.3 A Custom BLDC Motor Controller**

Benjamin é o tutor deste projeto. Um controlador de motores *Brushless DC* personalizado (Figura C.1) [8]. Durante cerca de 3 anos Benjamin tem estado a projetar um controlador de motores *Brushless DC* com as seguintes características:

- Tensão de alimentação na faixa de 8V a 60V;

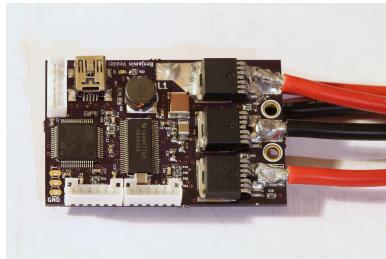


Figura 2.5: A Custom BLDC Motor Controller

- Corrente máxima na casa dos 50A (ou 240A por breves instantes);
- Tamanho reduzido (40 mm x 60 mm);
- Possibilidade de determinar a corrente em todas as fases;
- Odometria;
- Controlo da velocidade atual;
- Possibilidade de controlar o motor através de um *Raspberry PI* ou telemóvel;
- Porta USB para *debug*

Relativamente ao software, foi usado um Sistema Operativo (*ChibiOS*) servindo de base pois possibilita a criação de *threads*, aumentando assim a eficiência do microcontrolador utilizado.

Este projeto foi testado no *buggy* 1/8 de rádio modelismo e posteriormente numa *longboard*

## 2.4 Resumo

De modo a concluir este capítulo é possível ver que existem bastantes alternativas *open source* que nos podem servir como base para o nosso projeto. Porém, é preciso analisar cada uma das opções de modo a verificar a fiabilidade do projeto base.

## Capítulo 3

# Análise de Requisitos

Para a execução das aplicações apresentadas anteriormente, no capítulo 1 temos que ter em conta alguns requisitos:

- Base *Open Source*;
- Microcontrolador *ARM*;
- Diversas *interfaces* de comunicação;
- Controlador de motores *Brushless DC* de dimensões reduzidas;
- Sensor de tensão e corrente;
- Monitorizar a velocidade.

### 3.1 Microcontrolador

O controlador de motores *Brushless DC* terá de possuir um microcontrolador ARM para maior facilidade de interligação entre os restantes controladores de motores *Brushless DC* e *robots* existentes no Laboratório de Sistemas Autónomos .

## 3.2 Alimentação

O controlador de motores *Brushless DC* terá que ser alimentado por baterias visto ser endereçado a *robots* (i.e.: *drones*). Na maioria dos *drones* de médio porte é comum a utilização de baterias LIPO.

## 3.3 Comunicações

Grande parte dos *robots* disponíveis no Laboratório de Sistemas Autónomos possuem *interface CAN*. Assim sendo, este controlador de motores *Brushless DC* terá obrigatoriamente ter essa *interface* de comunicação, porém não impede a implementação de outras. Uma das grandes vantagens da comunicação CAN é a sua flexibilidade de adicionar membros à rede sem ser necessário uma reprogramação dos dispositivos já presentes na rede, assim como a sua imunidade ao ruído eletromagnético.

## 3.4 Aplicações

No Laboratório de Sistemas Autónomos existe duas possíveis aplicações para este projeto. Uma delas, a mais tendencial, é a utilização em *drones*, e a outra possibilidade é a aplicação em veículos com porte superior, nomeadamente numa ATV (elevada potência).

## Capítulo 4

# Arquitetura de Alto Nível

As estrutura principal do projeto pode ser representada de uma forma genérica com o seguinte diagrama de blocos.

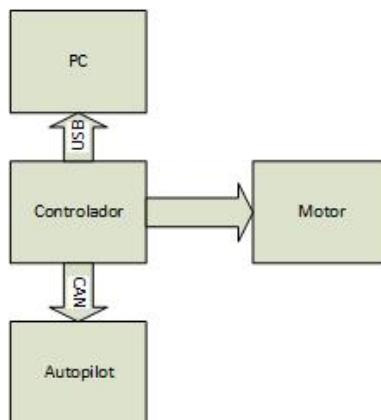


Figura 4.1: Diagrama de blocos

Temos um computador com uma aplicação desenvolvida em QT que é um *framework* multiplataforma para desenvolvimento de interfaces gráficas em C++ criado pela empresa norueguesa *Trolltech* [9]. Esta aplicação permite ao utilizador visualizar uma série de informações tais como a corrente do motor, tensão de alimentação, velocidade, entre outras. Foi necessário alterar a GUI para que seja possível configurar o tipo de informação que

será enviada via CAN.

O controlador em si é uma placa composta por 4 camadas possibilitando assim a sua montagem com cerca de 65mm por 40mm. Este dispõem de um ARM (STM32F4) que trabalha a 168 MHz. Tem ainda um driver para disparar os JFET's de modo a comutar a tensão aplicada aos enrolamentos. Com este controlador também é possível escolher diferentes *profiles* dependendo do motor utilizado.

O motor pode ser de diferentes potências mas para testes foi utilizado um motor de baixo porte com cerca de 1240kv nominais.

# Capítulo 5

## Projeto

Neste capítulo será feita a descrição do projeto. Irá ser abordado vários pontos relevantes no seu desenvolvimento.

### 5.1 Descrição de *Hardware*

Neste secção iremos abordar alguns aspetos referente ao *hardware* do sistema.

#### 5.1.1 Controlo

Para o controlo usou-se o microcontrolador ARM Cortex da ST com *interface* CAN, conforme requerido no capítulo 3. Temos uma porta USB disponível para controlar o motor *Brushless DC* e recolher informação e configuração de alguns parâmetros, como por exemplo, a informação disponibilizada via CAN. Podemos ver o esquema elétrico na figura 5.1 ou no Anexo D

#### 5.1.2 Potência

Para a camada de potência utilizou-se 6 JFET's e um *driver* (DRV8302), que é responsável pelas comutações entre os JFET's. Estes permitem até 240A

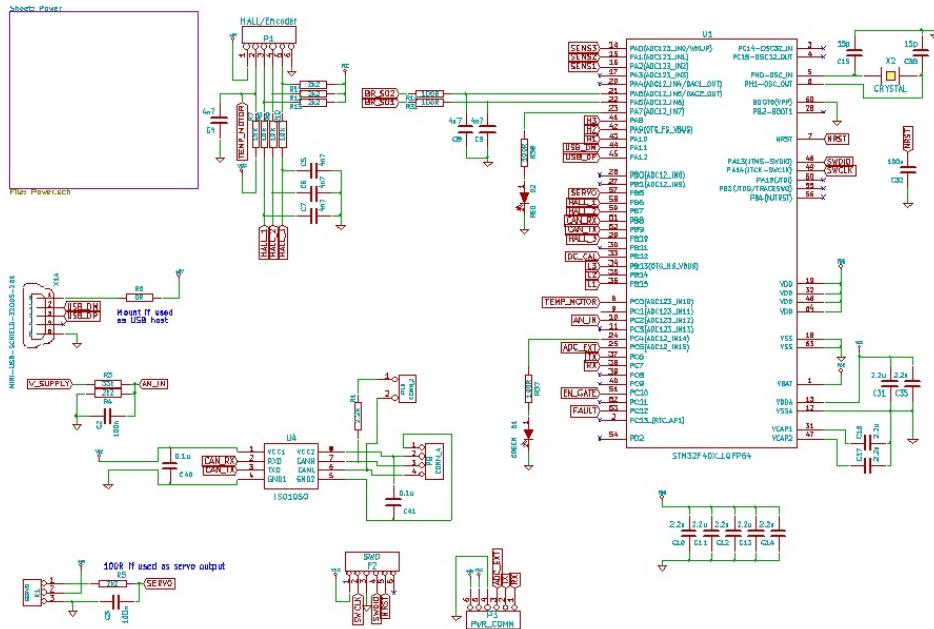


Figura 5.1: Esquema elétrico da parte de Controlo

durante alguns segundos ou cerca de 50A de forma contínua dependendo da capacidade de dissipação de calor.

Para ”disparar” os JFET’s foi usado um driver também da TI. Este trabalha numa gama de tensão de 6V a 60V. Com este driver temos acesso a 5V 1A do seu *buck converter* o que nos confere a presença de 5V fixos em toda a placa. A alimentação de alguns componentes é estabelecida a uma tensão de 3.3V proveniente um regulador de tensão. Ficando assim com 3.3V, 5V e tensão de alimentação disponíveis na placa. Podemos ver o esquema elétrico 5.2 na figura ou no Anexo D.

A placa de circuito impresso teve se sofrer alterações para que fosse possível acrescentar um *transceiver* CAN. Foi usado o *transceiver* ISO1050 da TI pois este *transceiver* é opto-isolado, ou seja, permite que se possa transferir um sinal entre elementos mantendo os mesmos isolados eletricamente.

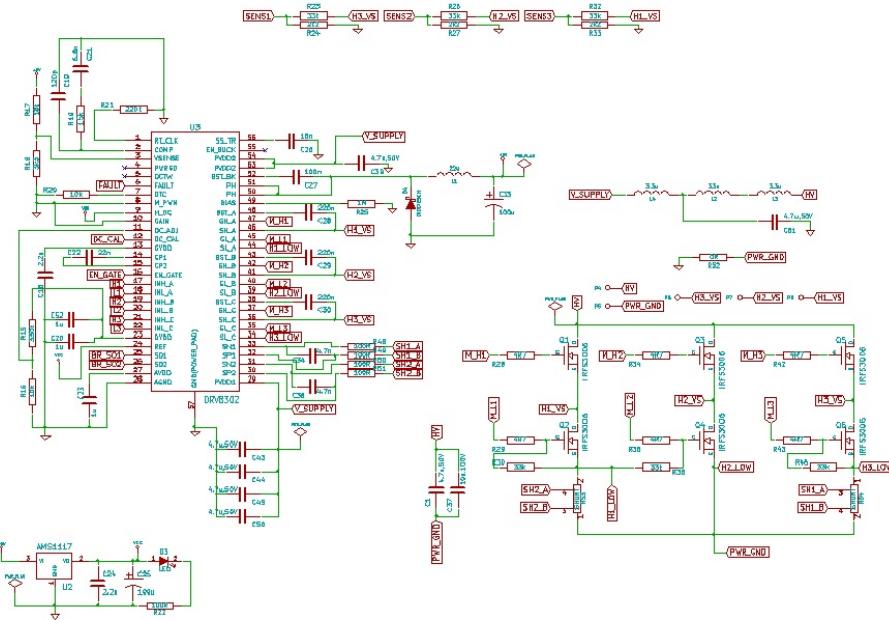


Figura 5.2: Esquema elétrico da camada de Potência

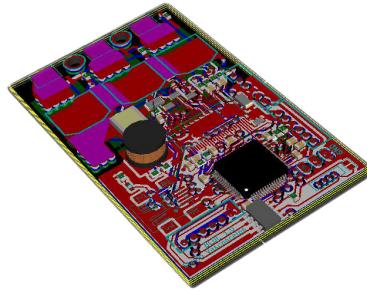


Figura 5.3: Modelo CAD da placa de circuito impresso

Os circuitos opto-isolados são dispositivos que no seu interior contém uma fonte emissora de luz (LED) e um foto sensor de silício (fototransistor) figura 5.4, sensível às variações espectrais da fonte emissora. Assim, deste modo é possível a transferência de sinais mantendo os lados eletricamente isolados.

Por motivos técnicos não foi possível utilizar este *transceiver*. Em alter-

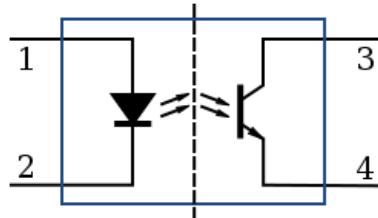


Figura 5.4: Circuito Opto-isolado

nativa optou-se por usar um *transceiver* idêntico mas sem ser opto-isolado, o SN65HVD230 da TI.

## 5.2 Descrição do Firmware

Neste secção iremos abordar alguns aspectos referentes ao *firmware* do sistema. É também apresentado um pequeno fluxograma do *firmware* (figura 5.5).

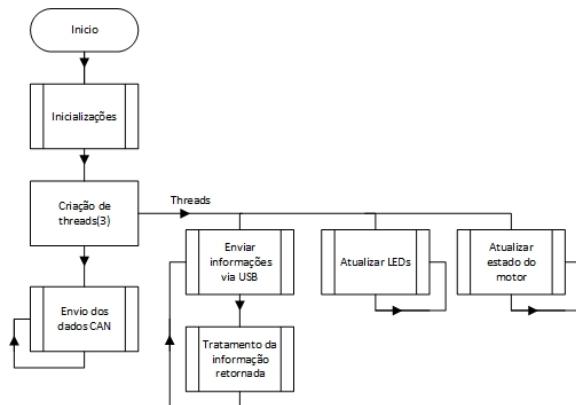


Figura 5.5: Fluxograma do firmware

### 5.2.1 BLDC

O *firmware* do controlador *Brushless DC* é desenvolvido usando o sistema operativo CHIBIOS que nos permite programar o ARM Cortex de forma

mais eficiente usando por exemplo *threads*. Este sistema operativo tem funções específicas para por exemplo configurar uma ADC ou um PWM facilitando assim a programação tornando assim a programação mais *user friendly*.

Quando o motor está a girar e o controlador se desliga por algum motivo, as comutações e a direção são detetáveis. É calculado o *duty-cycle* para que o motor gire à mesma rotação arrancando suavemente.

É possível executar travagens regenerativas, o que permite transformar a energia cinética gerada pelo motor em energia elétrica aumentando assim a sua eficiência.

Com este *firmware* é possível configurar alguns parâmetros de proteção tais como

- Tensão mínima de entrada;
- Tensão máxima de entrada;
- Corrente máxima do motor;
- Corrente máxima da entrada;
- Corrente máxima na travagem regenerativa;
- Máxima rotação para cada um dos sentidos.

### 5.2.2 CAN

Irá ser apresentado aqui o cálculo dos vários tempos de bit que variam dependendo do microcontrolador utilizado. Para calcular esses tempos e recorrendo-se ao *datasheet* do microcontrolador podemos obter esses tempos da seguinte forma.

$$BaudRate = \frac{1}{NominalBitTime} \quad (5.1)$$

$$NominalBitTime = 1 * t_q + t_{BS1} + t_{BS2} \quad (5.2)$$

Com

$$t_{BS1} = t_q * (BS1 + 1) \quad (5.3)$$

$$t_{BS2} = t_q * (BS2 + 1) \quad (5.4)$$

$$t_q = (prescaler + 1) * t_{PCLK} \quad (5.5)$$

Onde  $t_{PCLK}$  é o período do APB1. *Time quantum* ou  $t_q$ , é a unidade de medida do *nominal CAN bit time*, apresentado na figura 5.6. [6]

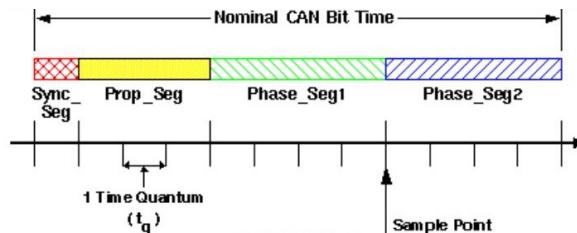


Figura 5.6: *Nominal CAN Bit Time*

Porém existe calculadoras *online* que fazem esse cálculo, para isso é necessário indicar apenas o frequência de *clock* do porto ao qual está ligado o periférico CAN, no caso APB1 que trabalha a 42Mhz.

Para 500Hz foi utilizado os seguintes parâmetros.

- *Prescaler*: 6;
- *BIT SEGMENT 2*: 8;
- *BIT SEGMENT 1*: 1.

É ainda necessário indicar outros parâmetros tais como

- Automatic Bus-Off Management;
- Automatic Wakeup Mode;
- Transmit FIFO Priority;
- Loop Back Mode.

Com estas configurações estaremos prontos para comunicar via CAN.

Neste pequeno excerto de código é onde indicamos as configurações necessárias para colocar o protocolo CAN a funcionar

---

```
static const CANConfig cancfg = {
    CAN_MCR_ABOM | CAN_MCR_AWUM | CAN_MCR_TXFP,
    CAN_BTR_LBKM | CAN_BTR SJW(0) | CAN_BTR_TS2(1) |
    CAN_BTR_TS1(8) | CAN_BTR_BRP(6);
```

---

É declarada esta variável de maneira a guardar as configurações para depois enviar por parâmetro na função *canStart*. Esta função, *canStart*, permite a inicialização do CAN com as configurações passadas por parâmetro.

Nesta variável indicamos que o periférico CAN irá trabalhar no modo *Automatic Bus-Off Management*, *Automatic Wakeup Mode*, *Transmit FIFO Priority*. Após a vírgula, as configurações recaem sobre o *baud rate*.

Para o envio de dados, o controlador fica à espera que seja informado via USB sobre que tipo a informação terá que transmitir e de que modo. Caso o controlador esteja no modo *debug*, toda a informação possível será enviada de forma periódica. No caso do modo *debug* esteja desativado a informação CAN será apenas enviada quando requerida pelo utilizador.

Quando recebe ordem para transmitir por exemplo a velocidade por CAN é feito o seguinte:

---

```
txmsg.data16 [0] = 0XAA00;
txmsg.data16 [1] = mcpwm_get_rpm();
canTransmit(&CAND1, CAN_ANY_MAILBOX, &txmsg, MS2ST(100));
```

---

No código acima, txmsg.data16 é um *array* em que na 1<sup>a</sup> linha é indicado o prefixo da mensagem. No caso será enviada a velocidade, o prefixo será AA00. Na 2<sup>a</sup> posição do array é colocada a velocidade em RPM. Após isso é iniciada a função canTransmit, onde o 1º parâmetro define que estamos a utilizar o CAN1. O 2º serve para definir a MAILBOX (no caso é utilizada qualquer uma que esteja livre). O 3º campo desta função é utilizado para inserir a mensagem a enviar, que no caso, seria algo como AA00XXXX. Por último, no 4º campo, é o *timeout* que no caso será algo como 100ms.

### 5.3 Descrição da aplicação

Foi implementada uma aplicação em QT (figura 5.7) permitindo a visualização de gráficos e valores numéricos relativos as variáveis associáveis ao controlo do motor. De referir que esta GUI não foi desenvolvida de raiz.

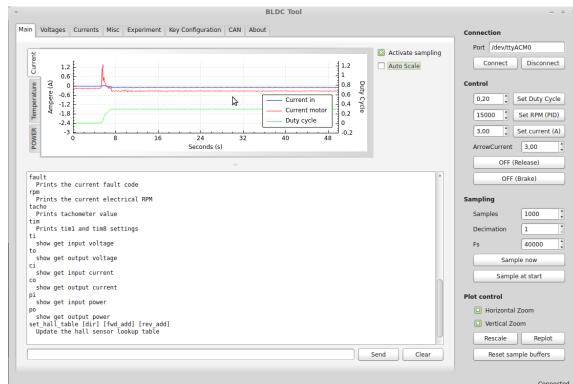


Figura 5.7: Separador *main* da aplicação

Nesta aplicação é possível:

- Rodar o motor a uma determinada rotação;
- Rodar o motor a uma determinada intensidade de corrente;
- Travar o motor de forma regenerativa;

- Parar o motor travando ou rodar livremente até parar;
- Obter gráficos das correntes e tensões nas 3 fases;
- Determinar a posição em graus do motor (rotor em relação ao estator);
- Definir dados a enviar através do barramento CAN.

É possível através da aplicação comunicar com o controlador *Brushless DC* através de um terminal. Podendo assim solicitar o envio de algumas informações tais como memória utilizada, listar todas as *threads* ou ainda ver a "quilometragem" do motor. As informações sobre o estado do motor podem ser transmitidas via CAN selecionando na aba "CAN" do software (figura 5.8).

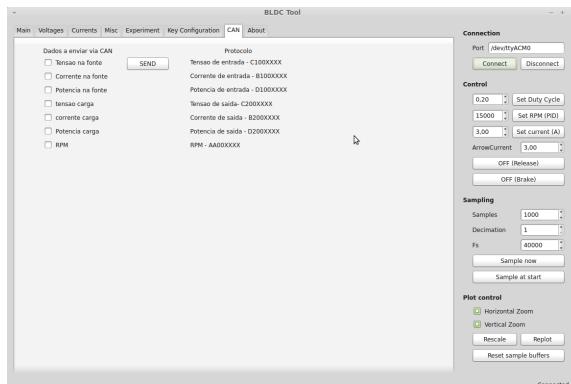


Figura 5.8: Aba CAN da GUI

### Main

Nesta aba 5.7 está disponível um terminal para comunicar com o controlador *Brushless DC*. É possível gerar alguns gráficos onde podemos ver em tempo real a corrente de entrada e saída bem como a sua potência.

Apesar de o software já estar preparado para representar o valor da temperatura apenas é possível para um determinado *profile* do controlador.

É possível observar do lado direito da janela uma série de comandos que podemos escolher para colocar o motor em movimento ou paragem.

Se simplesmente queremos que o motor rode basta fazer o *Connect*, situado no canto superior direito e confirmar a sua conexão no canto inferior direito. Após isso, basta usar as setas do teclado para escolher o sentido e o motor rodará a uma determinada velocidade pré-definida. Podemos ainda experimentar a travagem regenerativa carregando na seta para cima (é desaconselhado utilizar este modo numa fonte de bancada).

### ***Voltages/Currents***

Nestas secções é possível tirar amostras totalmente configuráveis sobre a tensão e corrente tanto no arranque ou já com o motor em andamento. Após a aquisição é possível ver em cada gráfico a tensão/corrente de cada fase.

### ***MISC***

Aqui é possível obter a velocidade e ainda a posição do rotor face ao estator.

### ***Key Configuration***

Nesta aba podemos definir as teclas numéricas de 0 a 9 para diferentes velocidades do motor bem como o sentido (valor positivo corresponde ao sentido horário, já o valor negativo representa o sentido anti-horário).

### ***CAN***

Nesta secção podemos definir a informação que o controlador *Brushless DC* envia via CAN. Podemos escolher varias informações tais como a corrente de entrada bem como a velocidade ou ainda a potência consumida pelo motor. Temos ainda o modo *debug* que permite o envio, de forma periódica, de todas as mensagens disponíveis.

***About***

Por último, nesta aba é apresentada alguma informação sobre os criadores da aplicação.

Esta página foi intencionalmente deixada em branco.

# Capítulo 6

## Resultados

Neste capítulo irá ser apresentado alguns resultados do trabalho realizado, nomeadamente alguns *printscreens*.

A primeira figura, 6.1, demonstra as formas de onda da tensão no arranque do motor *Brushless DC*.

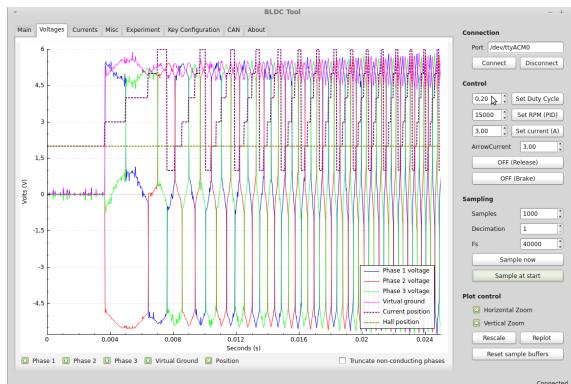


Figura 6.1: Tensão no arranque

Deve-se ignorar a reta referente ao sensor de efeito de HALL uma vez que, o ensaio, não foi utilizado qualquer tipo de sensor de posicionamento.

A próxima figura, 6.2, ilustra as formas de onda da corrente igualmente no arranque do motor *Brushless DC*.

Por fim, para demonstrar a comunicação a funcionar, foi utilizado um

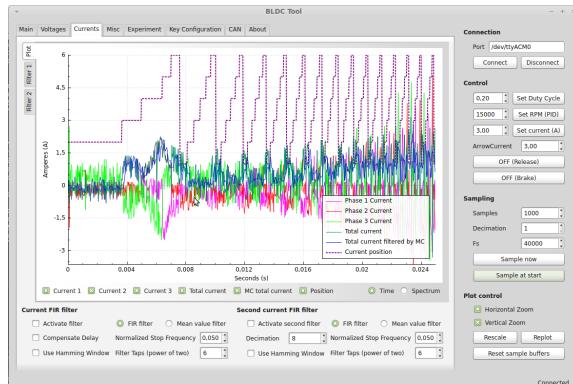


Figura 6.2: Corrente no arranque

conversor CAN para USB da ESD-Electronics. Na figura, 6.3, pode-se observar a janela do *software CANReal*, disponibilizado pela ESD-Electronics, que permite visualizar as mensagens recebidas pelo controlador de motores *Brushless DC*. É de salientar que, para efeito de testes, como estamos a lidar com baixas correntes, os valores obtidos, da corrente e da potência, estão numa escala de  $\times 10^{-3}$ . Já na figura 6.4 temos a aplicação a obter os dados de modo a termos um termo de comparação. Relembro o protocolo de comunicação.

- RPM - AA00XXXX
- Corrente de entrada - B100XXXX
- Corrente de saída - B200XXXX
- Tensão de entrada - C100XXXX
- Tensão de saída - C200XXXX
- Potencia de entrada - D100XXXX
- Potencia de saída - D200XXXX

Os caracteres do protocolo de comunicação são em hexadecimal. Os caracteres "X" são valores variáveis dependendo da informação a transmitir.

## Capítulo 6

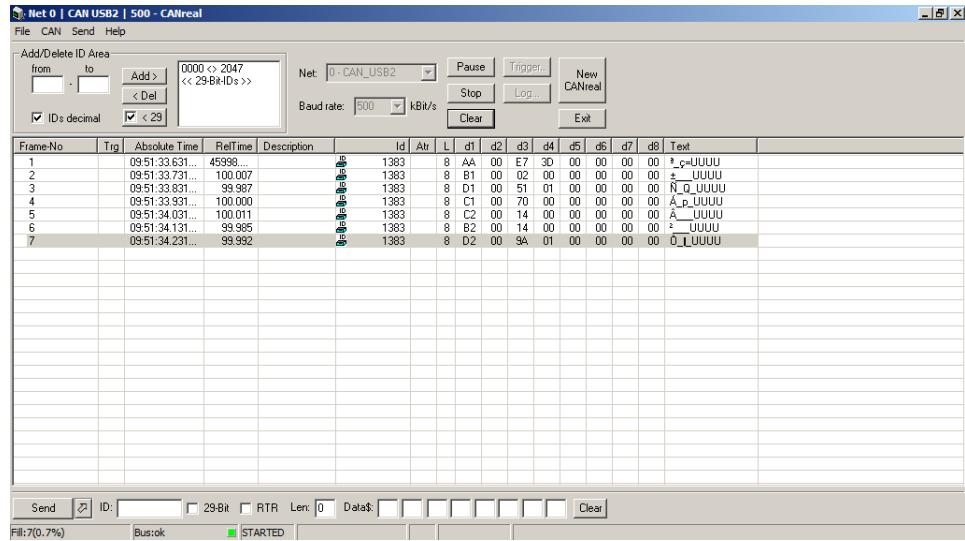


Figura 6.3: Software CANReal

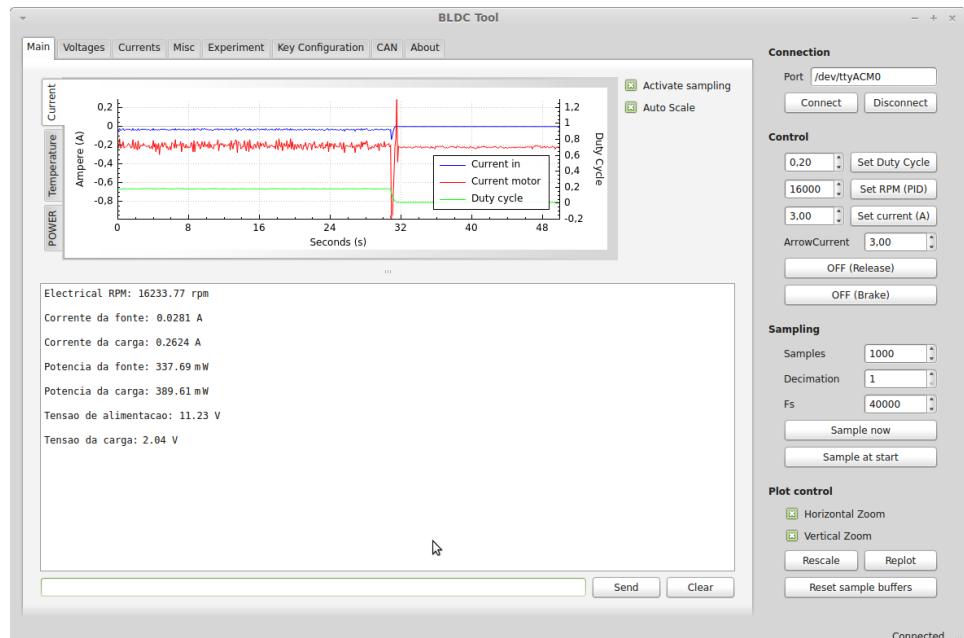


Figura 6.4: GUI BLDC

Esta página foi intencionalmente deixada em branco.

## Capítulo 7

# Conclusão e Trabalho Futuro

Após o término deste projeto e alguma reflexão sobre o trabalho desenvolvido, chega-se à conclusão que são claros os benefícios gerados pela utilização de motores *Brushless DC* em diversas aplicações em que tipos de motores são tipicamente empregados.

Foram estudadas varias alternativas open-source de um controlador *Brushless DC* e, depois de analisar caso a caso, escolheu-se a opção do Sr. Benjamin Vedder pois reunia os requisitos mínimos para a elaboração deste projeto.

Neste projeto foram apresentadas as principais vantagens da utilização da comunicação CAN num ambiente vulnerável a ruído eletromagnético como é o caso. Com esta comunicação é possível adicionar módulos à rede sem qualquer alteração do software dos dispositivos já existentes na rede tornando assim esta rede muito versátil.

Com isto temos total controlo sobre o motor ligado a este controlador, desde a velocidade à corrente consumida passando pela potência quer do lado da carga quer pela fonte.

Futuramente prevê-se a implementação do sistema FOC, que é uma técnica de controlo de forma a maximizar a produção de torque, para motores *sensorless* que permite que motores de alto torque possam operar de

forma mais suave podendo gerar o seu torque máximo numa velocidade próxima de zero. Com este sistema é possível ainda obter acelerações e desacelerações ainda mais rápidas. É possível adicionar ainda a comunicação I2C ou UART utilizando os pinos disponíveis para o servo motor. Outro melhoramento seria a possibilidade de guardar dados obtidos num ficheiro de texto.

# Bibliografia

- [1] Esden. Open-bldc project, 2012.
- [2] P. Esden-Tempski. Open-bldc v0.3 hardware based closed loop control, acedido em 16 de abril de 2014, 2010.
- [3] M. Fernandes. Modelação e controlo de um motor brushless pdi -. Technical report, FEUP, 2012.
- [4] M. Fernandes. Modelação e controlo de motores dc brushless, 2013.
- [5] Nicolas. Um passo em direção à miniaturização dos drones. 2013.
- [6] B. Semiconductors. The configuration of the can bit timing, 2010.
- [7] D. Strother. Brushless dc motor controller board, 2008.
- [8] Vedder. A custom bldc motor controller, 2014.
- [9] Wikipedia. Qt (software), 2014.

Esta página foi intencionalmente deixada em branco.

## Apêndice A

### Motores *Brushless*

Os motores de corrente contínua sem escovas ou motores *brushless* oferecem diversas vantagens face aos tradicionais motores DC tais como:

- Baixo nível de ruído;
- Vida útil mais longa (devido a ausência de desgastes das escovas);
- Baixa Interferência Eletromagnética EMI;
- Alta fiabilidade;
- Alta rotação possível.

A desvantagem principal do motor *brushless* é necessitarem de um circuito (*driver*) normalmente mais caro e mais complexo, para oferecer o mesmo tipo de controlo variável que os motores com escovas.

Para saber a posição do rotor existem várias maneiras. Pode ser utilizado codificadores rotativos (*rotary encoders*) juntamente com seus controladores é possível determinar exatamente o ângulo em que o rotor está. Outra alternativa é usar um sensor Hall, este é colocado numa determinada posição para que o sensor determine onde está o pólo Norte ou o pólo sul. O sensor Hall transmitirá então essa informação para o controlador do motor.

## A.1 Tipos de motores BLDC

Atualmente existem 2 tipos de motores *brushless*. *Inrunner* e *outrunner* são tipos de motores brushless que obviamente proporcionam *performance* diferentes.

### A.1.1 Movimento

Essa é a característica mais visível que nos permite a diferenciar o *inrunner* do *outrunner*.

No caso dos *outrunner* o corpo principal do motor gira junto com o eixo. Na extremidade do motor há uma parte separada do corpo principal que é fixa e também serve como montante para a fixação de todo o conjunto.

Já nos *inrunner* o corpo do motor é fixo e apenas o eixo gira.



Figura A.1: Exemplo de motor brushless *inrunner* (esquerda) *outrunner* (direita)

### A.1.2 Ventilação

Nos *outrunner* ventilação é facilitada por ter mais entradas de ar. Nos *inrunner*: a ventilação é menor por ser mais fechado.

Para reforçar essas características podemos também considerar o tipo de instalação, onde os *outrunners* ficam mais expostos ao fluxo de ar, ao contrário dos *inrunners* que geralmente são instalados internamente onde o local é menos exposto a fluxos de ar, algo que pode ser minimizado com um esquema de fluxo de ar.

Outro aspecto importante é como ocorre o aquecimento. No *inrunner* o aquecimento ocorre na parte mais externa e no *outrunner* o aquecimento é na parte mais interna.

### A.1.3 Desempenho

Essa é a principal e mais importante diferença entre os *inrunners* e *outrunners*. Esse é o aspecto mais relevante a ser considerado quando se escolhe o tipo de motor *Brushless DC*. O *outrunner* produz menor KV, maior torque e é menos eficiente, ou seja, gira mais lento, produz mais força e aproveita menos a energia. Já nos *inrunners*, produz maior KV, ou seja, menor torque e é mais eficiente, o que significa que gira mais rápido, produz menos força e aproveita mais a energia.

Esta página foi intencionalmente deixada em branco.

## Apêndice B

# Protocolo CAN

Como foi referenciado no capítulo 3 foi utilizado este protocolo de comunicação.

### B.1 Um pouco de história

Protocolo CAN foi desenvolvido por Robert Bosch em 1986 para aplicação na indústria automóvel, com o objetivo de simplificar os complexos sistemas de fios em veículos com sistemas de controlo compostos por múltiplos micro-controladores para gestão do motor, sistema ABS, controlo da suspensão, etc.

A aplicação da tecnologia CAN, para partilha de dados e controlo em tempo real, tem vindo a tornar-se cada vez mais popular. Ao longo dos anos, o protocolo CAN evoluiu de aplicações dedicadas à indústria automóvel para outras de uso industrial. CAN é suprimir a necessidade de sistemas complexos de fios substituindo-os por um simples cabo. É também considerado uma solução para implementar comunicação em rede de uma forma simples, barata e robusta, nomeadamente ao ruído eletromagnético.

## B.2 Características do CAN

CAN é um protocolo de comunicações série, que permite controlo distribuído em tempo real, com elevado nível de segurança. É um sistema em barramento com capacidades multi-mestre, isto é, vários nós podem pedir acesso ao meio de transmissão em simultâneo. Este protocolo comporta também o conceito de *multicast*, isto é, permite que uma mensagem seja transmitida a um conjunto de receptores simultaneamente.

Nas redes CAN não existe o endereçamento dos destinatários no sentido convencional, em vez disso são transmitidas mensagens que possuem um determinado identificador. Assim, um emissor envia uma mensagem a todos os nós CAN e cada um por seu lado decide, com base no identificador recebido, se deve ou não processar a mensagem. O identificador determina também a prioridade intrínseca da mensagem, ao competir com outras pelo acesso ao barramento.

A taxa máxima de transmissão especificada é de 1 Mbit/s, correspondendo este valor a sistemas com comprimento de barramento até 40 m. Para distâncias superiores a taxa de transmissão, recomendada, diminui. Alguns dos valores recomendados são: 50 Kbit/s para distâncias até 1 Km e 125 Kbit/s para distâncias até 500 m. Se a distância do barramento for superior a 1 Km pode ser necessária a utilização de dispositivos repetidores (*repeater*) ou ponte (*bridge*).

O protocolo CAN permite flexibilidade uma vez que podem ser adicionados novos nós a uma rede CAN sem requerer alterações do software ou hardware dos restantes nós.

## B.3 Método de Endereçamento

Quando são transmitidos dados utilizando o CAN, não existe endereço fonte ou destino numa mensagem. Os identificadores únicos, das mensagens, ser-

vem para caracterizar o conteúdo da mensagem (ex. RPM ou corrente do motor) sendo da competência de cada nó da rede decidir se a mensagem é ou não válida, para esse nó, realizando para isso um teste de aceitação ao identificador da mesma, este teste é designado por filtragem.

Outra característica importante do identificador, para além de definir o conteúdo da mensagem, é a de este estabelecer, também, a prioridade da mensagem. Isto é importante para a atribuição do barramento quando várias estações competem pelo acesso ao barramento.

O processo de transmissão e receção de mensagens CAN, ilustrado na figura B.1, consiste no seguinte: se a unidade central de processamento, *Central Processing Unit* (CPU), de um dos nós desejar enviar uma mensagem para um ou mais nós da rede, este "passa" os dados a serem transmitidos e o respetivo identificador para o controlador CAN ("Preparar") desse nó. Isto é tudo o que o CPU necessita realizar para iniciar a transferência de informação. A mensagem é então composta e transmitida pelo controlador CAN. Logo que o controlador consiga acesso ao barramento ("Enviar mensagem") todos os outros nós na rede CAN tornam-se receptores da mesma ("Receber mensagem"). Cada estação na rede CAN, tendo recebido corretamente uma mensagem, realiza um teste de aceitação para determinar se os dados recebidos são ou não relevantes para essa estação ("Seleção"). Se os dados tiverem significado são processados ("Aceite"), caso contrário são rejeitados ("Não Aceite").

## B.4 Formato da mensagem

Originalmente as mensagens (*frames*) do protocolo CAN possuíam um identificador de 11 *bits*. Com a especificação 2.0 foi definido um novo formato, o qual se baseia num identificador de 29 *bits* de comprimento, aumentando consideravelmente o número possível de identificadores únicos. Este novo formato é opcional, o que significa que as mensagens com identificadores de

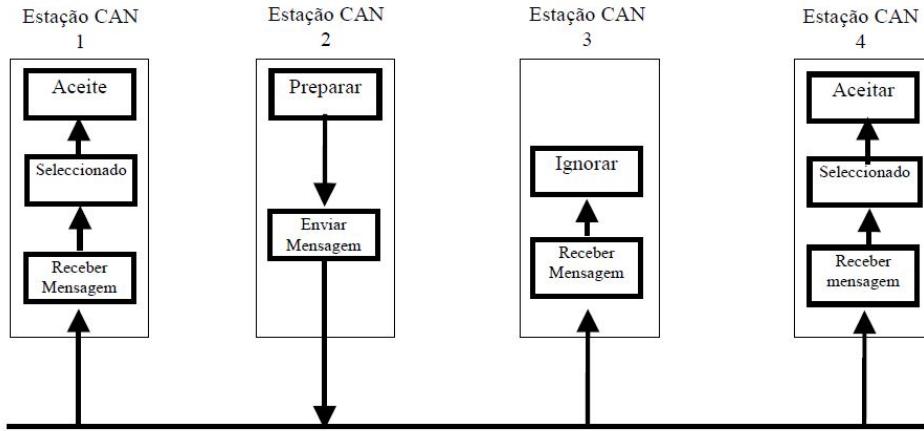


Figura B.1: Método de endereçamento

11 bits continuarão a ser as mensagens *standard*. De acordo com a especificação 2.0 as *frames* com 11 bits identificadores designam-se por *frames standard* e as *frames* com 29 bits identificadores designam-se por *frames estendidas*.

A Figura B.2 ilustra os diferentes tipos de formato possíveis, onde se pode verificar que em todos os formatos, as mensagens começam com o bit de início de frame, *start of frame* (SOF), seguido dos bits do identificador. Com o identificador existem um ou três bits de controlo (RTR, SRR e IDE), no campo de arbitragem. Estes bits definem quando se trata de uma *frame standard* ou estendida e quando se trata de uma *frame* de dados ou remota.

O significado dos três bits de controlo é o seguinte:

- O bit *Remote Transmit Request* (RTR), distingue entre *frames* de dados e remotas. Em *frames* de dados este bit é "dominante", sendo "recessivo" em *frames* remotas.
- O bit *Substitute Remote Request* (SRR), é um bit "recessivo". Este bit é transmitido em *frames* no formato estendido na posição que o bit RTR ocupa em *frames standard*.

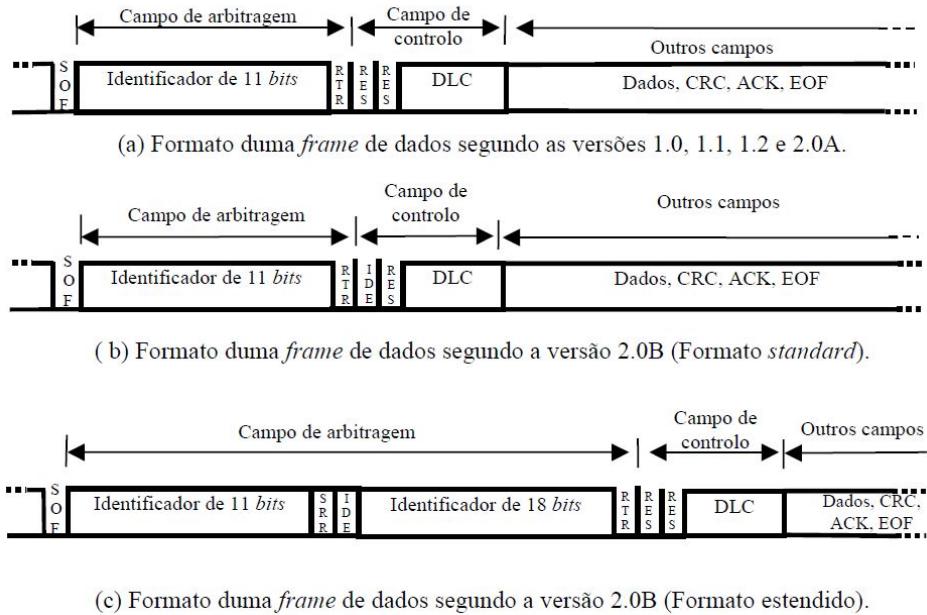


Figura B.2: Método de endereçamento

- O bit *Identifier Extension* (IDE), faz a distinção entre *frames standard* e estendidas. Em *frames standard* este bit é ”dominante”, enquanto que em *frames estendidas* é ”recessivo”.

Assim, num sistema onde vários nós iniciem a transmissão simultânea de *frames* com o mesmo identificador, as seguintes regras são aplicadas: *frames* de dados possuem maior prioridade do que *frames* remotas, e *frames standard* possuem maior prioridade do que *frames* estendidas. Isto significa que uma *frame standard* remota ”vence”, no processo de arbitragem, uma *frame* de dados estendida, se os 11 bits mais significativos do identificador forem iguais. No entanto, em sistemas onde tenham que coexistir os dois formatos é necessário ter alguns cuidados em termos de implementação.

O tempo nominal do bit,  $t_q$ , é definido como sendo a duração de um bit, correspondendo ao inverso da taxa nominal de transmissão, número de bit/s transmitidos, de um emissor ideal na ausência de ressincronização [4]. Os

segmentos que constituem o tempo de bit são ilustrados na Figura B.2 .

## B.5 O Nível Físico

O nível físico é responsável pela transferência de *bits* entre os diferentes nós de uma rede CAN; este define o modo como os sinais são transmitidos, lidando, por isso, com parâmetros como temporização, codificação e sincronização das sequências de *bits* a serem transmitidos.

Com o CAN é possível utilizar diversos meios físicos, tais como: par de fios entrelaçados, fibra óptica, rádio frequência, etc. Actualmente, a maioria das aplicações utiliza um barramento diferencial a dois fios.

## B.6 Transmissão diferencial

No CAN os sinais são transmitidos utilizando tensões diferenciais, derivando daí muita da imunidade ao ruído e tolerância a falhas que o caracterizam. As duas linhas de sinal são designadas por "CAN H" e "CAN L". Um "0" corresponde ao sinal CAN H superior ao CAN L e como tal designado por bit "dominante". A situação contrária, CAN L superior a CAN H, corresponde a um bit "recessivo" ou "1" conforme ilustra a Figura B.3.

A utilização de diferenciais de tensão permite às redes CAN funcionar quando uma das linhas de sinal for danificada, ou em situações extremas de ruído. Recorrendo a um simples par entrelaçado, as entradas CAN diferenciais cancelam o ruído de forma efectiva.

## B.7 Tempo de *bit* e Sincronização

O tempo nominal do bit,  $t_q$ , é definido como sendo a duração de um *bit*, correspondendo ao inverso da taxa nominal de transmissão, número de *bit/s*

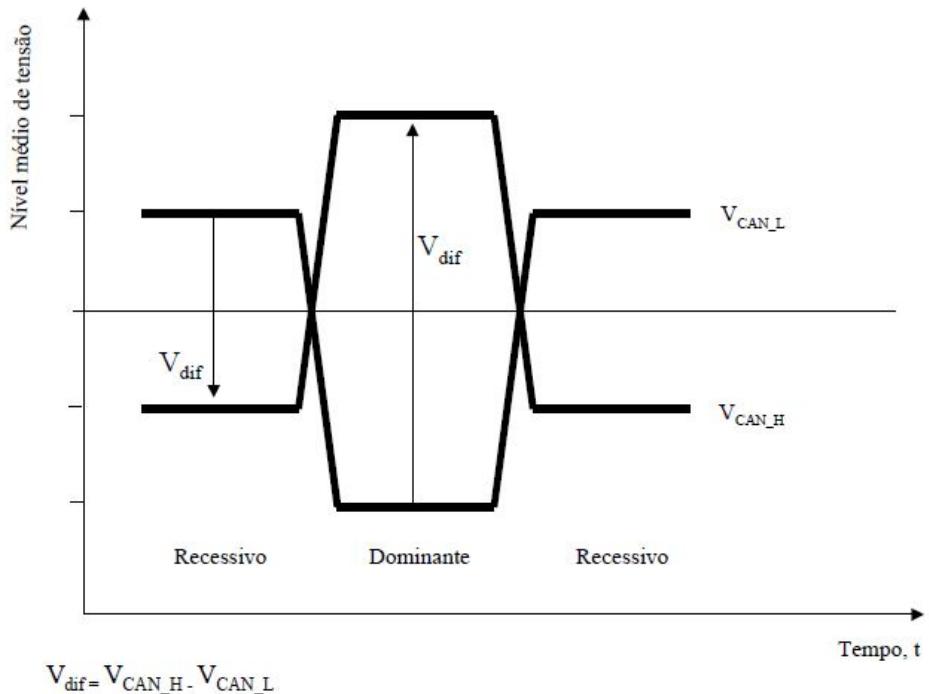


Figura B.3: Representação física de bit.

transmitidos, de um emissor ideal na ausência de ressincronização. Os segmentos que constituem o tempo de bit são ilustrados na Figura B.4.

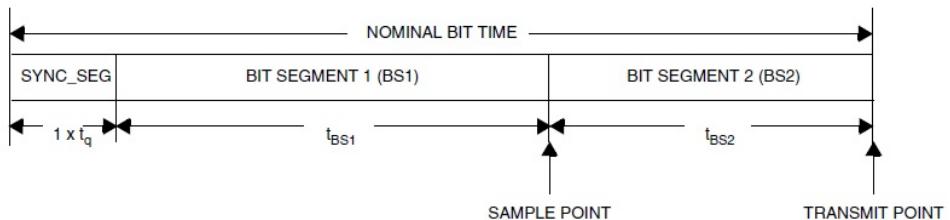


Figura B.4: Divisão do tempo de bit.

As funções de gestão do barramento executadas durante do tempo do *bit*, tais como o ajuste da sincronização de um nó, atraso de compensação da transmissão da rede e a posição dos pontos de amostragem, são definidas pela lógica programável do tempo de *bit* do controlador CAN. Os segmentos

que constituem o tempo bit são:

- Segmento de Sincronização, SYNC SEG.. Este segmento é usado para sincronizar os vários nós do barramento, sendo esperada uma transição durante o mesmo;
- Segmentos 1 e 2 dos *buffers* de fase, *BIT SEGMENT 1* e *BIT SEGMENT 2*. Estes segmentos são utilizados para compensar erros de fase nas transições. Estes segmentos podem ser alongados ou encurtados por ressincronização;
- Ponto de amostragem (*Sample Point*) é o instante no qual é lido e interpretado o nível do barramento como sendo o valor do respetivo bit. Ocorre no final do segmento *BIT SEGMENT 1*.

Na subsecção 5.2 é explicado como se procede ao calculo do *prescaler* e respetivos segmentos.

# **Apêndice C**

## **Soldadura**

Este anexo é dedicado à apresentação de detalhes referentes à soldadura dos componentes na placa de circuito impresso do controlador *Brushless DC*

### **C.1 Estação de Soldadura**

A estação de soldadura utilizada foi a disponível no Laboratório de Sistemas Autónomos . Dependendo do componente a soldar é necessário escolher a ponta apropriada para o efeito. Por exemplo para a soldar microcontrolador é boa prática usar um ponta em forma de colher (*Hoof Concave*), para a soldadura de outros componentes tais como resistências ou condensadores deve se utilizar uma ponta em forma de bico (*Conical Sharp*).

### **C.2 Fluxo e solda**

O fluxo é um fluido próprio para que propagação do calor imposto pelo ferro de soldar seja mais fácil. Este deve ser aplicado nas pernas dos componentes e nas ilhas na placa de circuito impresso para que a soldadura seja mais fácil.

A solda deve ser fina, pois estamos a lidar com componentes bastante pequenos, deve conter fluxo no seu interior para que seja em alguns casos

dispensável a adição de fluxo extra.

### C.3 Placa de Circuito Impresso

A placa de circuito impresso foi projetada em *KICAD* e realizada numa empresa especializada na execução de tarefas relacionadas com placa de circuito impresso e soldaduras. A placa de circuito impresso conta com 4 camadas sendo impossível realizar a placa de circuito impresso nas instalações do Laboratório de Sistemas Autónomos .

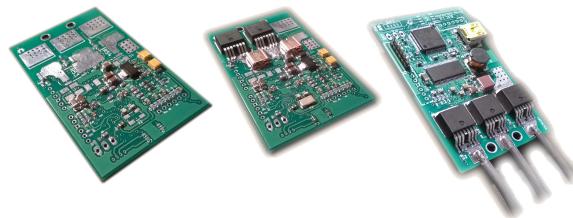


Figura C.1: Placa de Circuito impresso durante o processo de soldadura

## Apêndice D

# Esquemas Elétrico

Este anexo é dedicado à apresentação dos esquemas elétricos

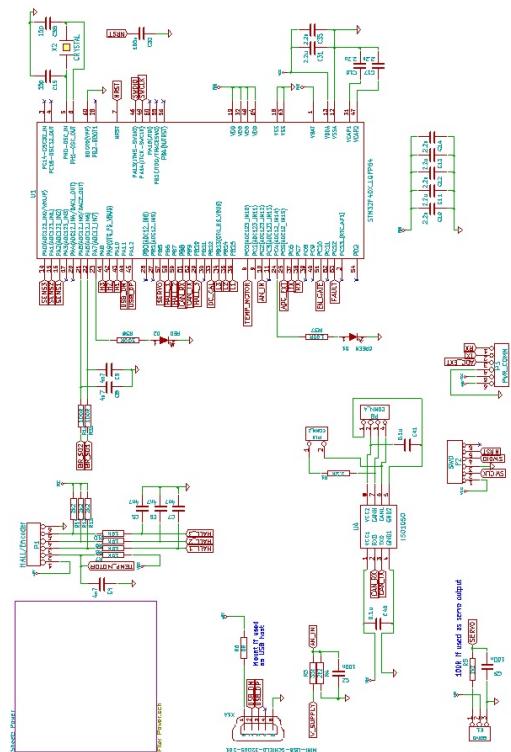


Figura D.1: Esquema elétrico da parte de Controlo

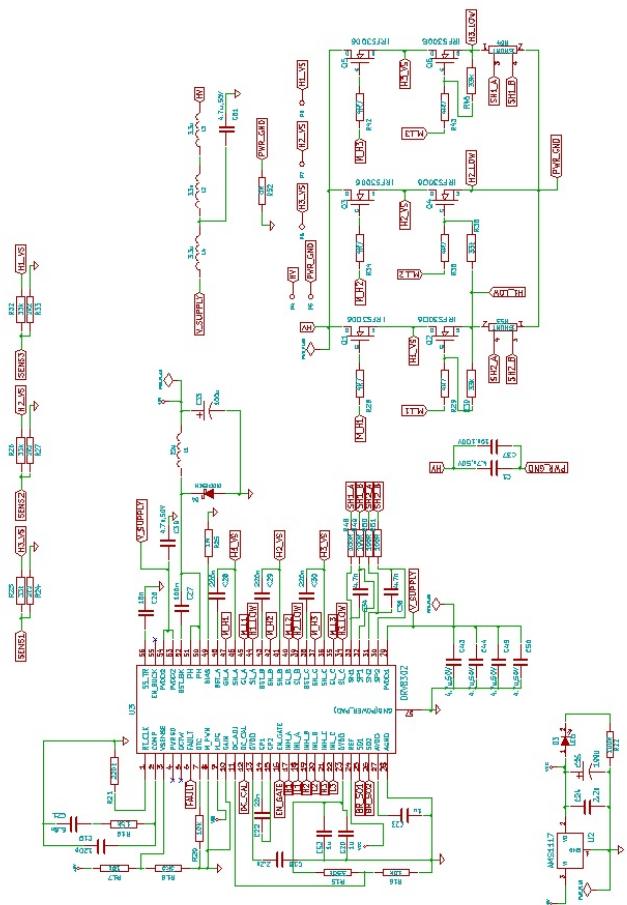


Figura D.2: Esquema elétrico da parte de Potência