Relatório de Projeto para Dissertação de Mestrado

Driver de Controle Laser

Orientando: Vitor Amadeu Souza
Orientador: Prof. DSc. Pablo Diniz Batista

Rio de Janeiro, RJ

Novembro, 2012

RESUMO

O objetivo do projeto ora proposto é desenvolver um driver de controle e potência para controle de um laser, onde um sistema composto de um microcontrolador dsPIC30F2020 procura controlar a corrente fornecida ao laser dentro de suas características, em uma faixa que varia de 0 a 200 mA através da técnica de PWM e Conversor DAC de acordo com os comandos parametrizados por um software desenvolvido no PC usando o ambiente de desenvolvimento Delphi. No PC, o mesmo programa irá plotar um gráfico com a curva de operação do diodo laser ao longo do seu funcionamento. A conexão entre a placa de controle e o PC é feita por uma porta USB, usando-se o conversor baseado na solução PIC18F14K50. Este projeto é parte do projeto de dissertação de mestrado profissional que tem como objetivo final o desenvolvimento de um Interferômetro de Michelson.

Palavras-Chave: Driver, Laser, Interferômetro, DAC, PWM.

ABSTRACT

The objective of the proposed project is to develop a driver and control power to control a laser, where a system composed of a microcontroller dsPIC30F2020 seeks to control the current supplied to the laser within its features in a range that varies from 0 to 200 mA through the technique of PWM and DAC converter according to the parameterized commands by a PC using software developed in the Delphi development environment. On the PC, the same program will plot a graph of the operation curve of the laser diode during its operation. The connection between the control board and PC is done through a USB port, using the converter based on the solution PIC18F14K50. This project is part of the project dissertation professional whose ultimate goal the development of a Michelson Interferometer.

Keywords: Driver, Laser, Interferometer, DAC, PWM.

SUMÁRIO

INTRODUÇÃO

O Interferômetro de Michelson é o tipo mais fundamental de interferômetro de dois feixes utilizado por Albert Michelson e Edward Morley em 1887 como instrumento para medir comprimentos de onda com grande precisão [1]. O principio básico de um interferômetro de Michelson é baseado no efeito causado devido à interferência entre dois feixes. Esse fenômeno gera uma intensidade de radiação na qual depende de frequência, polarização, fase e irradiância dos feixes que produz tal efeito. Nestes dispositivos se observar a interferência de duas ou mais ondas eletromagnéticas podendo-se verificar na saída do mesmo o resultado deste fenômeno. Este sistema pode ser utilizado em diversas aplicações em que eventos observados na natureza podem ser investigados tendo como princípio fundamental a interferência entre ondas eletromagnéticas. O mesmo consiste de um feixe de luz monocromático, normalmente um laser [02,03, 04] que passa por um espelho no qual a luz fica dividida em duas. Um dos principais elementos do interferômetro é o diodo laser, que tem sido utilizado como uma importante ferramenta em diversas aplicações, seja na área científica ou comercial [05]. Quando os dois componentes da luz são recombinados no detector, pode haver uma diferença de fase entre eles, já que estes podem ter percorrido caminhos diferentes. Eles interferem construtiva ou destrutivamente, dependendo da diferença de caminho. Se os dois caminhos percorridos forem iguais ou diferirem por um número inteiro de comprimento de onda, ocorre uma interferência construtiva e é registrado um sinal forte no detector. Se, no entanto, a diferença for um número inteiro e mais meio comprimento de onda, ocorre uma interferência destrutiva e é registrado um sinal muito fraco no detector [06].

Na emissão espontânea do laser, como ocorre por fontes de luz como um led ou lâmpada elétrica, os fótons são emitidos de forma aleatória, tanto em direção quanto no tempo, caracterizando a chamada luz incoerente, ou seja, as ondas dos fótons não apresentam uma relação de fase constante. Além disso, apresentam um grande faixa de comprimentos de onda (várias cores), sendo chamada de luz policromática enquanto que no laser há uma luz com alto grau de coerência obtendo-se assim feixes de onda paralelos e desta forma de grande potência [02, 03, 04]. A próxima figura apresenta um interferômetro de Michelson típico, onde este possui como parte central um *beamspliter* cuja função principal é fazer com que o feixe de luz seja dividido em dois outros.

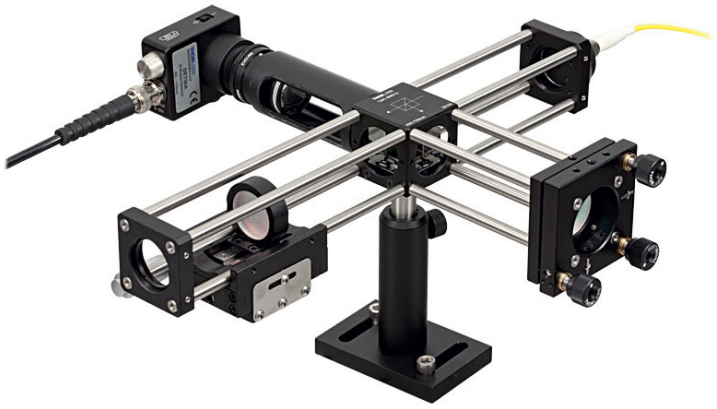

Figura 1: Interferômetro de Michelson típico [2].

O interferômetro é uma tecnologia usada para examinar e medir superfícies com alta-precisão. O interferograma é o cerne principal da interferometria, já que nele é registrado o sinal de interferência de dois feixes de luz que saem da mesma fonte e este leva detalhes sobre o perfil do objeto em análise e características deste material. Este equipamento é um dispositivo ótico que divide uma fonte de luz em dois feixes separados e recombina os mesmos de forma que o resultado dos fenômenos de interferência são registrados através de um interferograma. Normalmente, os interferômetros empregam um sistema onde um dos sinais é a reflexão do objeto em teste e o outro é refletido a partir de um espelho de referência. Os feixes recombinados formam luzes claras e escuras que compõem o sinal do interferograma. Este sinal é captado por um detector CCD para que se faça o processamento e mapeamento da imagem [7].

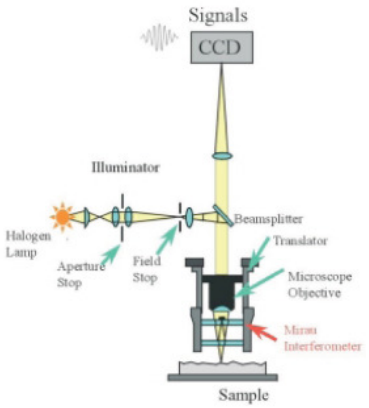

Figura 2: Estrutura de microscópio baseado em interferômetro [7].

O objetivo principal deste trabalho de pesquisa é desenvolver a instrumentação científica necessária para o desenvolvimento de um interferômetro de Michelson. Desta forma, os seguintes módulos serão desenvolvidos:

(a) Driver para o laser [5] [6] [7] [8] [9] [10];
(b) Detector de laser [11] [12];
(c) Programação de controlador digital [13] [14] [15] ;
(d) Software para plotagem da imagem de interferometria [16].

A descrição destes módulos é apresentada a seguir.

(a) Driver para o laser:

Consiste em uma fonte de corrente capaz de regular em regime dinâmico o laser e assim evitar que o mesmo venha a danificar, já que com a dissipação de potência sua resistência muda em regime dinâmico precisando assim de um controlador PID discreto que faça o ajuste necessário de forma a evitar a sua perda. Além disso, circuitos auxiliares farão a proteção em modo contínuo de forma a evitar que o laser venha a danificar caso a resposta do sistema não atenda sua velocidade de operação.

(b) Detector de laser:

Consistirá do circuito capaz de detectar o sinal e assim formar o pixel recebido. Possivelmente um detector Lock In será empregado nesta detecção.

(c) Programação de controlador digital:

Será usado um microcontrolador para este controle, onde a programação deste item será feita em conjunto com o driver e detector de laser de forma a testar o funcionamento do sistema.

(d) Software para plotagem da imagem de interferometria:

Programa que receberá a informação do item (c) de acordo com a detecção e fará a plotagem de forma a visualizar a superfície que está sendo analisada. A conexão entre o programa e o controlador poderá ser feita de diversas formas, com o uso de interface USB, Ethernet, Bluetooth e etc. Como ferramenta de programação será utilizado o Borland Delphi.

CAPÍTULO I
DRIVER DO TRANSMISSOR DE LASER

1. Laser

O laser utilizado na geração do feixe de luz em interferometria deve ser estável, monocromático e paralelo [2]. O laser a gás atende esta exigência, como exemplo os que são construídos por Hélio-Neon (He-Ne). Todavia, para este experimento foi adotado o diodo laser semicondutor de comprimento de onda específico, neste caso o CPS 180 [17] da empresa Thorlabs exibido na figura a seguir é um diodo laser de 635 nm de comprimento de onda e opera numa faixa de 1mW a 10 mW, oferecendo uma alternativa ao laser de He-Ne.

Figura 5: Laser CPS 180
Fonte: http://www.thorlabs.com/thorProduct.cfm?partNumber=CPS180

A construção típica de um diodo laser está apresentada na próxima figura.

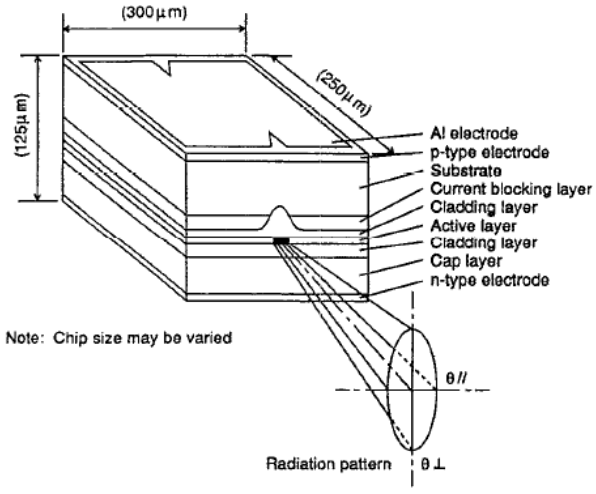

Figura 6: Construção típica de um diodo laser [8]

A emissão de luz laser é gerada pela injeção da corrente entre as camadas da região ativa *n* e *p*. Neste momento os elétrons se recombinam e emitem fótons onde o tamanho da onda é determinado pelo tipo de semicondutor utilizado. Uma grande variedade de diodos laser tem sido utilizados na região próxima a do infravermelho como os que constituídos pro GaAlAs. Tais dispositivos produzem saídas típicas de potência entre 5mW e 15mW e são comercialmente disponíveis no mercado [8].

Como resultado do progresso na área tecnologia para fabricação de semicondutores, hoje já estão disponíveis em forma comercial diodos laser com potência superior a 1 W. Em aplicações de alta velocidade, modulação ou controle de corrente estes dispositivos são importantes e tem sido utilizados em novas aplicações que requeiram processamento de imagens e comunicações em altas velocidades[18].

A próxima figura apresenta a região de operação em que ocorre a emissão do laser onde se observa que a potência desenvolvida cresce rapidamente com o aumento da corrente [19].

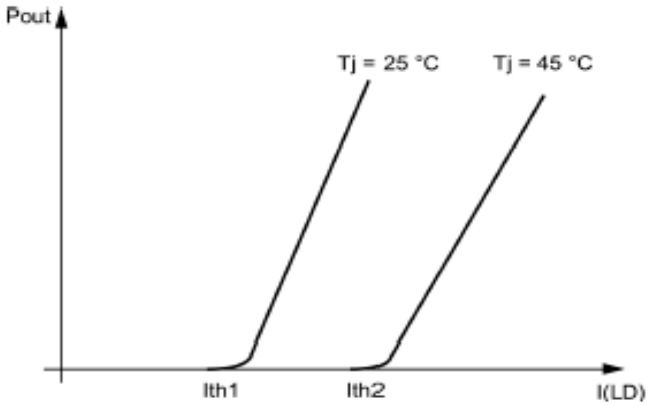

Figura 7: Resposta típica de um diodo laser
Fonte: http://www.thorlabs.com/thorProduct.cfm?partNumber=CPS180

Abaixo da corrente de limiar (I_{th}) o diodo laser emite luz incoerente (emissão espontânea), atuando como um LED (*Light Emmiting Diode*) comum. A partir deste ponto ele passa a emitir laser (emissão estimulada). A relação entre a potência óptica e a corrente no diodo é linear até a potência máxima definida pelo fabricante. Isto ocorre tanto no modo contínuo como no modo pulsado. Esta potência máxima não pode ser ultrapassada, sob pena de danificar o diodo. De acordo com a figura, observa-se que a corrente de limiar aumenta com a temperatura enquanto a potência óptica e a eficiência diminuem. Dados tais motivos, é essencial que os diodos laser sejam utilizados juntos com um *driver* que possa

atuar basicamente de dois modos, neste caso o ACC (*Automatic Current Control*) ou APC (*Automatic Power Control*) [38]. O circuito de controle também deve possuir um limitador, o qual desliga o dispositivo para que uma grande variação de temperatura não o danifique. O driver de corrente regula a corrente do laser comparando o retorno de tensão proporcional a corrente do laser [19].

2. Driver de Controle

O driver de controle é o hardware responsável por fazer a interface entre o computador e a placa de potência, fazendo o controle de corrente fornecido ao laser pela placa de potência assim como informando ao PC a faixa atual de consumo de corrente.

Os esquemas elétricos e layout que serão apresentados a seguir foram feitos com o auxílio do CAD de layouts e esquema elétrico Altium Designer®[20]. O diagrama de blocos do circuito de controle e potência está apresentado na próxima figura.

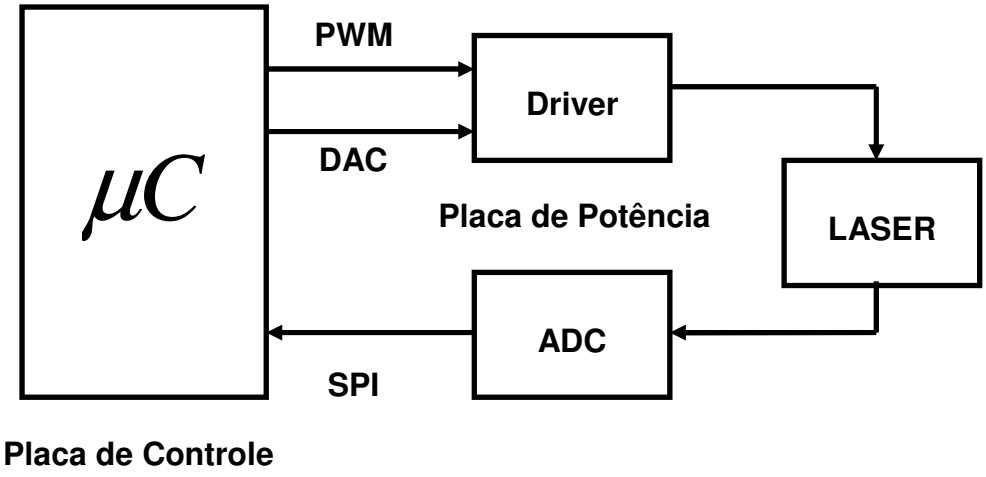

Figura 8: Diagrama de blocos do sistema microcontrolado
Fonte: Elaborado pelo autor

2.1 PWM

O PWM (Pulse Width Modulation – Modulação por Largura de Pulso) é uma forma de trabalho em que a frequência de funcionamento é constante, porém a largura do pulso, ou seja o ciclo ativo (duty cycle) pode alterar, permitindo desta forma com que o microcontrolador tenha controle de potência de cargas DC, como por exemplo motores,

lâmpadas, led e etc [21]. O gráfico a seguir é um exemplo de saída PWM em que o ciclo ativo está ajustado para 50%, obtendo-se assim metada da tensão média na carga.

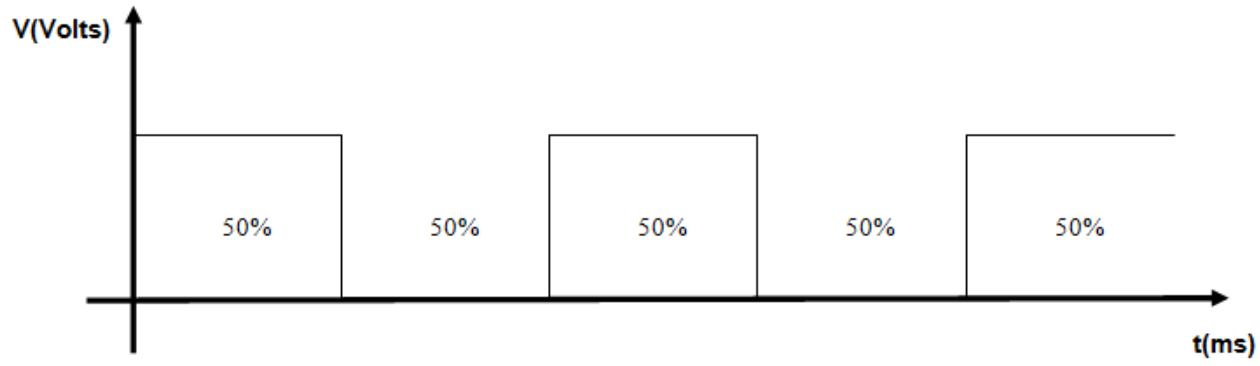

Figura 9: Saída PWM com ciclo ativo de 50%

Fonte: Elaborado pelo autor

O próximo gráfico apresenta uma saída PWM com ciclo ativo de 20%. Neste caso a tensão média fornecida a carga (V_m) será um quinto da tensão máxima.

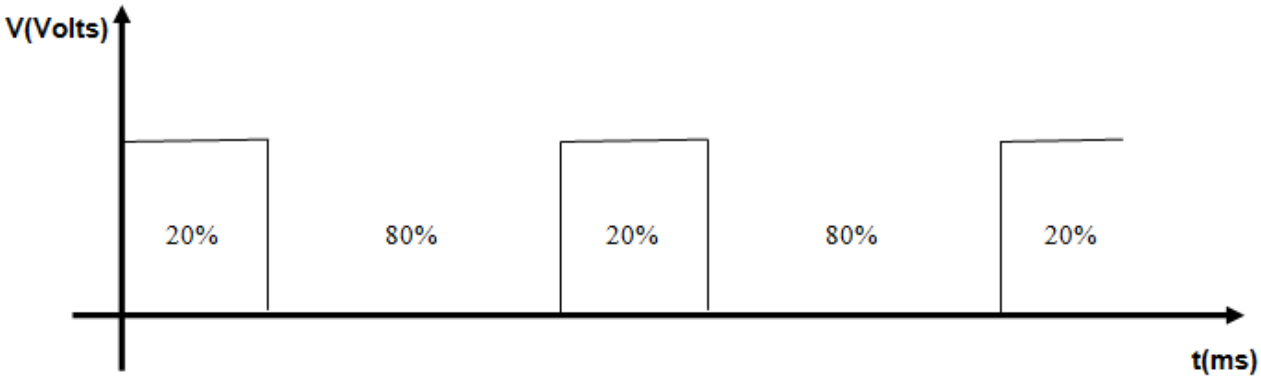

Figura 10: Saída PWM com ciclo ativo de 20%

Fonte: Elaborado pelo autor

2.2 Barramento SPI

O Barramento Periférico Serial ou SPI (Serial Peripheral Interface) é um protocolo de comunicação criado pela Motorola, que permite que haja comunicação em modo full-duplex entre dois dispositivos, ou seja, enquanto uma informação está sendo enviada, outra pode estar sendo recebida simultaneamente[22]. Esta interface é chamada de mestre-escravo

(master-slave), pois a comunicação sempre é inicializada por um dispositivo e o outro responde as solicitações feitas por este. Além disso, a comunicação é feita através de 4 pinos, chamados de MISO (Master Input Slave Output ou Entrada do Mestre e Saída do Escravo), MOSI (Master Output Slave Input ou Saída do Mestre e Entrada do Escravo), SCLK (Serial Clock) e $\overline{CS}$ (Chip Select ou Seleção de Chip). Na figura a seguir está apresentada a conexão física que há entre um mestre e um escravo de forma genérica.

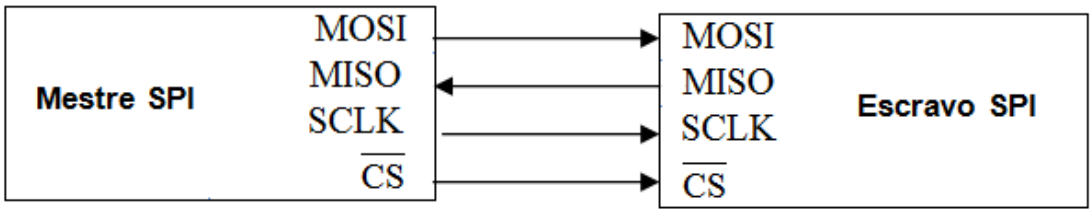

Figura 11: Diagrama de conexão SPI com um escravo
Fonte: Elaborado pelo autor

Através do pino $\overline{CS}$ podemos selecionar se o escravo ficará ativo ou não no barramento onde esta linha igual a 0 indica que o escravo está selecionado e em 1 que não está. Através da linha MOSI o mestre poderá enviar uma informação ao dispositivo SPI, já que esta linha é a *Master Output Slave Input.* A linha MISO permitirá receber informações do escravo, já que esta é a linha de *Master Input Slave Output*. Pela linha SCLK há como controlar a sincronização, ou seja, a velocidade em que a comunicação terá no barramento de dados, sendo esta gerada pelo Master. As setas usadas pelas linhas representam o fluxo de dados na comunicação SPI em que apenas a linha MISO é uma informação de entrada para o Master e o restante são pinos de saída. Diversos dispositivos podem ser conectados ao barramento (BUS) SPI, não estando limitado a apenas um escravo. Neste caso, o Master precisará dispor de mais pinos de seleção $\overline{CS}$ para escolher o dispositivo que ficará ativo no barramento para comunicação. Na próxima figura, podemos observar dois escravos conectados ao mestre para realizar a comunicação. Observe que os pinos de MOSI, MISO e SCLK estão interligados, tendo apenas a linha de seleção separada para cada um dos escravos, de forma a selecionar cada um deles no momento em que a comunicação for ser realizada.

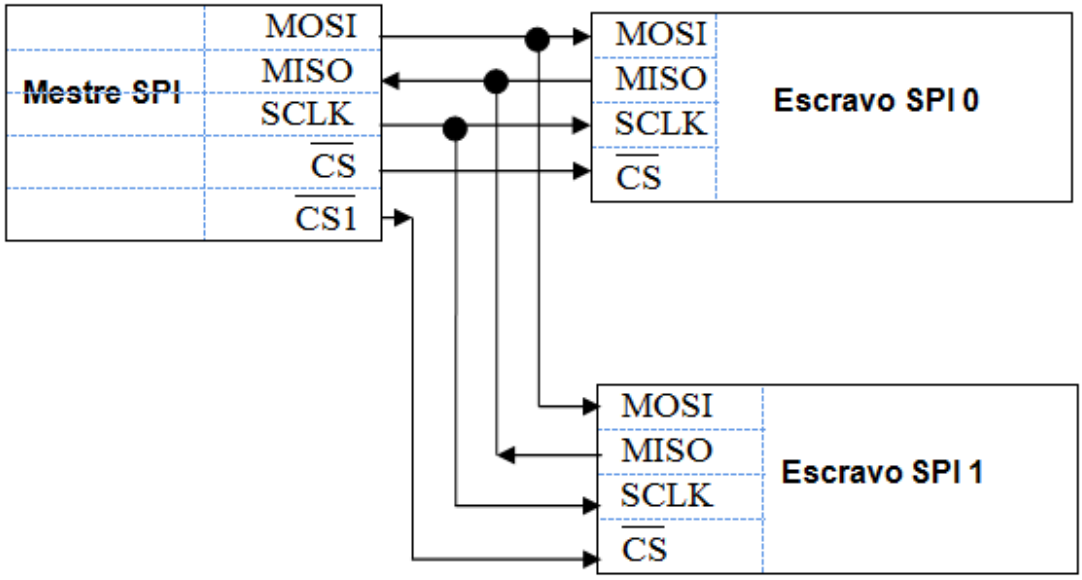

Figura 12: Diagrama de conexão SPI com mais escravos

Fonte: Elaborado pelo autor

A seleção neste caso funciona como habilitador do barramento, onde no momento em que um escravo estiver selecionado o outro não poderá ocupar o barramento e vice-versa, garantindo assim uma linha de comunicação direta entre o slave e o master. Esta mesma ideia pode ser aumentada de forma a colocar mais dispositivos ligados ao barramento SPI e obter assim altas taxas de comunicação [23].

2.3 Alimentação da placa

A alimentação da placa está sendo feita através de uma entrada DC onde nesta teremos as tensões disponíveis de +12V, -12V, +5V e GND. Na próxima figura está apresentado o bloco referente a alimentação do circuito.

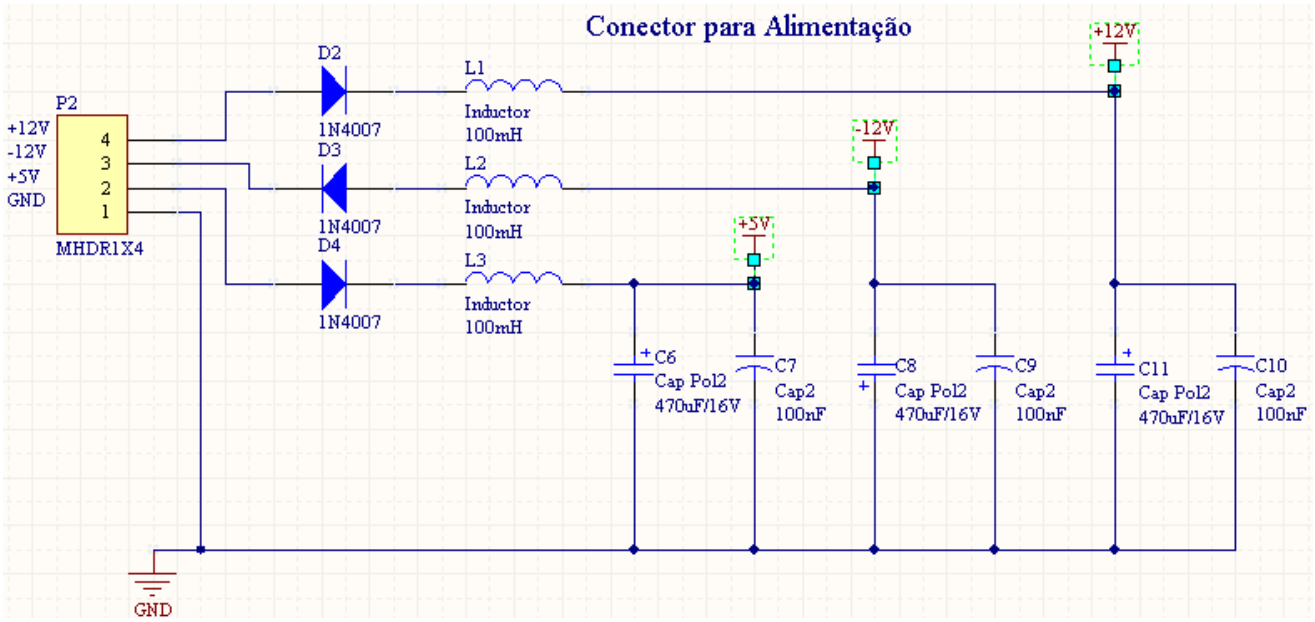

Figura 13: Esquema elétrico da parte de alimentação

Fonte: Elaborado pelo autor

Os indutores e capacitores foram utilizados para manter a tensão o mais estável possível [24], evitando variações bruscas de tensão em função da operação do driver. Os diodos de proteção D2, D3 e D4 são utilizados para que não ocorra inversão de sinais que leve a placa a ficar danificada.

2.4 Microcontrolador dsPIC30F2020

O driver de controle possui como cerne principal o dsPIC30F2020[25], cuja pinagem está apresentada na figura a seguir.

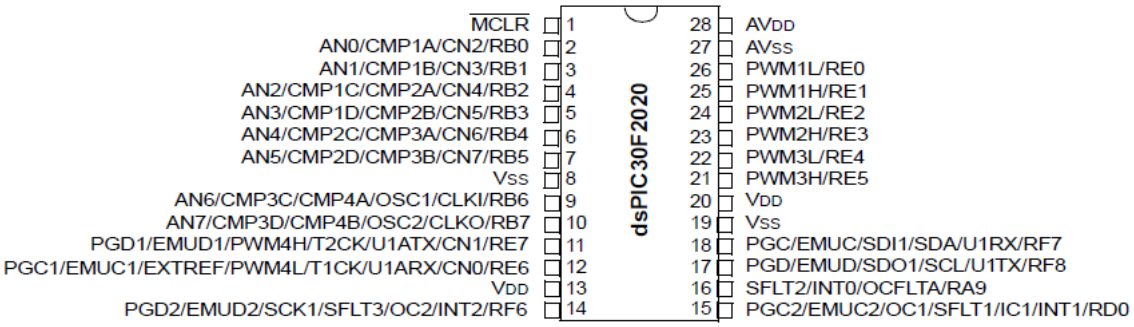

Figura 14: Pinagem do dsPIC30F2020

Fonte: http://ww1.microchip.com/downloads/en/devicedoc/70178c.pdf

As principais características deste microcontrolador estão apresentadas na tabela 1.

Características
Memória de Programa de 12 kB
Memória de Dados SRAM de 512B
Processamento de até 30 MIPS
Três timers de 16 bits
Comunicação SPI, I^2C e UART
Conversor AD de 10 bits
Duas saídas de PWM

Tabela 1: Características do dsPIC30F2020

Fonte: http://ww1.microchip.com/downloads/en/devicedoc/70178c.pdf

O dsPIC foi escolhido em função de apresentar uma ampla faixa de recursos disponíveis importantes para o projeto, como porta de comunicação serial RS232, PWM, SPI, conversor AD de 10 bits, CPU de 16 bits, processamento de até 30 MIPS e ser encontrado no mercado por um preço acessível. No esquema da placa driver de controle, a parte referente ao dsPIC é a apresentada na próxima figura.

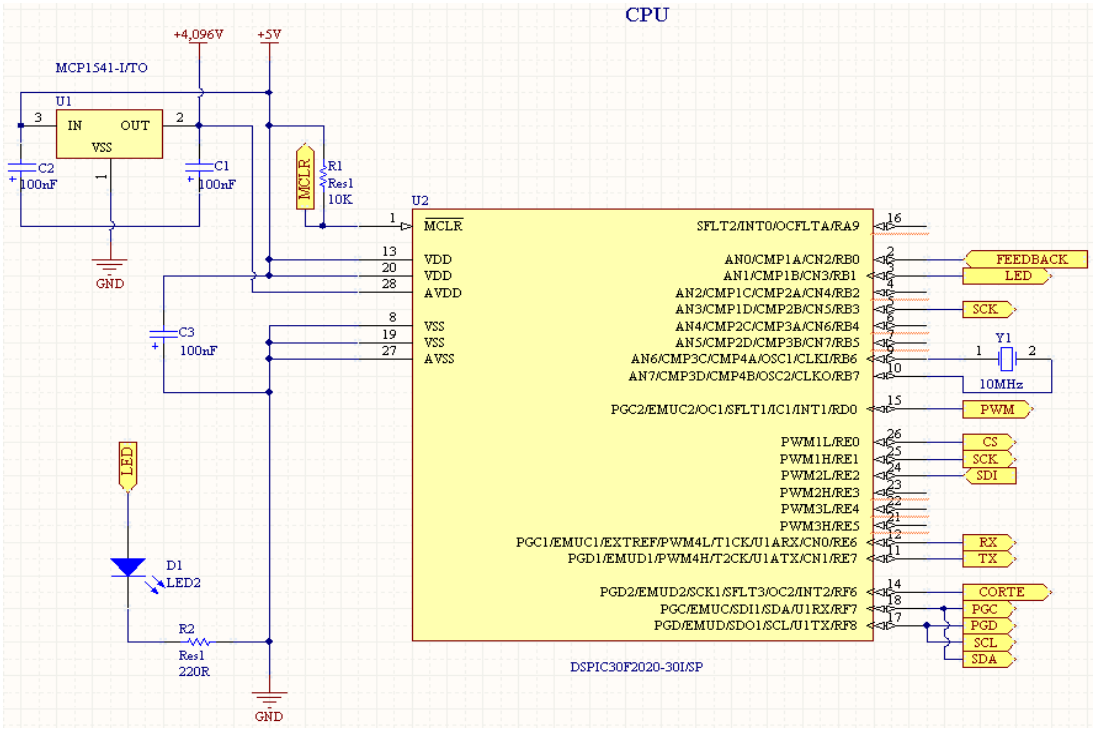

Figura 15: Esquema do microcontrolador dsPIC30F2020 na placa driver

Fonte: Elaborado pelo autor

2.5 Microcontrolador PIC18F14K50

O microcontrolador PIC18F14K50[26] foi utilizado para realizar a comunicação entre o PC e o dsPIC, funcionando como conversor serial para USB. A pinagem deste microcontrolador está apresentada na próxima figura.

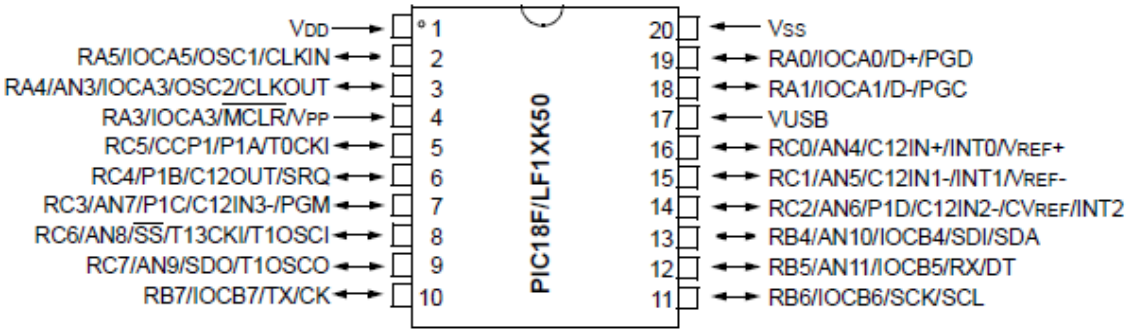

Figura 16: Pinagem do PIC18F14K50

Fonte: http://ww1.microchip.com/downloads/en/DeviceDoc/41350E.pdf

As principais características deste microcontrolador estão apresentadas na tabela a seguir.

Características
Memória de Programa de 16 kB
Memória de Dados SRAM de 768 B
Processamento de até 12 MIPS
Três timers de 16 bits e um de 8 bits
Comunicação SPI, I^2C e UART
Conversor AD de 10 bits
Uma saída de PWM

Tabela 2: Características do PIC18F14K50

Fonte: http://ww1.microchip.com/downloads/en/DeviceDoc/41350E.pdf

Este chip faz uso da solução fornecida pelo próprio fabricante, neste caso a Microchip para funcionar como chip conversor serial para USB [27]. O esquema elétrico que foi implementado no projeto para funcionar como conversor é o que está apresentado na figura abaixo.

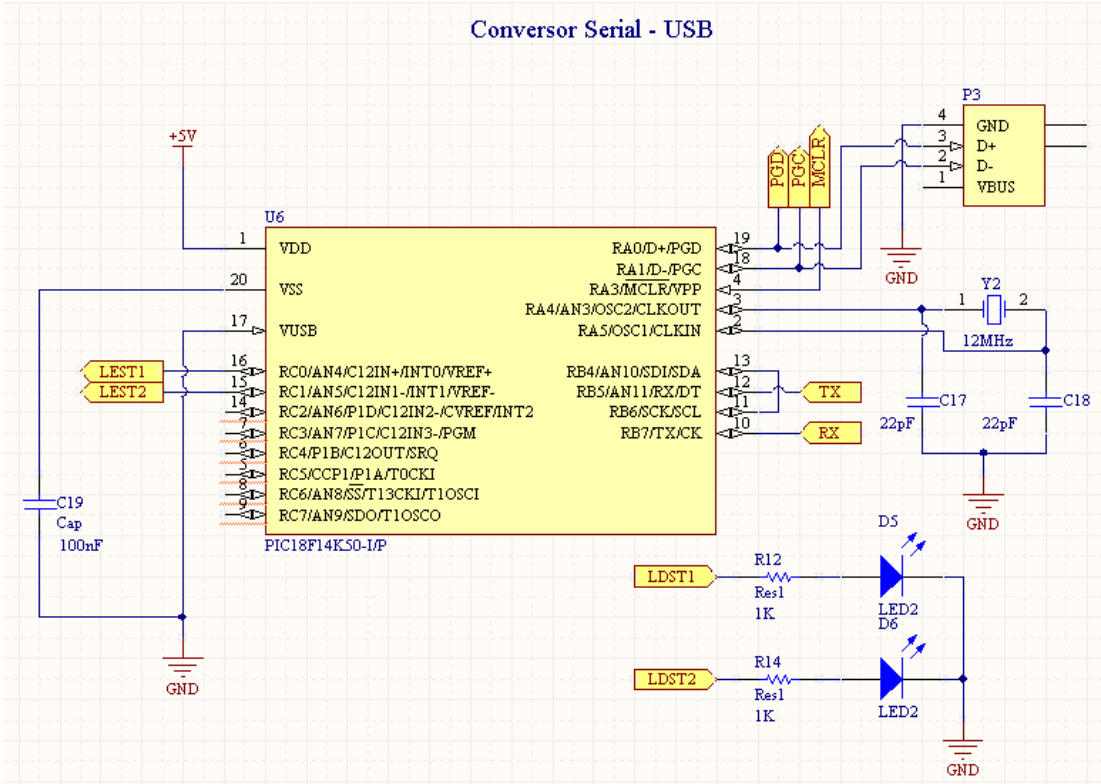

Figura 17: Esquema do conversor serial para USB implementado no projeto

Fonte: Elaborado pelo autor

Os pinos de TX e RX são os que fazem comunicação com o microcontrolador dsPIC30F2020, permitindo assim que o mesmo faça a conexão do tipo USB. Através do conector P3 será possível conectar o cabo USB que fará a comunicação com o PC através desta porta.

2.6 Comunicação RS232

A comunicação serial do tipo RS232 ainda é uma das mais usadas para troca de informações entre o computador e o mundo externo[28] [29]. A taxa de comunicação utilizada para comunicação entre o dsPIC e o conversor serial para USB será de 9600 bps, sem paridade com 1 stop-bit e um start-bit. Os níveis de comunicação usados pelo RS232 são diferentes do nível utilizado pelo dsPIC. Neste caso, normalmente é utilizado um ci chamado MAX232 que faz a conversão de TTL (que é o nível do PIC) para RS232 (que é o nível da entrada de dados do PC). Como neste projeto a comunicação será feita entre estes microcontroladores, não haverá necessidade de um circuito conversor de nível já que ambos funcionam a nível de tensão TTL. A seguir está apresentado um diagrama de blocos que ilustra a conexão RS232 entre um microcontrolador e a entrada de um computador.

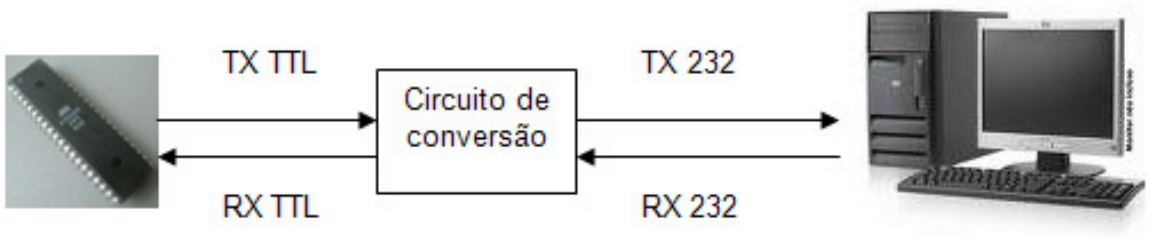

Figura 18: Conexão entre um microcontrolador e um PC por RS232

Fonte: Elaborado pelo autor

O CI MAX232 [30] pode ser usado como circuito de conversão neste circuito, onde sua função principal será de simplesmente converter o nível TTL no padrão 232 e vice-versa, já que ele também converte 232 em TTL. A comunicação do tipo RS232 é chamada de full-duplex, pois há uma linha somente de transmissão e outra somente de recepção. Isso quer dizer que enquanto o sistema está transmitindo um byte pela linha de TX, ele pode perfeitamente receber outro pela linha de RX. Na próxima tabela está apresentado os níveis de tensão da lógica TTL e RS232.

Nível	TTL	RS232
1	5 V	-3V a -15V
0	0 V	3V a 15V

Tabela 3: Níveis de tensão na lógica TTL e RS232

O nível 1 no padrão TTL está associado ao 5 V enquanto no RS232 está associado à tensão de –3V a –15V. Qualquer tensão nessa faixa (-3 até -15V) será entendida como 1 no receptor do sistema. Já o 0 lógico está associado 0V no padrão TTL enquanto no RS232 está associado de 3 até 15V.

O tempo de transmissão de 1 bit no barramento pode ser descoberto dividindo 1 pela taxa de comunicação (baud-rate) da rede. Desta forma, encontra-se o período ou tempo de 1 bit na comunicação. Neste caso, que está sendo utilizada uma taxa de 9600bps, o tempo do bit será dado de acordo com a próxima tabela.

T= 1 / 9600 = 104 us (aproximadamente)

Tabela 4: Tempo de transmissão de 1 bit na RS232

Toda comunicação serial começa com um bit de início que é chamado de start-bit. Logo em seguida temos os bits de dados, a começar pelo bit menos significativo (LSB) do byte a ser transmitido[31]. Opcionalmente, pode ser usada a paridade. Como neste caso este item não está incluído, a comunicação acaba com um bit de parada ou *stop-bit.* Como exemplo, o byte 10110010 é enviado para o PC a uma taxa de 9600bps sem paridade e 1 stop-bit conforme o gráfico apresentado a seguir. Como já visto através do cálculo anterior, o tempo de 1 bit é de aproximadamente de 104us. O gráfico a seguir está plotado na linha TTL e também na linha RS232.

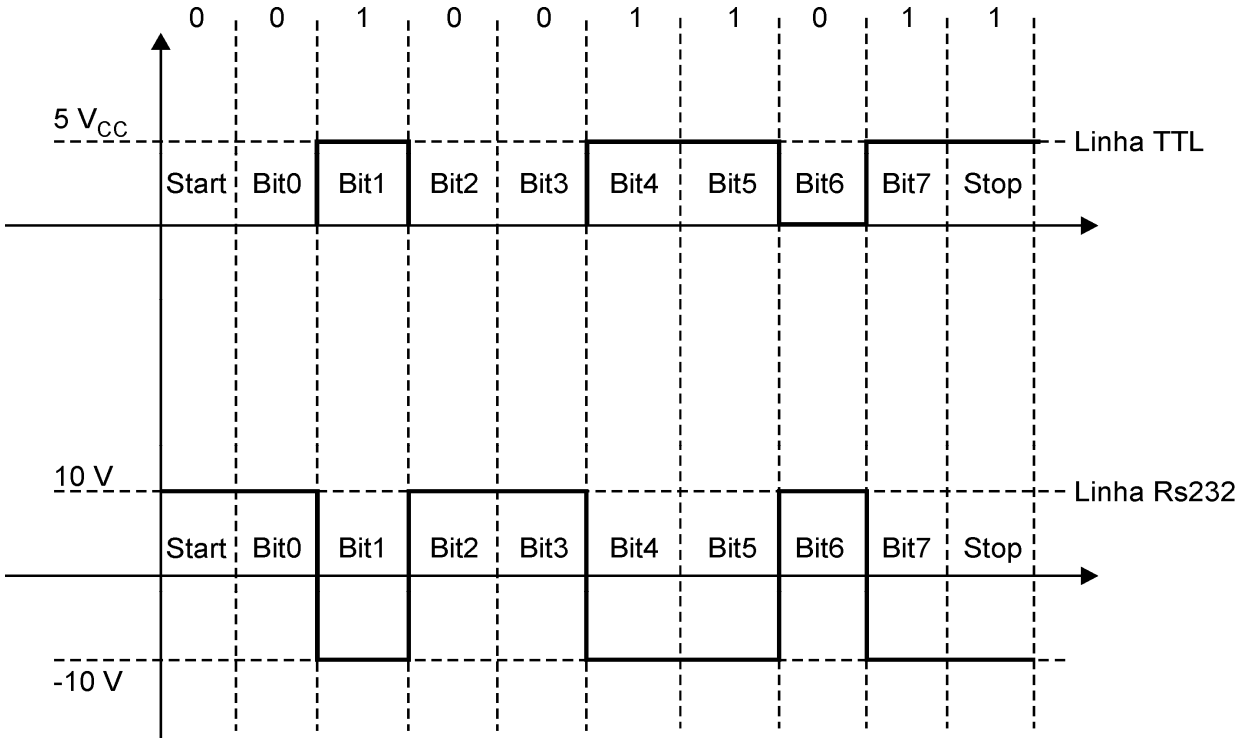

Figura 19: Gráfico para comunicação TTL e RS232

Fonte: Elaborado pelo autor

O bit de start sempre começa em 0 seguido dos bits de dados e finalizado pelo bit de stop que sempre finaliza em 1, e a linha de dados volta a seu estado de repouso. Para visualizar os dados recebidos da RS-232, o programa terminal embutido no próprio Windows, chamado HyperTerminal [32] pode ser utilizado. Para os usuários Linux, o programa Gtkterm[33] pode ser utilizado onde o mesmo está disponível para download e instalação nos repositórios do ubuntu [34].

2.7 Comunicação USB

Um sistema USB (Universal Serial Bus) é descrito em três diferentes áreas: Interconexão USB, USB Device e USB Host [35]. A interconexão USB é a maneira no qual os dispositivos USB estão conectados com o host. Os seguintes aspectos estão incluídos: Topologia do Barramento: Modo de conexão entre o device e o host; Relação entre camadas: Capacidade de cada tarefa ser executada na pilha USB; Modo de fluxo de dados: A maneira no qual os dados se movem no sistema sobre o protocolo USB;

A interconexão física utiliza a topologia estrela. Cada HUB (Concentrador) é o centro de cada estrela. Cada fio é ligado ponto-a-ponto entre o host e o HUB. Na figura abaixo podemos apreciar melhor este conceito.

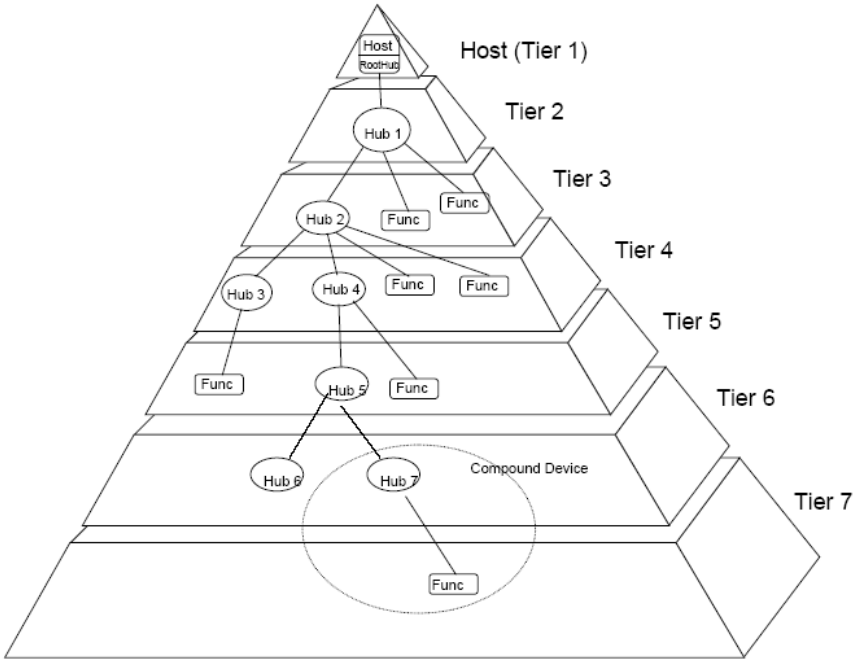

Figura 20: Interconexão tipo estrela [35]

Somente há um USB Host (Hospedeiro) no barramento USB. Esta interface é chamada de Host Controller. O Host Controller pode ser implementado por hardware, firmware ou software.

O USB device são elementos que fornecem recursos a um sistema. Existem apenas dois tipos de devices USB: Hubs, que disponibilizam pontos adicionais de acesso ao USB e

Funções, que disponibilizam capacidades adicionais ao sistema, como joystick digital ou alto-falantes.

Para que um dispositivo seja considerado USB ele deve atendem aos seguintes termos: Compreendem o protocolo USB e respondem as operações standars (padrão) do protocolo, como configuração e reset.

A USB implementa a codificação de dados do tipo NRZI para transmissão de dados. Na codificação do NRZI, o "1" é representado por não haver troca do nível enquanto o "0" representa uma troca. Uma string de zeros causa no NRZI uma troca de bit a cada tempo. Uma string de uns causa um período de inatividade no barramento. A próxima figura exemplifica tal codificação.

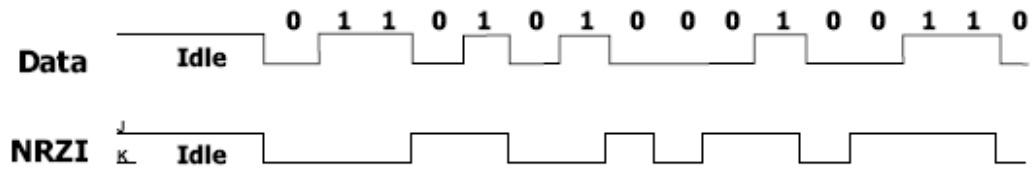

Figura 21: Codificação NRZI [35]

Para garantir sincronismo na rede, após a transmissão de 6 bits em estado "1", o NRZI impõe um "0" para garantir a transição na linha e assegurar o sincronismo na comunicação. Tal bit é chamado de bit de stuffing.

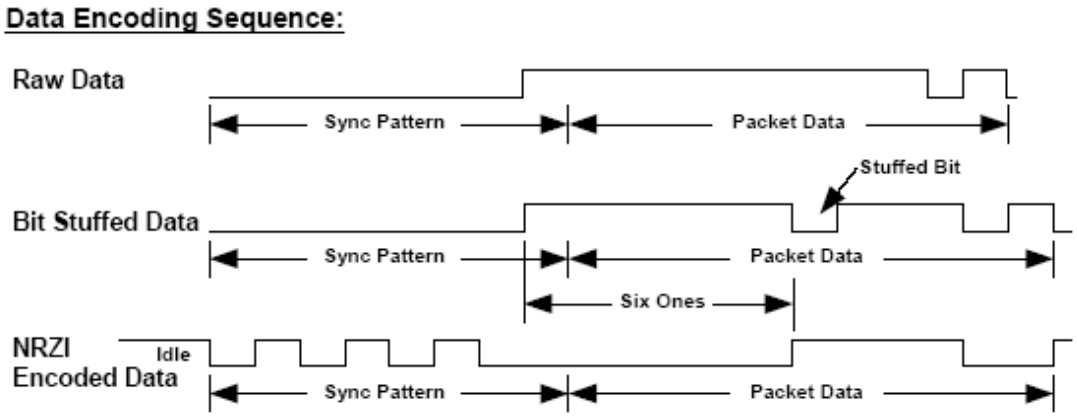

Figura 22: Bit de stuffing [35]

As velocidades de comunicação disponíveis para o USB são as seguintes: High Speed – 480 Mbps podendo variar +-0,05%; Full-Speed – 12 Mbps podendo variar +-0,25% e Low-Speed – 1,5 Mbps podendo variar 1,5 %. No projeto a interface entre o PC e a placa será feita em modo Full-Speed.

Os cabos USB devem atender aos seguintes critérios apresentados na próxima tabela em termos de impedância do cabo para poderem operar normalmente.

Frequency (MHz)	Attenuation (maximum) dB/cable
0.064	0.08
0.256	0.11
0.512	0.13
0.772	0.15
1.000	0.20
4.000	0.39
8.000	0.57
12.000	0.67
24.000	0.95
48.000	1.35
96.000	1.9
200.00	3.2
400.00	5.8

Tabela 5: Atenuação do cabo USB [35]

A potência exigida por cada dispositivo pode ser simplificada com a introdução do conceito de unidade de carga [35]. A unidade de carga é definida por uma corrente de 100 mA. O número de unidade de carga de um dispositivo pode consumir é um valor absoluto independente do tempo. Um dispositivo pode ser low-power consumindo uma unidade de carga e high-power consumindo cinco unidades de carga. Por default, todos os dispositivos são low-power. A transição para high-power é feita através de controle de software. É responsabilidade do software assegurar a potência adequada para o funcionamento do dispositivo. O USB suporta uma faixa de fontes de energia, que pode ser observado na próxima tabela.

Classe	Funcionalidade
Bus-powered hubs	Estas unidades somente podem consumir uma unidade de carga na configuração e cinco após a mesma.
Self-powered hubs	A energia para o funcionamento não são provenientes do Vbus do barramento.
Low-power bus-powered functions	Toda a energia para o funcionamento do dispositivo vem do Vbus. Este dispositivo pode consumir no máximo uma unidade de carga.
High-power bus-powered functions	Toda a energia para o funcionamento do dispositivo vem do Vbus. Este dispositivo pode consumir no máximo uma cinco unidades de carga.
Self-powered functions	Pode consumir uma unidade de carga do V restante é fornecido através de uma fonte ext

Tabela 6: Classes de alimentação USB [35]

A próxima figura apresenta alguns conectores (receptáculos) típicos utilizados no barramento USB.

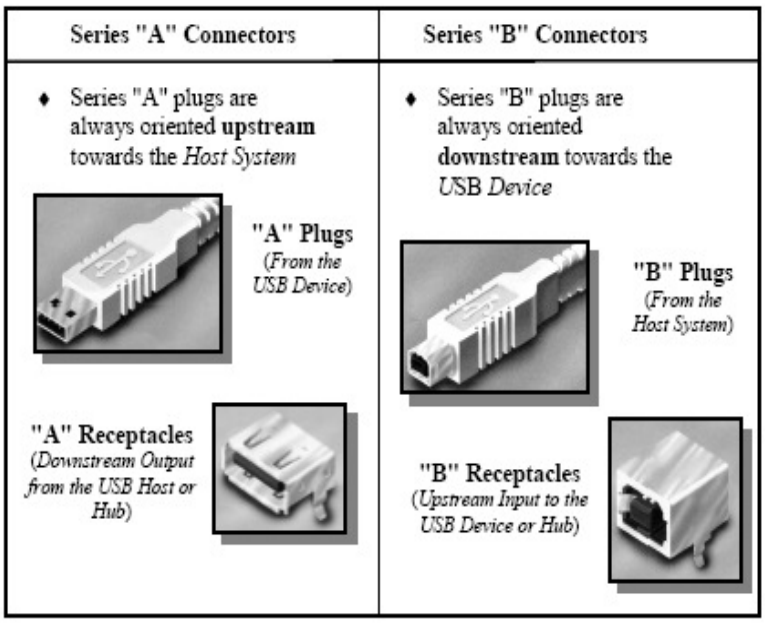

Figura 23: Receptáculos USB [35]

Na comunicação entre host e device todos os bits são transmitidos primeiramente pelo bit LSB (Less Siginificant Bit – Bit Menos Significativo) e todos os pacotes são inicializados pelo campo de SYNC [39]. A função deste campo é que o circuito de entrada do receptor ajuste o seu clock com o do transmissor. No modo full / low-speed são gerados 8 bits enquanto que no high-speed 32. Os pacotes iniciam com o campo de Start e finalizam com o campo de End. O Start-of-Packet (SOP) é uma parte do campo de SYNC e o End-of-Packet (EOP) sinaliza o fim do pacote. O campo de identificação do pacote (PID) segue imediatamente após o campo de SYNC. Um PID consiste de quatro pacotes seguidos de quatro bits de checagem de campo, conforme sugere a próxima figura.

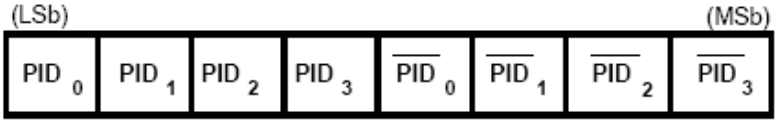

Figura 24: Campo de PID [35]

O campo de endereço possui a função de especificar o dispositivo em uma rede. Assim que o device é energizado, ele recebe o endereço 0 e aguarda o Host Controller endereçar o mesmo. O endereço 0 não pode ser usado em função disto.

O campo de end-point é usado para permitir mais flexibilidade no endereçamento nos momentos em que há as transações de IN, OUT e SETUP. Full-speed e High-Speed suportam até 16 endpoints enquanto o low-speed suportam no máximo 3.

O frame de número de campo é uma seqüência de 11 bits que é incrementado pelo host a cada frame trafegado pela rede. Quando o valor máximo é atingido (0x7FFF) este valor retorna a 0. O campo de dados pode variar na faixa de 0 a 1024 bytes e tem um número integral de bytes. Na figura a seguir está apresentado o formato da transmissão dos bytes:

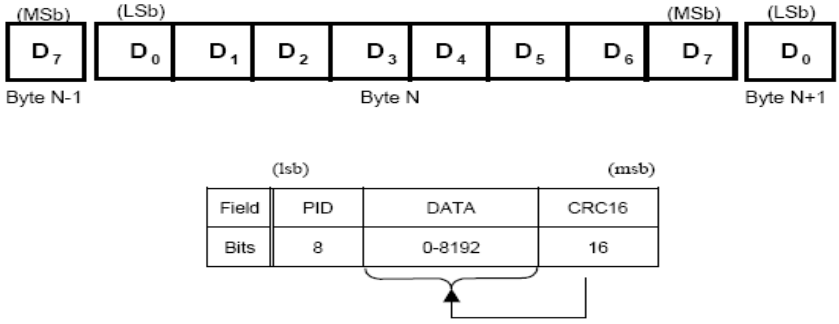

Figura 25: Campo de Dados [35]

O campo de CRC (Cyclic Redundanct Check) é usado para garantir a integridade na comunicação de dados. Desta forma, pode se garantir que o pacote transferido pelo Host será recebido pelo device.

Após o dispositivo ser energizado, o mesmo não pode responder a qualquer transação do barramento até que seja resetado pelo barramento. Após a condição de reset, o dispositivo é endereçado para o seu endereço default (0). Quando o processo de reset está completo, o dispositivo USB opera de acordo com a sua velocidade (low/full/high). A velocidade é selecionada através da terminação de resistores do dispositivo, conforme apresentado na figura abaixo.

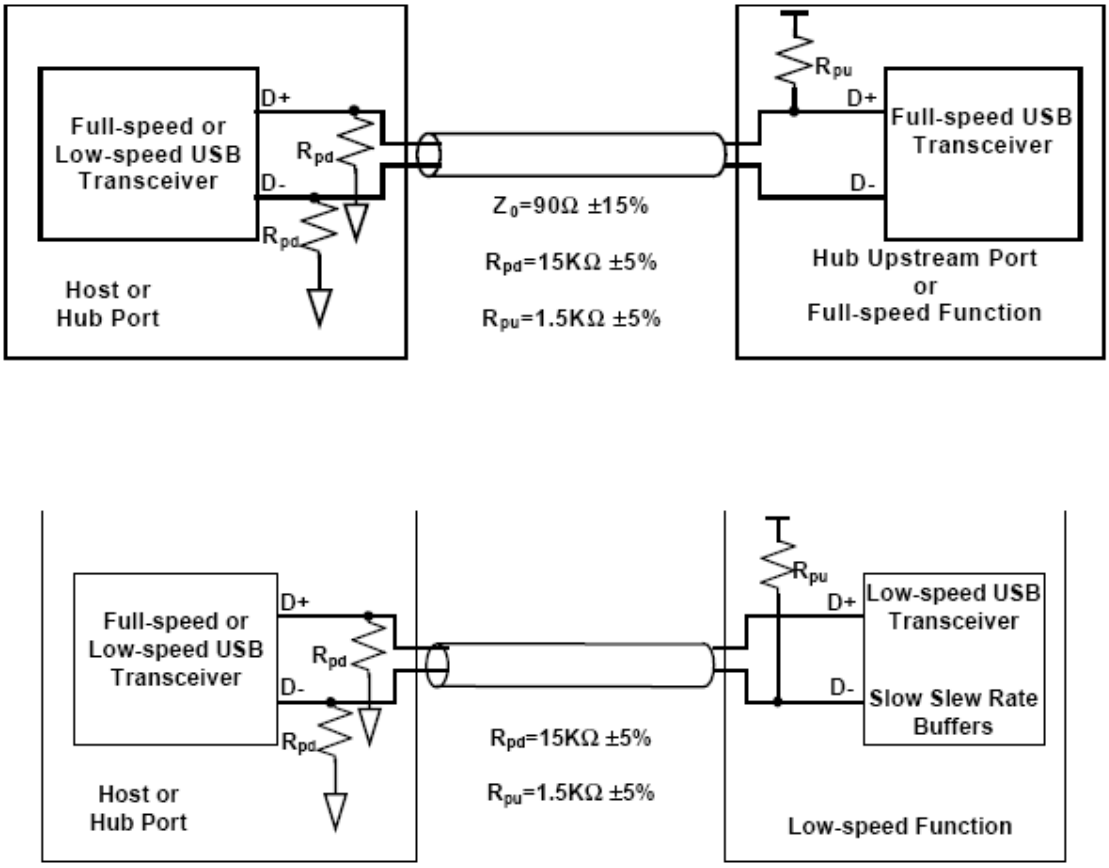

Figura 26: Detecção da velocidade de comunicação [35]

Para o modo High-Speed, o mesmo inicia operando em Full-Speed e através de protocolo de software, altera para o modo High-Speed.

Todos os dispositivos USB têm um endereço único após a condição de reset. Cada device recebe o seu endereço pelo host controller. O dispositivo mantém este endereço até que o mesmo entre em condição de suspensão.

Existem quatro modos de comunicação utilizados na USB, sendo estes o Control Transfer, Bulk Transfer, Interrupt Transfer e Isochronous Transfer. No primeiro caso o controle de dados é usado pelo sistema de software USB para configurar os dispositivos quando os mesmos são conectados ao host. Outros drivers de software podem usar este modo para implementações específicas. O modo bulk consiste no envio de dados em massa, comumente utilizado para impressoras e scanners onde a transferência é sequencial. No modo de interrupção temos uma latência limitada de comunicação entre o dispositivo e o host por um determinado tempo síncrono. Já no modo Isochronous Transfers ***a*** transferência continua em tempo real. É muito usado para aplicações que envolvam voz, por exemplo.

2.8 Controle de corrente por DAC e PWM

Como fonte de corrente um FET como o IRF540N[36] pode ser utilizado. A simbologia deste componente está apresentada na figura a seguir.

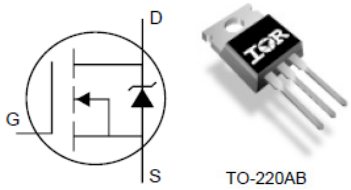

Figura 27: IRF540N [36]

Esse FET apresenta uma resistência DRENO-SOURCE (V_{DS}) de 44 mΩ. O FET irá operar nesta aplicação em sua região linear, em que alterando a tensão do GATE obtém-se assim a corrente que será fornecida a carga. A curva da região linear do FET está apresentada a seguir.

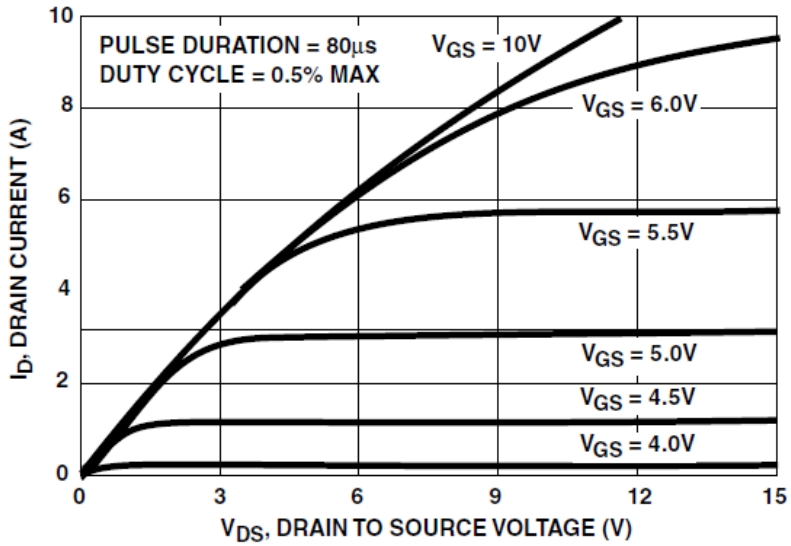

Figura 28: Curva característica do IRF540N [36]

Quanto maior a tensão aplicada ao GATE *(V_{GS})*, maior é a corrente $I_{D,}$ ou seja, a que alimenta a carga na fonte de corrente. Notamos assim que o microcontrolador irá controlar a corrente fornecida a carga ajustando a tensão no GATE do FET (V_{GS}).

Para fazer isso, um PWM com um filtro na saída ou DAC podem ser utilizados, onde assim poderemos escalonar de uma tensão que irá de 0 a 5V, como pode ser visto a seguir.

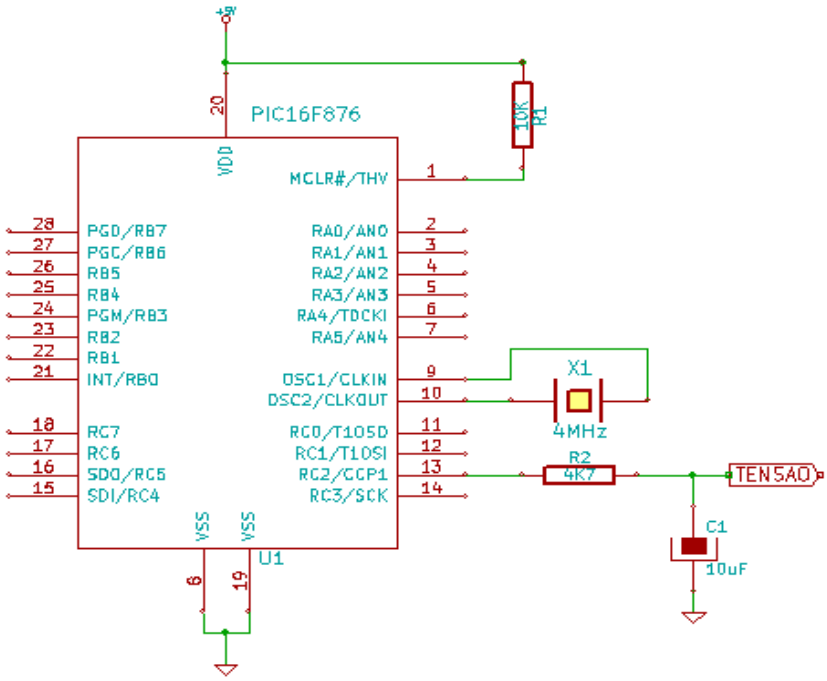

Figura 29: PWM como filtro na saída

Fonte: Elaborado pelo autor

Com o PWM operando em modo de 8 bits podemos variar em uma faixa de 0 a 255 a saída do mesmo. Sendo assim, quando o duty cycle for ajustado para 255, teremos 5V e quando for ajustada, por exemplo, para 127 o valor de 2,5 V. Esta técnica pode ser representada em um gráfico linear como o apresentado a seguir em que apresenta o valor ajustado no duty cycle e a tensão obtida na saída do filtro.

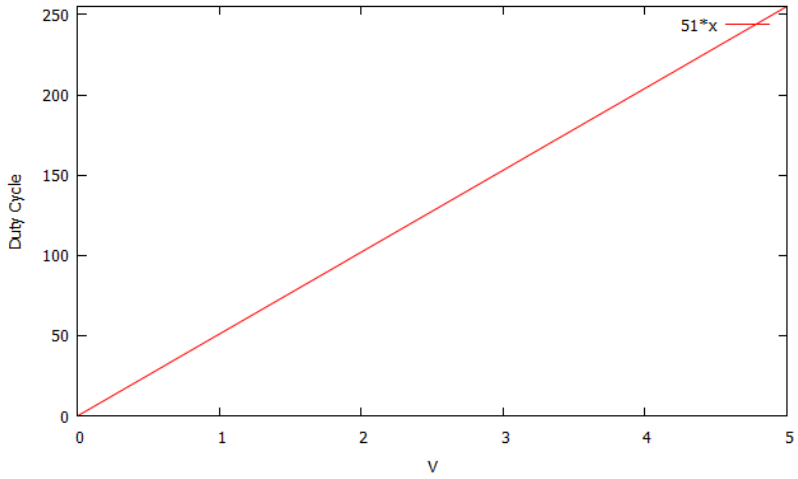

Figura 30: Saída do AOP de acordo com o duty cycle

Fonte: Elaborado pelo autor

Porém o FET é alimentado com uma tensão DC de 12V mas a saída do microcontrolador é de 0 a 5V, tensão insuficiente para controlar o V_{GS} do FET. Sendo assim, há a necessidade de um amplificador não inversor [37] baseado em um AOP LM358[38] para termos na entrada do FET a faixa de 0 a 12V. Isso será conseguido com o circuito apresentado a seguir.

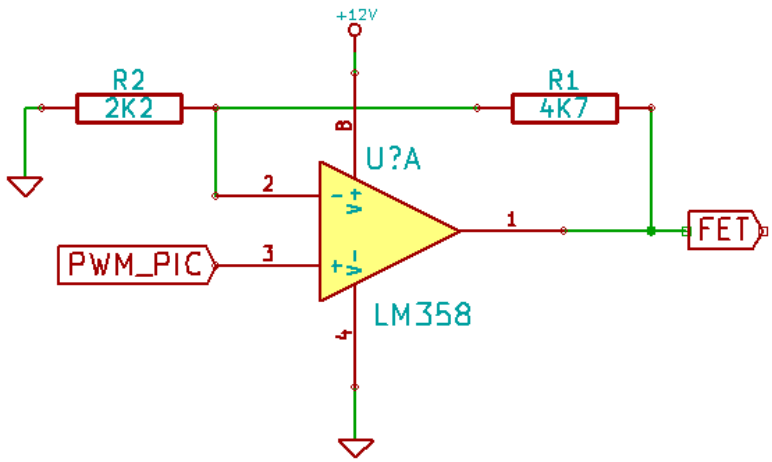

Figura 31: Circuito amplificador para o FET

Fonte: Elaborado pelo autor

O ganho será determinado pela relação entre R1 e R2. O próximo gráfico mostra a relação entre entrada e saída do AOP.

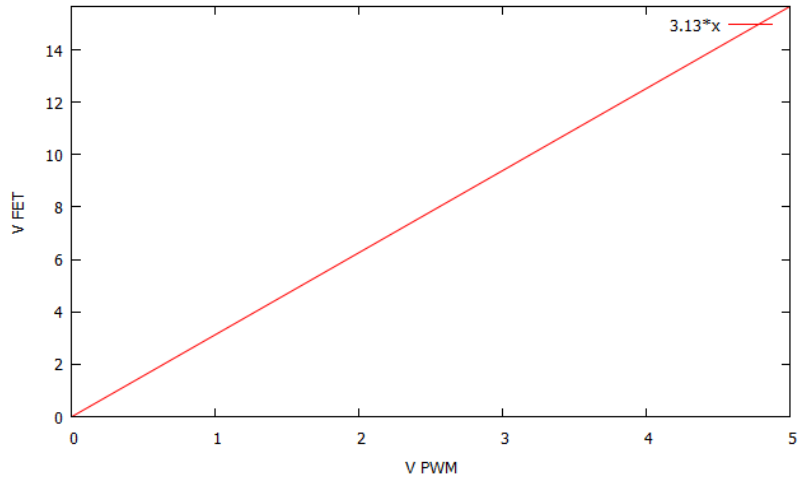

Figura 32: Relação entre V_{in} e V_{out}

Fonte: Elaborado pelo autor

Na placa driver proposta haverá duas formas de realizar o controle de corrente do laser, sendo uma através de um DAC e outra através de PWM. Ambas permitem operar na região linear do FET IRF730[39] da placa de potência, mudando assim a corrente fornecida a carga, neste caso o laser já que varia a tensão V_{GS} do máximo até o mínimo, alterando assim a corrente fornecida a carga. A curva característica do IRF730 que mostra a relação da corrente I_d em relação a tensão V_{GS} está apresentada a seguir.

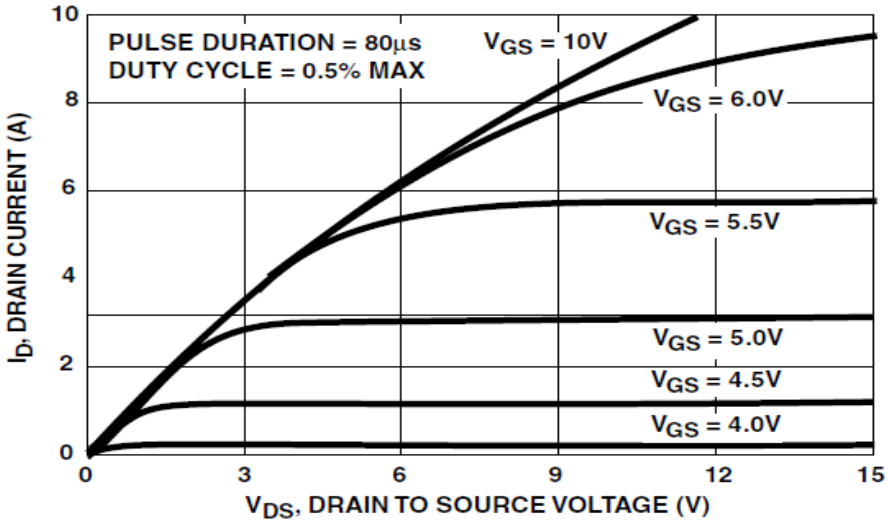

Figura 33: Curva I_D x $_{VDS}$

Fonte: IRF730 Datasheet

O DAC utilizado foi o MCP4822[40], cuja pinagem está apresentada na próxima figura.

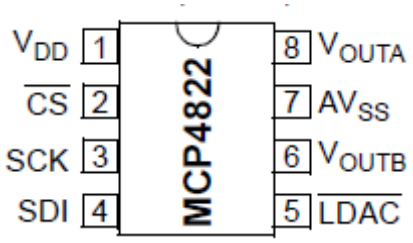

Figura 34: Pinagem do MCP4822

Fonte: http://ww1.microchip.com/downloads/en/DeviceDoc/41350E.pdf

Este chip apresenta conexão SPI com o mestre, neste caso o dsPIC30F2020 e duas saídas de tensão com resolução de 12 bits disponível através dos pinos V_{OUTA} e V_{OUTB}, apesar de apenas a primeira saída ser utilizada. A referência interna de 4,096 V garante uma saída linear em toda a faixa de tensão, garantindo assim uma boa precisão no passo de corrente fornecido ao laser. A seguir está apresentado o esquema elétrico referente a conversão DAC, onde observa-se que apenas uma das saídas de tensão está sendo utilizada.

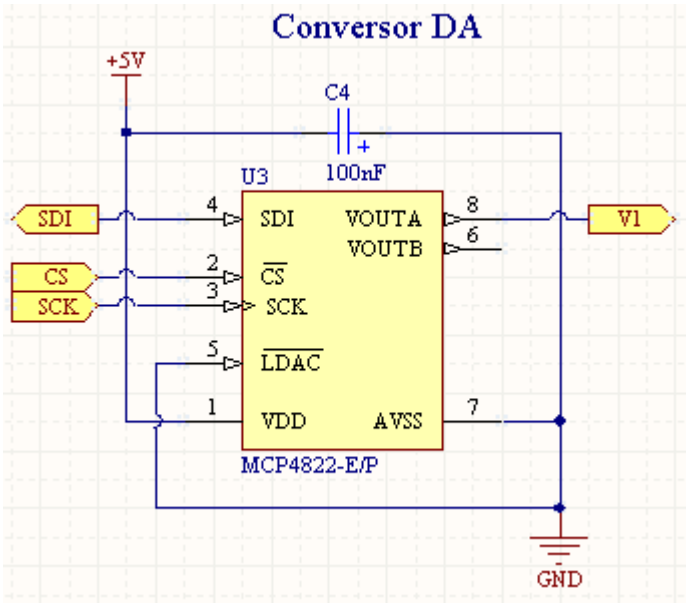

Figura 35: Esquema do conversor DAC

Fonte: Elaborado pelo autor

A outra saída de tensão linear é obtida através do PWM do microcontrolador, que também possui resolução de 12 bits. Neste caso, na saída encontramos um circuito do tipo passa baixa como apresentado a seguir que tem a função de obter a tensão média através

do sinal digital obtido da saída de PWM. A frequência utilizada está centrada em 5 kHz, onde alterando o ciclo ativo (duty cycle) será possível assim alterar a tensão média obtida. O circuito passa baixa está apresentado a seguir.

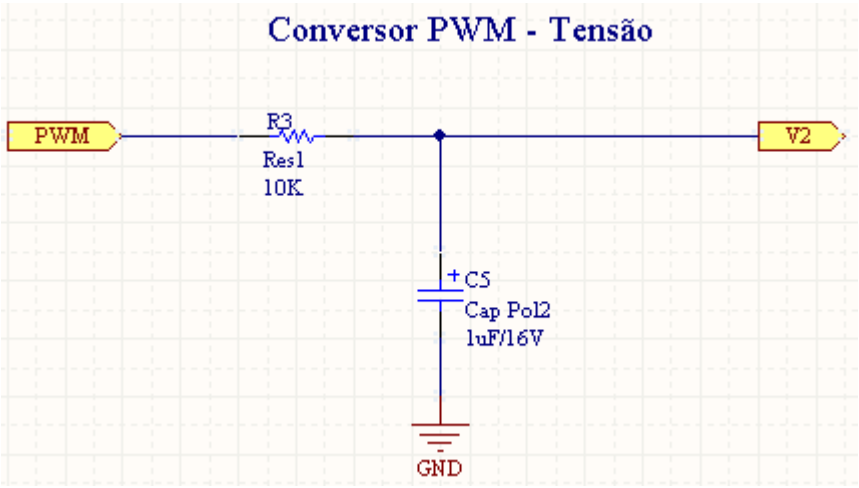

Figura 36: Esquema da saída de PWM

Fonte: Elaborado pelo autor

Em ambos os casos, a saída de tensão está abaixo da tensão máxima no qual o FET está alimentado, neste caso 12 V. Para isso, adicionalmente há um somador de tensão para unir ambos os sinais e em seguida um amplificado com ganho próximo a 3 que faz que desta forma o controlador possa variar em toda a faixa de tensão de alimentação do FET a corrente fornecida ao laser. O circuito somador e amplificador podem ser observados no esquema a seguir.

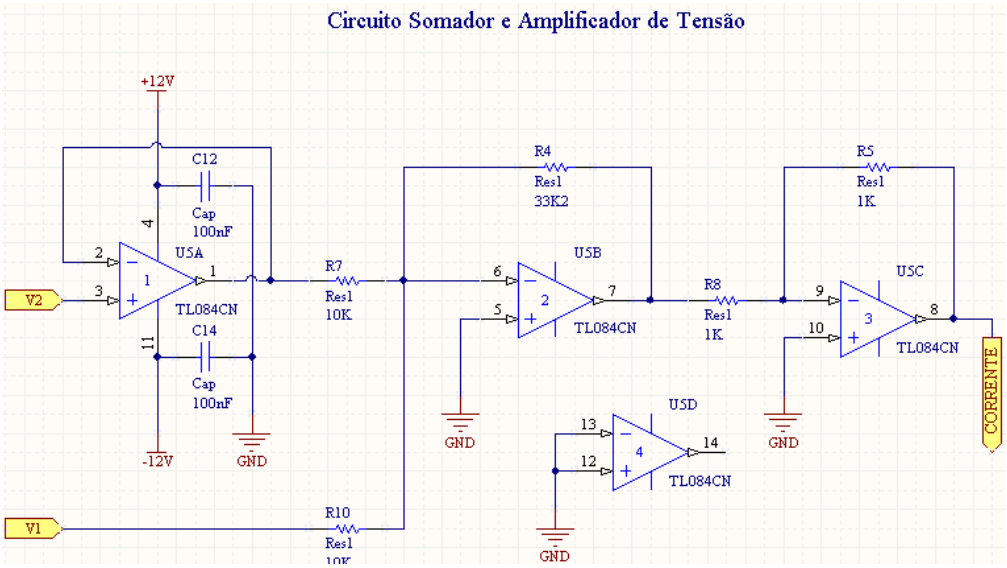

Figura 37: Esquema do circuito somador e amplificador

Fonte: Elaborado pelo autor

A saída denominada CORRENTE é aplicada a entrada do FET IRF730 onde desta forma o microcontrolador poderá fazer o ajuste da corrente fornecida a carga de 0 a 200 mA. Este circuito foi construído baseado no amplificador operacional TL084[41], que é um AOP de baixo consumo e alta impedância de entrada, já que sua construção é feita baseada na tecnologia JFET. A pinagem deste CI (Circuito Integrado) está apresentada a seguir.

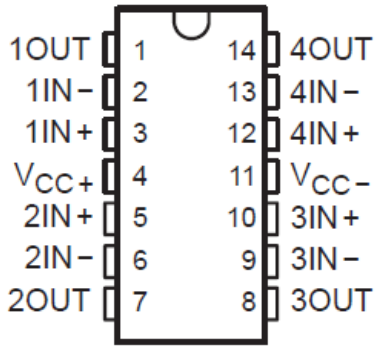

Figura 38: Pinagem do TL084
Fonte: http://www.ti.com/lit/ds/symlink/tl084.pdf

2.9 Feedback de Corrente

Uma fonte de corrente é representada pelo símbolo apresentado a seguir.

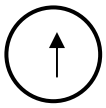

Figura 39: Representação de uma fonte de corrente
Fonte: Elaborado pelo autor

A principal característica da fonte de corrente é manter o fornecimento de corrente constante independente da carga que é colocada em série com a mesma. Na figura abaixo, podemos visualizar um resistor (carga) ligado em série com a fonte de corrente ajustada para 50 mA.

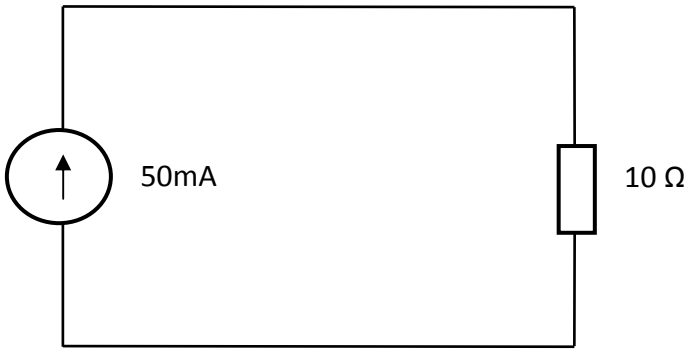

Figura 40: Representação de uma fonte de corrente em série com carga de 10 Ω

Fonte: Elaborado pelo autor

O exemplo acima mostra uma fonte de corrente que fornece 50 mA a uma resistência de 10 Ω. Já o próximo exemplo mostra a mesma fonte de corrente, porém alimentando outra carga.

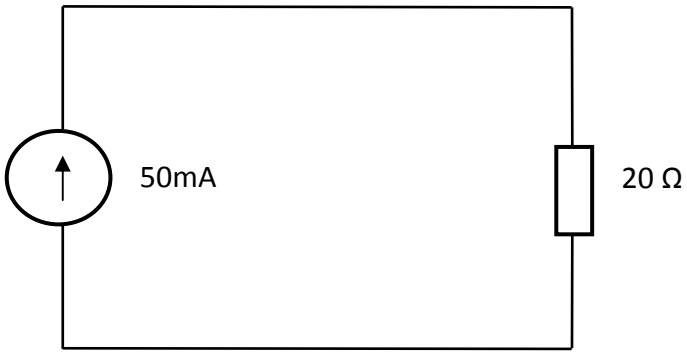

Figura 41: Representação de uma fonte de corrente em série com carga de 20 Ω

Fonte: Elaborado pelo autor

Essa é a principal característica da fonte de corrente, ser capaz de manter uma corrente constante independente do valor da carga a ela conectada.

O resistor shunt nada mais é que um resistor de baixíssimo valor ligado em série com a carga que é usado para fornecer um feedback (retorno) ao microcontrolador de modo a este saber se a corrente que está sendo fornecida a carga está dentro do esperado ou não. Por exemplo, a figura a seguir ilustra como é conectado este resistor no circuito.

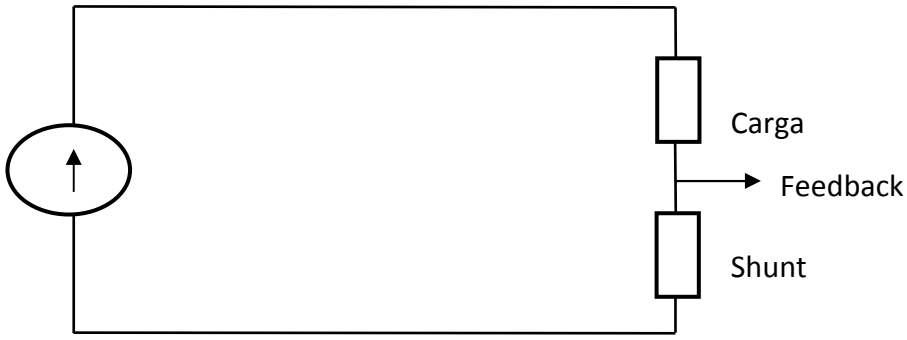

Figura 42: Circuito com resistor shunt

Fonte: Elaborado pelo autor

Toda a corrente que passa pela carga passa também pelo resistor de shunt, gerando assim uma tensão de feedback proporcional a corrente do circuito. O resistor shunt usado na placa Cerne Fonte de Corrente é de 0,33 Ω. Logo, a tensão de feedback proporcional a corrente será dada pela Lei de Ohm como expresso na próxima tabela.

$$V = R.I$$

Tabela 7: Lei de Ohm

Como neste caso o valor de R vale 0,33 Ω, teremos o resultado expresso a seguir.

$$V = 0{,}33.I$$

Tabela 8: Lei de Ohm aplicado ao circuito

Onde I dependerá da corrente que estiver passando pelo circuito. Por exemplo, digamos que tenhamos o circuito com a seguinte carga conectada.

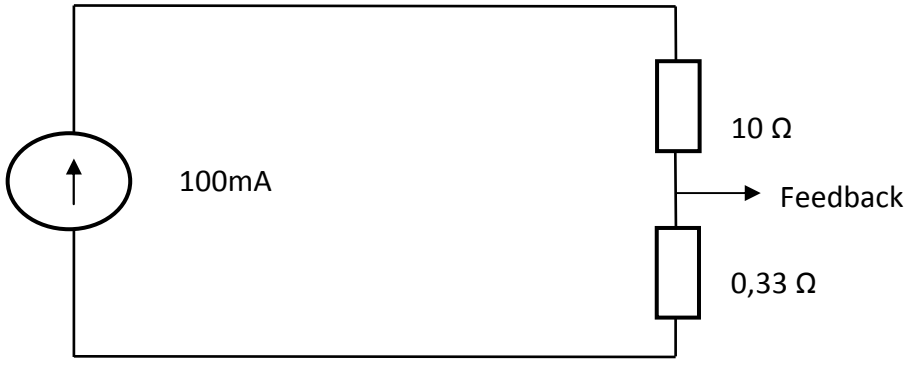

Figura 43: Simulação de circuito para uma carga de 10 Ω

Fonte: Elaborado pelo autor

A tensão no ponto feedback será dada de acordo com a tabela a seguir.

$$V = 0{,}33.I$$
$$V = 0{,}33.100x10^{-3}$$
$$V = 0{,}33\,V$$

Tabela 9: Obtendo a tensão de feedback

Este valor será alcançado independente da carga ôhmica que esteja conectada ao circuito. Observe que no caso abaixo, o valor da tensão de feedback seria o mesmo.

Figura 44: Simulação de circuito para uma carga de 30 Ω
Fonte: Elaborado pelo autor

O circuito de driver de laser circuito está preparado para suportar uma corrente máxima de até 200 mA, logo a tensão máxima de feedback será a apresetada na tabela a seguir.

$$V = 0{,}33.I$$
$$V = 0{,}33.200x10^{-3}$$
$$V = 0{,}066\,V$$

Tabela 10: Tensão de feedback do circuito driver de laser

Esta tensão é relativamente pequena para a entrada analógica do microcontrolador que trabalha na faixa de 0 a 5 V. Sendo assim, será utilizado um circuito amplificador de tensão

usando um amplificador de instrumentação conectado ao ponto de feedback que irá amplificar este sinal. Um circuito que faz uso de amplificador operacional em modo não inversor é obtido através da figura abaixo.

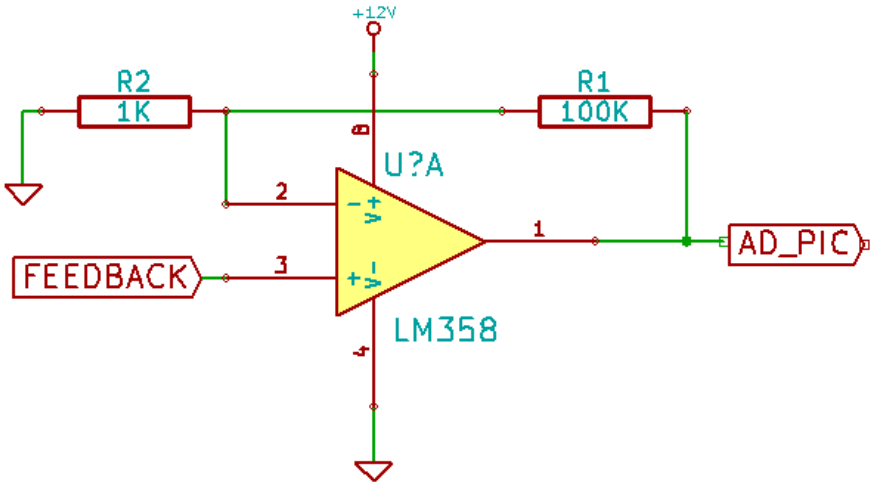

Figura 45: Exemplo de circuito amplificador não inversor
Fonte: Elaborado pelo autor

O ganho deste circuito é dado pela relação entre R1 e R2 conforme apresentado na tabela a seguir.

$$G = 1 + \frac{R1}{R2}$$

Tabela 11: Cálculo do ganho do AOP em modo não inversor

Considerando por exemplo R1=100kΩ e R2=1,5 kΩ temos o ganho apresentado a seguir.

$$G = 1 + \frac{100}{1,5} = 67,66$$

Tabela 12: Cálculo do ganho do AOP de acordo com R1 e R2

Podemos neste caso considerar o ganho igual a 68. Desta forma, quando tivermos a tensão de feedback igual a 0,066V, teremos na entrada do AD do dsPIC o valor de 4,48 V. De modo linear, teremos toda a leitura na faixa de 0 a 200 mA, o que dará uma tensão de 0 a aproximadamente 5 V como expresso no gráfico a seguir.

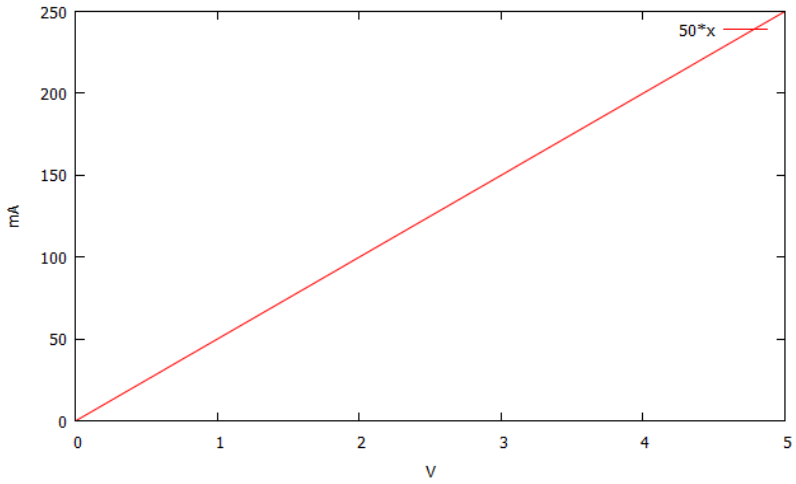

Figura 46: Relação linear entre corrente e tensão
Fonte: Elaborado pelo autor

Desta forma, o microcontrolador será capaz de controlar a corrente fornecida a carga de modo a deixá-la no valor ajustado na fonte de corrente, daí a importância do retorno de feedback usando o resistor shunt. Um exemplo de circuito completo da parte de feedback pode ser visualizado a seguir.

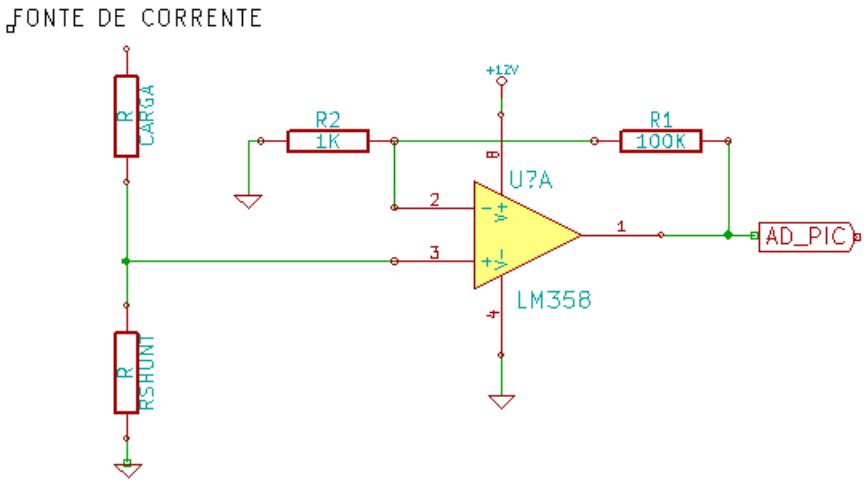

Figura 47: Exemplo de circuito de feedback
Fonte: Elaborado pelo autor

O feedback de corrente é o circuito que permitirá ao microcontrolador mensurar se a corrente que foi programada para ser fornecida está de acordo com o que está sendo consumido de maneira real, naquele instante. Para isso, foi utilizado o amplificador operacional de instrumentação INA121[42], cuja figura abaixo apresenta a sua pinagem.

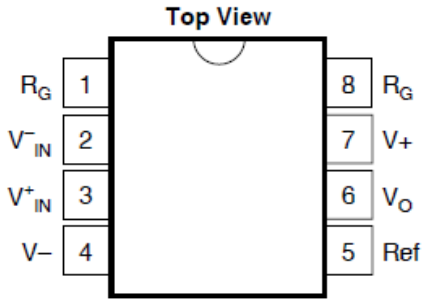

Figura 48: Pinagem do INA121

Fonte: http://www.ti.com/lit/ds/symlink/ina121.pdf

Um circuito do tipo shunt proverá a corrente que estará sendo consumida pela carga (laser) no qual este retorno estará conectado as entradas V^-_{IN} e V^+_{IN}. De acordo com a diferença desta entrada diferencial, o INA121 fornecerá na saída V_o uma tensão proporcional de acordo com o ganho ajustado em R_G, que segue a fórmula da tabela abaixo.

$$G = 1 + \frac{50k\Omega}{R_G}$$

Tabela 13: Configurando o ganho do INA121

Fonte: http://www.ti.com/lit/ds/symlink/ina121.pdf

Como o resistor shunt utilizado foi de 0,33Ω e a corrente máxima projetada é de 200mA, sabemos, segundo a Lei de Ohm que a tensão diferencial será dado de acordo com o apresentado na tabela a seguir.

$$V = RI$$
$$V = 0{,}33 * 200x10^{-3}$$
$$V = 0{,}066V$$

Tabela 14: Tensão de saída do shunt

Fonte: Elaborado pelo autor

O resistor R_G utilizado no projeto foi de 1 kΩ, onde desta forma observa-se que o ganho será de 51, segundo a tabela 3. Desta forma, a saída de tensão do INA121 será o valor apresentado na tabela 5.

$$V_{OUT} = G.V_{IN}$$
$$V_{OUT} = 51.0{,}066$$
$$V_{OUT} = 3{,}366V$$

Tabela 15: Saída de tensão amplificada

Fonte: Elaborado pelo autor

Este valor será fornecido ao dsPIC30F2020, que terá assim o feedback de corrente de forma a manter a mesma dentro da faixa parametrizada no programa. O circuito apresentado na próxima figura é o que está implementado no driver para fazer o feedback de corrente.

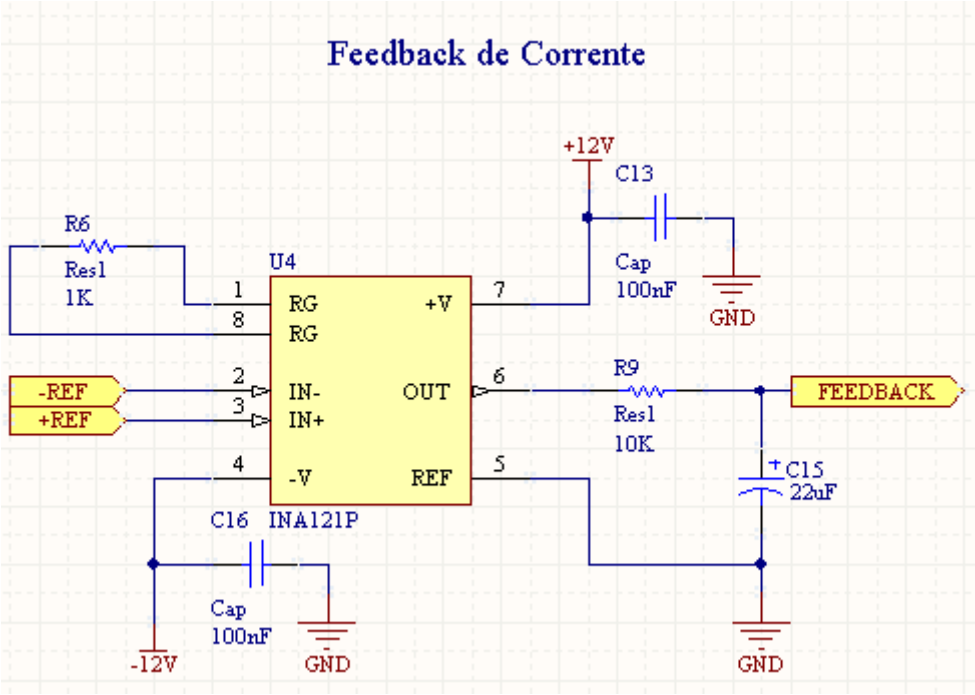

Figura 49: Esquema do circuito feedback de corrente

Fonte: Elaborado pelo autor

Na saída de tensão encontramos um filtro do tipo passa baixa que tem a função de filtrar esta saída para deixá-la o mais estável o possível. A saída FEEDBACK ficará conectada a entrada AD do microcontrolador de forma que este tenha a medição atual da corrente fornecida a carga.

O controle da potência do laser pode ser feita também através de um feedback óptico ao invés de um feedback por corrente [09]. Na próxima figura está apresentado um circuito que proporciona tal funcionamento.

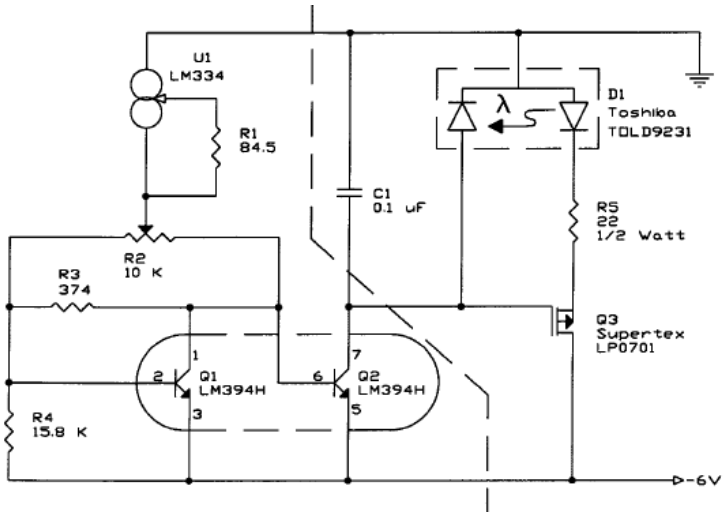

Figura 50: Esquema do circuito feedback óptico [09]

2.10 Comunicação I2C

O protocolo I^2C[43] foi desenvolvido pela Philips e utiliza dois fios para fazer a comunicação com outros periféricos. É um protocolo síncrono composto de uma linha de clock (SCL) e outra de dados (SDA). Através destas duas linhas é possível escrever e ler os dados com outros dispositivos como memória, conversor AD e DA, shift-registers e etc. As linhas de clock (SCL) e dados (SDA) não conseguem impor o nível lógico "1" para comunicação impondo somente o nível lógico "0". Para garantir este nível, é necessário a utilização de dois resistores de pull-up conectados nestas linhas afim de garantir o nível lógico alto. Estes resistores variam de 1K até 10K e quanto o menor o valor do resistor, maior pode ser a taxa de comunicação.

No protocolo I2C, enquanto não há comunicação de dados, as linhas ficam em seu estado de repouso. No repouso, as duas linhas, tanto de SDA quanto a de SCL ficam em nível alto como apresentado na figura a seguir.

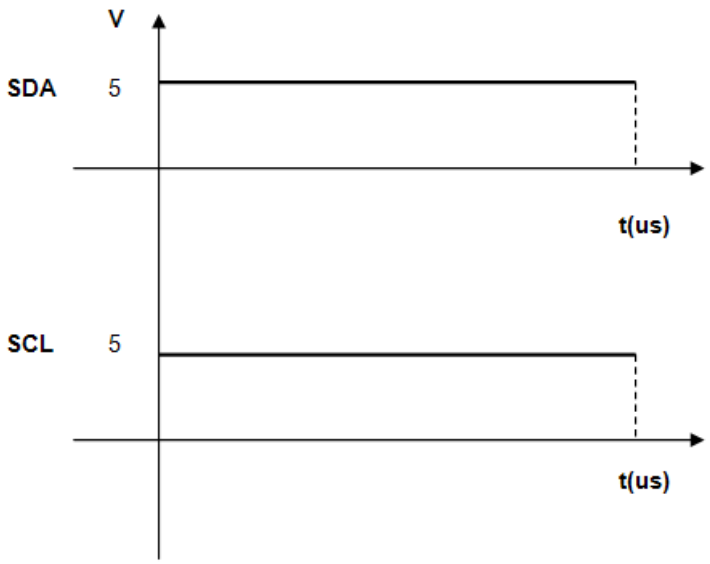

Figura 51: Estado de repouso no I2C

Fonte: Elaborado pelo autor

Toda vez que inicia-se uma comunicação no barramento I2C, existe uma condição que é chamada de start (início). Desta forma, o escravo sabe que o mestre (microcontrolador) iniciará uma nova comunicação na linha, pois a mesma fica da seguinte forma.

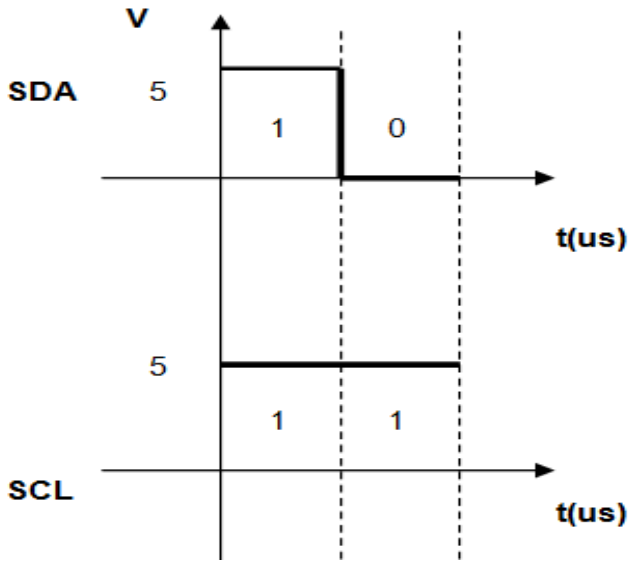

Figura 52: Estado de start no I2C

Fonte: Elaborado pelo autor

O protocolo I2C é chamado do tipo mestre-escravo. Este tipo de protocolo é chamado desta forma pois quem sempre inicia uma comunicação é o mestre restando ao escravo responder as solicitações do mestre. A linha de dados SDA somente pode alterar de estado

de baixo para alto ou de alto para baixo enquanto a linha SCL estiver em nível baixo. Caso esta linha saia do nível alto para o baixo enquanto a linha SCL estiver em nível alto, teremos a condição de start no barramento I2C e os dispositivos ligados na rede passam a receber os dados provenientes deste protocolo. A condição de STOP é gerada toda vez que a linha de dados sai do nível baixo para o alto enquanto a linha SCL estiver em nível lógico alto. Neste momento, todos os escravos ligados na rede deixam de “ouvir” a mesma. O gráfico a seguir apresenta a condição de STOP.

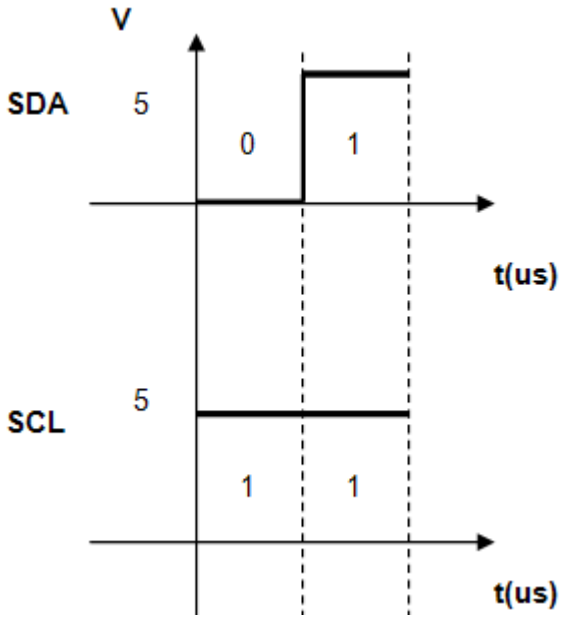

Figura 53: Estado de stop no I2C

Fonte: Elaborado pelo autor

Cada bit de dados é transmistido na borda de descida da linha SCL. Além disso, o mestre gera 9 pulsos de clock para transmissão de cada byte, sendo os oito primeiros para transmissão do byte e o último, neste caso o 9º bit para verificação do ACK do escravo. Quando o mestre gerar o nono pulso de clock, ele verificará o estado da linha de dados. Caso esteja em nível alto, será uma indicação que o dispositivo escravo gerou um NACK, ou seja, os dados não foram corretamente recebidos. Já se a linha de dados estiver em nível lógico baixo, é uma situação de ACK, ou seja o escravo recebeu perfeitamente os dados. Caso ocorra um NACK, o mestre encerra a comunicação com uma condição de STOP.

O bit de dado na linha SDA é transferido na borda de descida da linha SCL. Os dados presentes da linha SDA somente podem variar enquanto a linha SCL estiver em nível baixo.

Em nível alto, a linha SDA não pode ser alterada pois na borda de descida de SCL, o bit será transferido a não ser que seja uma condição de start ou stop.

Está previsto um conector de comunicação I2C na placa de controle, apesar de no momento a mesma não estar sendo utilizada. Isso foi feito para futuras comunicações com outros dispositivos, o que poderá ser feito usando-se este barramento. A figura apresentada a seguir mostra o conector que implementa tal comunicação, onde observa-se a presença dos pinos de comunicação SDA e SCL para comunicação e os pinos de alimentação.

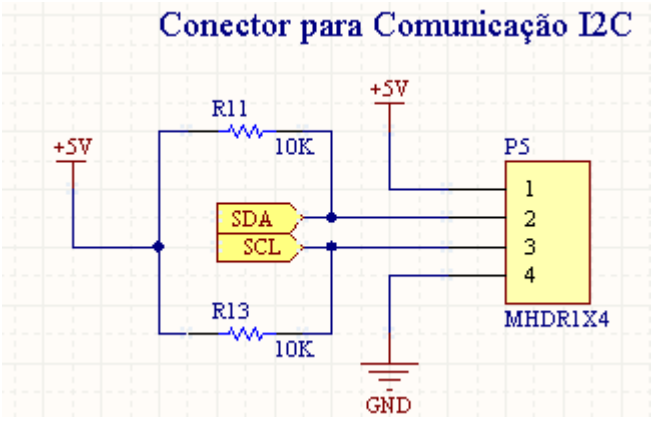

Figura 54: Esquema da conexão I2C

Fonte: Elaborado pelo autor

Os resistores de pull-up nas linhas SDA e SCL são necessários para garantir o estado lógico “1” na linha no momento em que não há comunicação, já que neste barramento apenas os níveis lógicos “0” são transmitidos.

2.11 Conexão com a placa de potência

A conexão com a placa de potência é disponibilizada através de um conector de 5 vias em que neste encontramos o feedback de corrente (+REF e –REF), a saída de tensão linear para controle de corrente (CORRENTE) e o pino que poderá realizar a habilitação ou não do circuito de potência (CORTE). Na próxima figura, tal conexão está apresentada no esquema elétrico da placa.

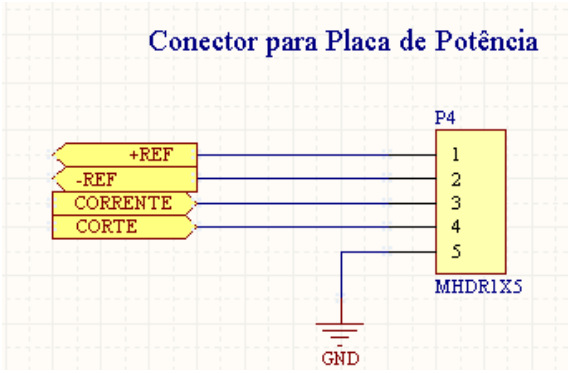

Figura 55: Esquema da conexão de potência

Fonte: Elaborado pelo autor

2.12 Conexão para gravação ICSP

A gravação in-circuit (ICSP) possibilitará que o firmware a ser gravado no microcontrolador seja feito sem ser necessário retirá-lo da placa para atualização. Há um conector deste tipo presente no esquema da placa de controle, como pode ser observado na figura a seguir.

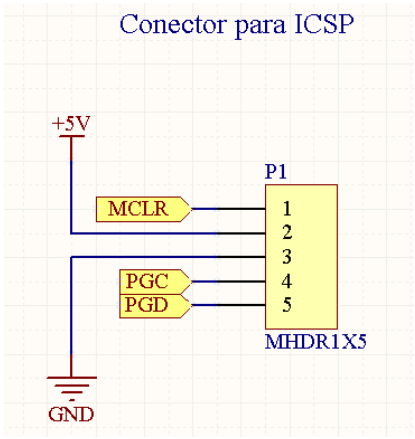

Figura 56: Esquema da conexão ICSP

Fonte: Elaborado pelo autor

Além dos pinos de alimentação, encontramos os pinos utilizados para gravação, a saber MCLR, PGD e PGC.

2.13 Esquema elétrico completo

Após a apresentação em partes do esquema, nas próximas páginas estão apresentados os esquemas da placa driver de controle de forma integrada, permitindo assim observar de forma generalizada o funcionamento desta etapa do projeto.

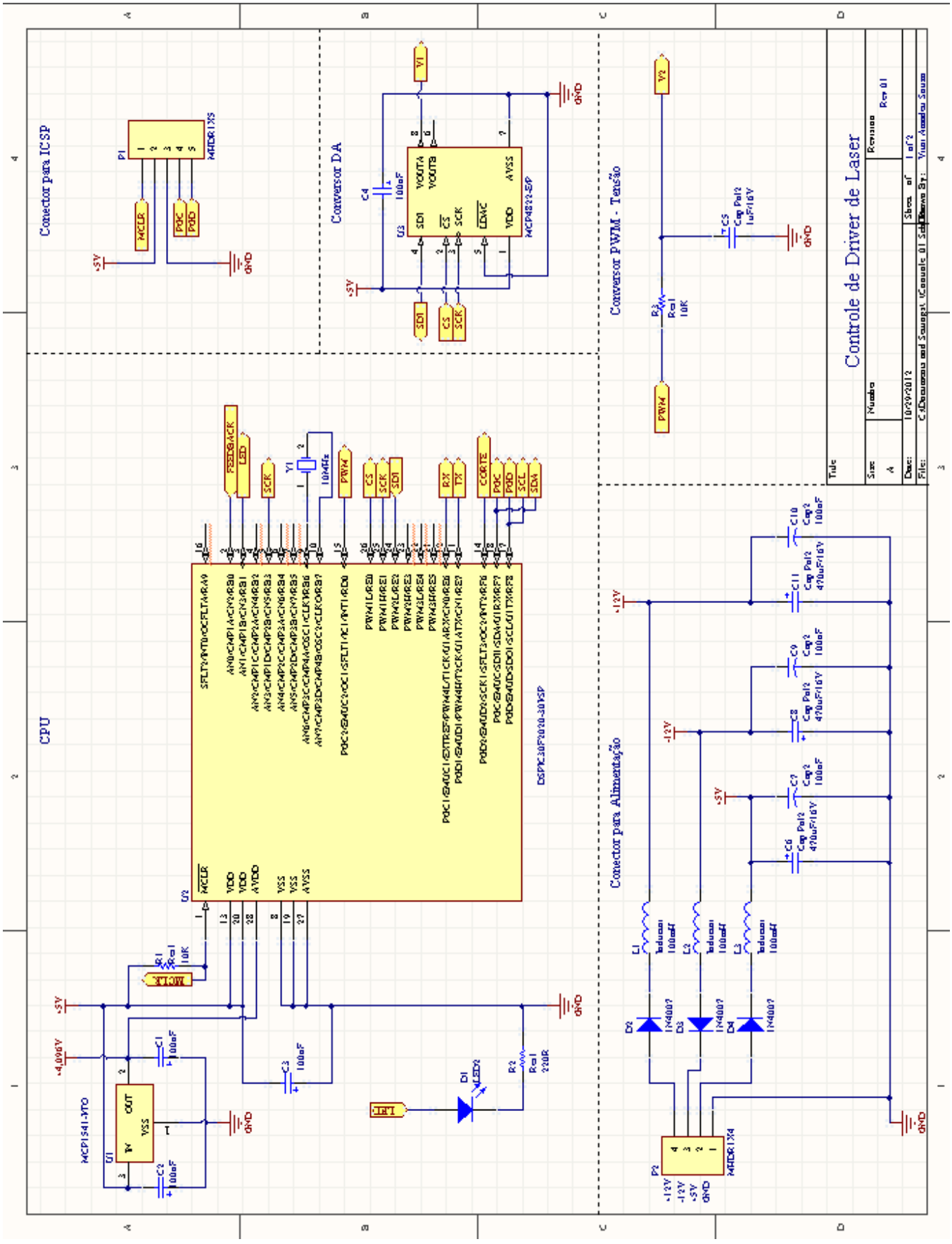

Figura 57: Esquema completo de controle – Parte 1

Fonte: Elaborado pelo autor

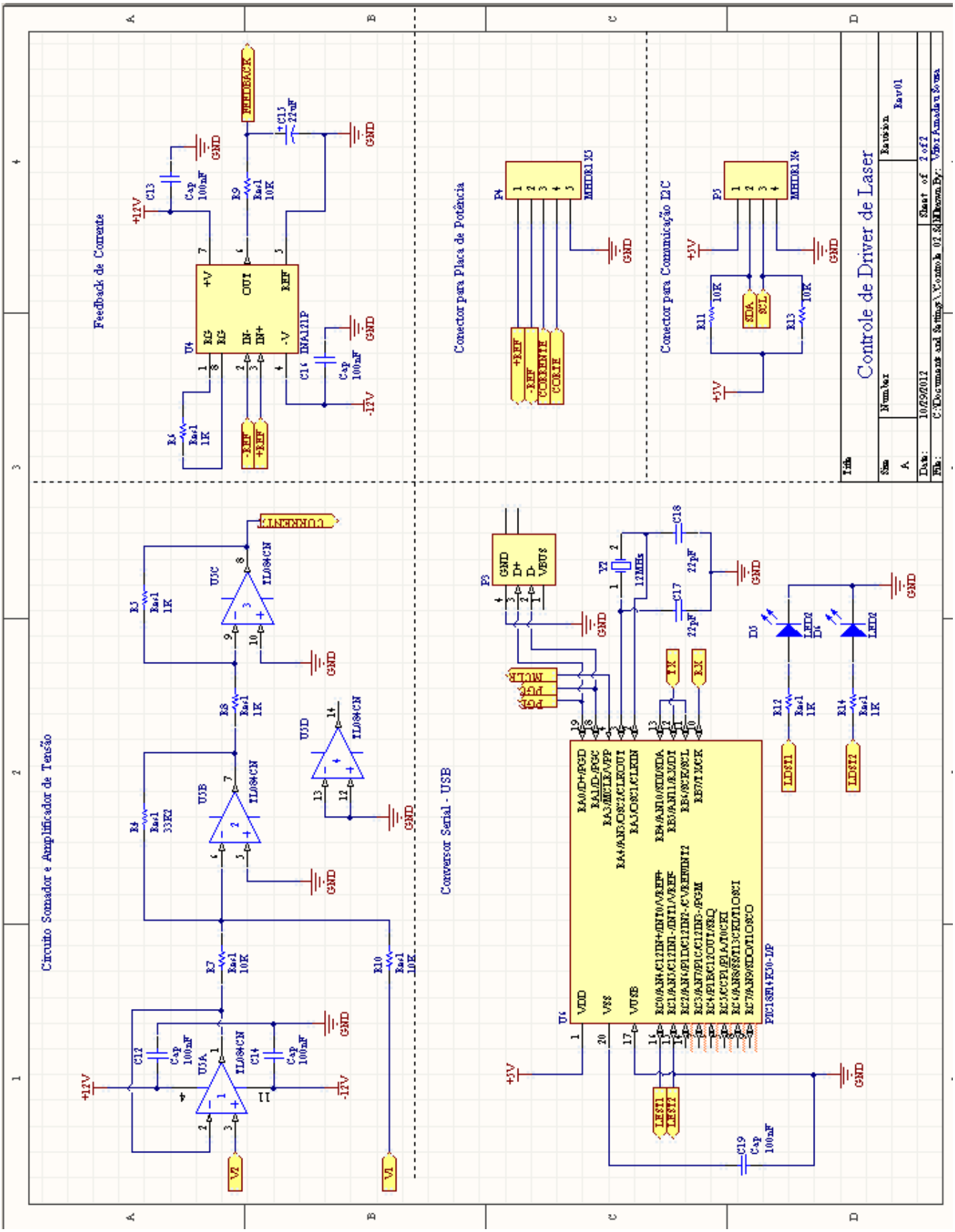

Figura 58: Esquema completo de controle – Parte 2

Fonte: Elaborado pelo autor

2.14 Lista de material

A lista de material para o desenvolvimento da placa de controle está apresentado na tabela abaixo.

Quantidade	Descrição	Referência
14	Capacitor 100 nF cerâmico	C1, C2, C3, C4, C7, C9C 10C12, C13, C14, C16, C17, C18, C19
1	Capacitor 22 uF/16 V eletrolítico	C15
3	Led 3 mm vermelho	D1, D5, D6
1	Receptáculo USB tipo B	P3
2	Conector MOLEX 5 vias 5045	P1, P4
2	Conector MOLEX 4 vias 5045	P2, P5
5	Resistor 1 kΩ ¼ W	R8,R5,R6,R12,R14
5	Resistor 10 kΩ ¼ W	R7,R9,R10,R11,R13
1	Resistor 33 kΩ ¼ W	R4
1	INA121	U4
1	TL084	U5
1	PIC18F14K50 I/P	U6
1	Cristal 12 MHz	Y2
1	Capacitor eletrolítico 1uF/16V	C5
3	Capacitor eletrolítico 470uF/16V	C6, C8, C11
3	Diodo 1N4007	D2, D3, D4
3	Indutor 100 mH axial	L1, L2, L3
1	MCP1541 I/TO	U1
1	dsPIC30F2020	U2
1	MCP4822	U3
1	Cristal 10 MHz	Y1

Tabela 16: Lista de material da placa de controle

Fonte: Elaborado pelo autor

Segundo cotações de preços obtidas em Outubro de 2012, o valor a nível de componentes para construção da placa de controle é o apresentado a seguir.

Quantidade	Descrição	Valor unitário (R$)	Valor total (R$)
14	Capacitor 100 nF cerâmico	0,20	2,80
1	Capacitor 22 uF/16 V eletrolítico	0,50	0,50
3	Led 3 mm vermelho	0,20	0,60
1	Receptáculo USB tipo B	2,00	2,00
2	Conector MOLEX 5 vias 5045	0,50	1,00
2	Conector MOLEX 4 vias 5045	0,50	1,00
5	Resistor 1 kΩ ¼ W	0,10	0,50
5	Resistor 10 kΩ ¼ W	0,10	0,50
1	Resistor 33 kΩ ¼ W	0,10	0,10
1	INA121	20,00	20,00
1	TL084	1,00	1,00
1	PIC18F14K50 I/P	12,00	12,00
1	Cristal 12 MHz	1,00	1,00
1	Capacitor eletrolítico 1uF/16V	0,50	0,50
3	Capacitor eletrolítico 470uF/16V	2,00	6,00
3	Diodo 1N4007	0,20	0,60
3	Indutor 100 mH axial	1,00	3,00
1	MCP1541 I/TO	3,00	3,00
1	dsPIC30F2020	30,00	30,00
1	MCP4822	16,00	16,00
1	Cristal 10 MHz	1,00	1,00
Total:			103,10

Tabela 17: Tabela de custos de material da placa de controle

Fonte: Elaborado pelo autor

2.15 Layout de circuito impresso

O primeiro layout de circuito impresso feito para a placa de controle foi feito usando-se o roteamento automático disponível no Altium Designer, obtendo desta forma o circuito apresentado na próxima figura.

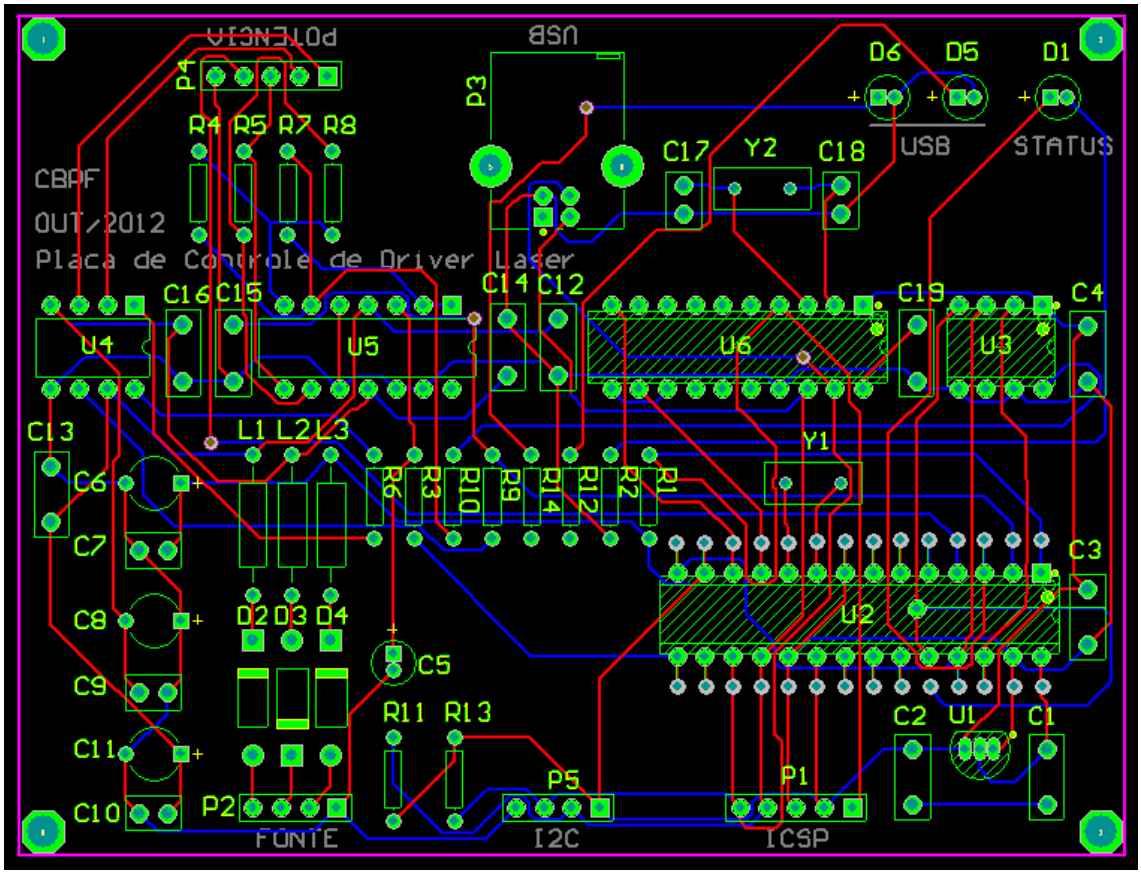

Figura 59: Layout da placa de controle com roteamento automático

Fonte: Elaborado pelo autor

Dada a quantidade de componentes neste circuito, o roteamento feito está em face dupla.

O orçamento realizado para produção de um lote piloto de 3 placas em Outubro de 2012 ficou no valor de R$ 360,00.

3. Driver de Potência

O driver de potência é responsável por agrupar a parte referente a alimentação do laser, sendo uma interface entre a placa de controle e a carga propriamente dita.

3.1 Alimentação da placa

A alimentação da placa está sendo feita através de uma entrada DC onde nesta teremos as tensões disponíveis de +12V e GND. Na próxima figura está apresentado o bloco referente a alimentação do circuito.

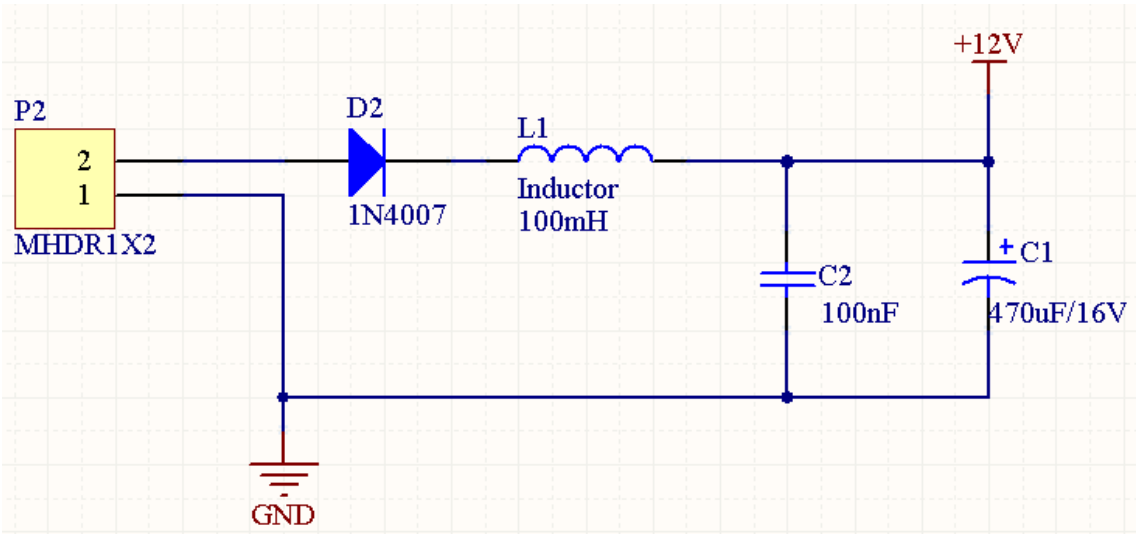

Figura 60: Esquema elétrico da parte de alimentação

Fonte: Elaborado pelo autor

Os indutores e capacitores foram utilizados para manter a tensão o mais estável possível, evitando variações bruscas de tensão em função da operação do driver. O diodo de proteção D2 é utilizado para que não ocorra inversão de polaridade que leve a placa a ficar danificada.

3.2 Saída para o laser

O esquema referente ao controle de potência do laser está apresentado na figura a seguir, onde observa-se o FET IRF730 sendo usado como fonte de corrente em sua região

linear de operação com a entrada CORRENTE alterando sua tensão V_{GS} e assim a corrente fornecida ao laser D1.

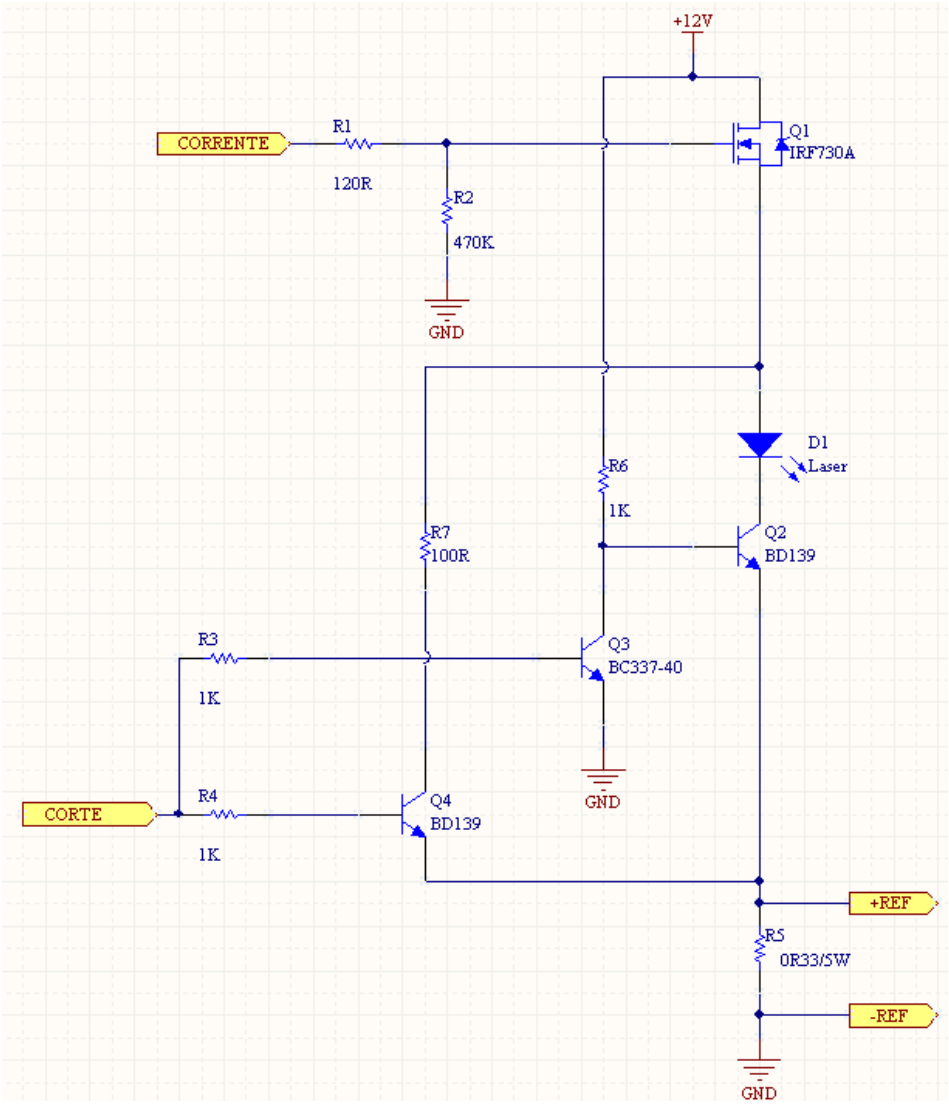

Figura 61: Esquema da parte de potência do laser

Fonte: Elaborado pelo autor

O transistor Q2 em série ao laser é usado para realizar o corte no mesmo, onde o pino CORTE poderá ser usado tanto para pulsar a corrente aplicada ao laser quanto também cortar sua alimentação. O transistor em paralelo Q4 em série com a resistência de 100Ω R7 é utilizado para no momento em que estiver pulsando o laser em uma dada frequência a impedância do circuito não altere de maneira significativa e fique praticamente no mesmo valor da resistência do laser, que é da ordem de 150 Ω. O transistor Q3 é utilizado para inverter o estado da entrada CORTE de forma a garantir que enquanto Q2 esteja saturado o

transistor Q4 esteja no corte e vice-versa. O resistor R5 é o de shunt, usado para fornecer a placa de controle o feedback da corrente consumida pelo laser.

3.3 Conexão com a placa de controle

A conexão com a placa de controle é feita através de um conector que permite obter o feedback de corrente, controle de corrente e corte do laser. A figura a seguir apresenta o conector utilizado para conectar com a placa de controle.

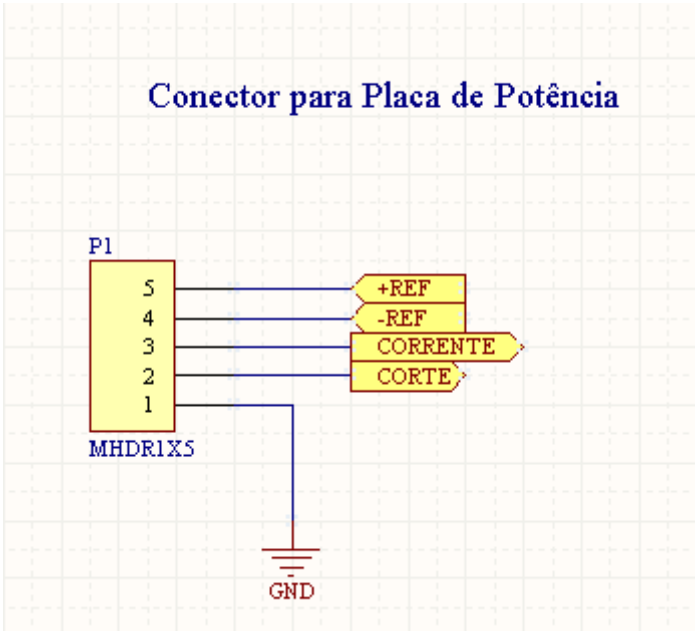

Figura 62: Esquema do conector de controle do laser

Fonte: Elaborado pelo autor

3.4 Esquema elétrico completo

Após a apresentação em partes do esquema, na próxima página está apresentado o esquema da placa driver de potência de forma integrada, permitindo assim observar de forma generalizada o funcionamento desta etapa do projeto.

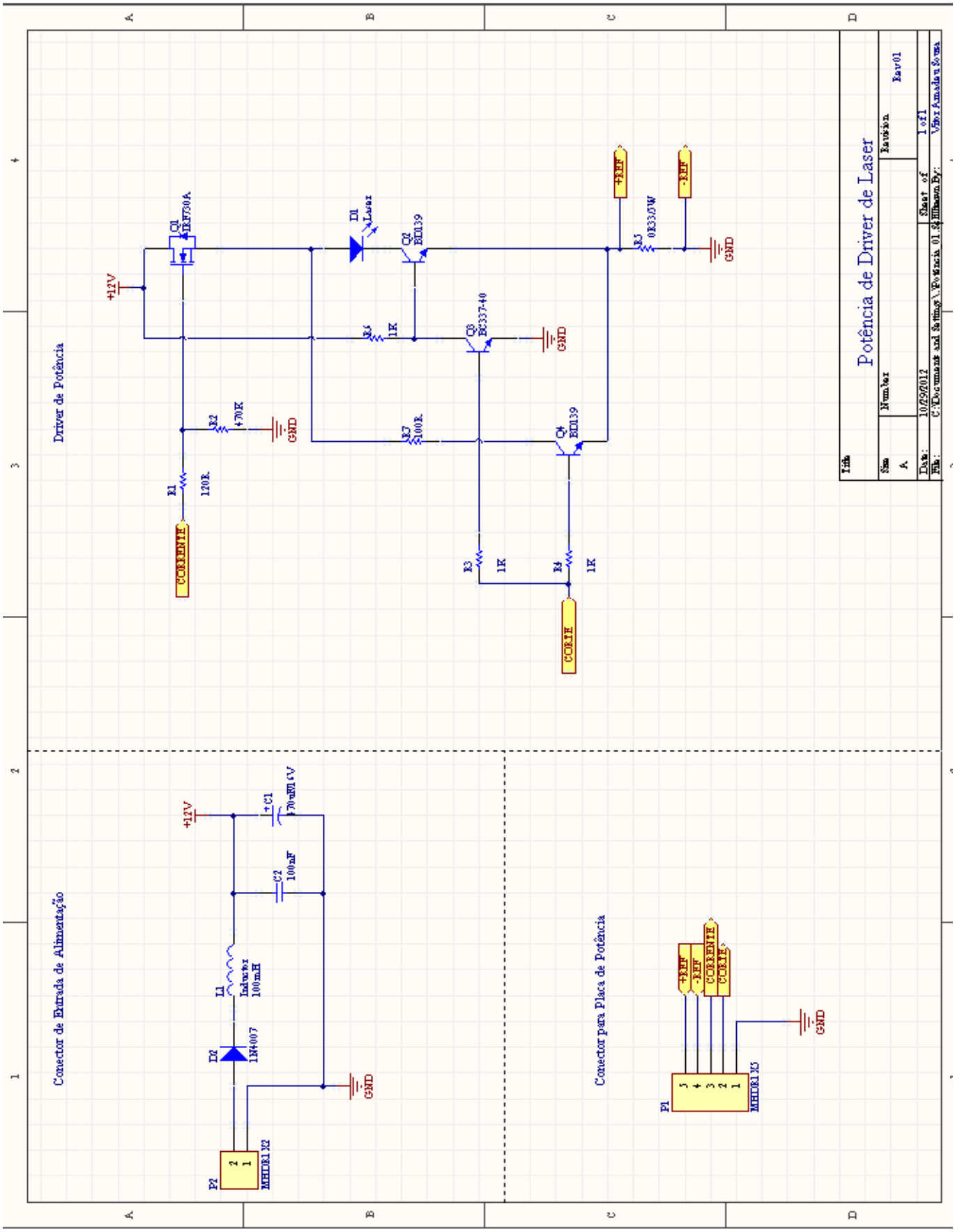

Figura 63: Esquema completo da placa de potência

Fonte: Elaborado pelo autor

3.5 Lista de material

A lista de material para o desenvolvimento da placa de potência está apresentado na tabela abaixo.

Quantidade	Descrição	Referência
1	Diodo 1N4007	D2
2	Transistor BD-139	Q2, Q4
1	Transistor BC337	Q3
1	Capacitor cerâmico 104	C2
1	Capacitor eletrolítico 470 uF/16V	C1
1	Indutor 100mH	L1
1	FET IRF730	Q1
1	Led vermelho 3mm	D1
1	Conector MOLEX 2 vias 5045	P2
1	Conector MOLEX 5 vias 5045	P1
1	Resistor 120 Ω ¼ W	R1
1	Resistor 470 kΩ ¼ W	R2
4	Resistor 1 kΩ ¼ W	R3, R4, R6, R7
1	0,33 Ω 5 W	R5

Tabela 18: Lista de material da placa de potência

Fonte: Elaborado pelo autor

Segundo cotações de preços obtidas em Outubro de 2012, o valor a nível de componentes para construção da placa de controle está apresentado a seguir.

Quantidade	Descrição	Valor unitário (R$)	Valor total (R$)
1	Diodo 1N4007	0,20	0,20
2	Transistor BD-139	2,00	4,00
1	Transistor BC337	0,50	0,50
1	Capacitor cerâmico 104	0,20	0,20
1	Capacitor eletrolítico 470 uF/16V	2,00	2,00
1	Indutor 100mH	1,00	1,00
1	FET IRF730	6,00	6,00
1	Led vermelho 3mm	0,20	0,20
1	Conector MOLEX 2 vias 5045	0,50	0,50
1	Conector MOLEX 5 vias 5045	0,50	0,50
1	Resistor 120 Ω ¼ W	0,10	0,10
1	Resistor 470 kΩ ¼ W	0,10	0,10
4	Resistor 1 kΩ ¼ W	0,10	0,40
1	0,33 Ω 5 W	2,00	2,00
Total:			17,70

Tabela 19: Tabela de custos de material da placa de potência

Fonte: Elaborado pelo autor

3.6 Layout de circuito impresso

O primeiro layout de circuito impresso feito para a placa de potência foi feito usando-se o roteamento automático disponível no Altium Designer, obtendo desta forma o circuito apresentado na próxima.

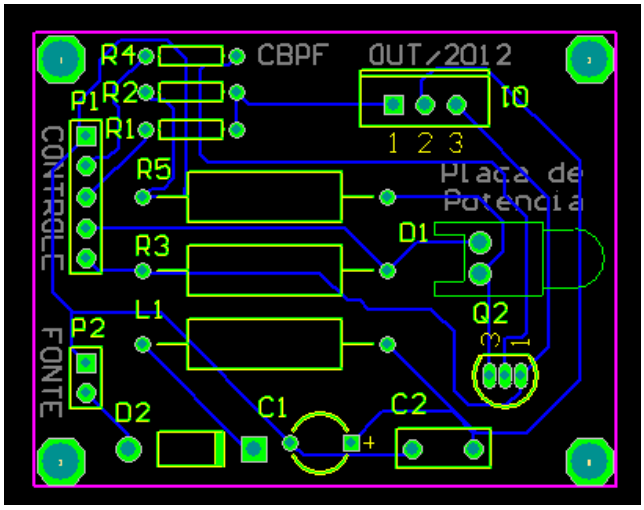

Figura 64: Layout da placa de potência com roteamento automático

Fonte: Elaborado pelo autor

Dada a simplicidade do circuito, foi possível rotear o mesmo em face simples.

O orçamento realizado para produção de um lote piloto de 3 placas em Outubro de 2012 ficou no valor de R$ 120,00.

3.7 Curvas levantadas na carga

O laser apresenta uma resistência típica da ordem de 150 Ω que varia conforme o seu funcionamento, apresentando assim uma resistência dinâmica. Para realizar os testes e evitar que o laser venha a danificar, foi utilizado um resistor de potência de mesma faixa de resistência para realizar as simulações. Conforme foi apresentado no esquema elétrico, há um circuito composto por transistores que funcionam em modo paralelo que permite pulsar o laser sem alterar significativamente a resistência do circuito, já que há dois circuitos paralelos com a mesma impedância onde conforme é pulsado o laser a corrente não altera de forma significativa, evitando assim picos de corrente. A seguir está apresentado algumas curvas que foram levantadas com o auxílio de um osciloscópio digital em diversas frequências alterando em alguns casos o ciclo ativo. Observa-se que a partir da frequência de 50 kHz, o comportamento da curva de corte passa a responder de maneira errada, onde isso se deve ao fato de estar sendo utilizado transistores comuns que possuem baixa frequência de chaveamento. Como proposta para sanar este problema, está sendo verificado a substituição de tais transistores por FETs de maior velocidade de chavemento como o IRLZ44N[44] que funciona na faixa de 1 MHz.

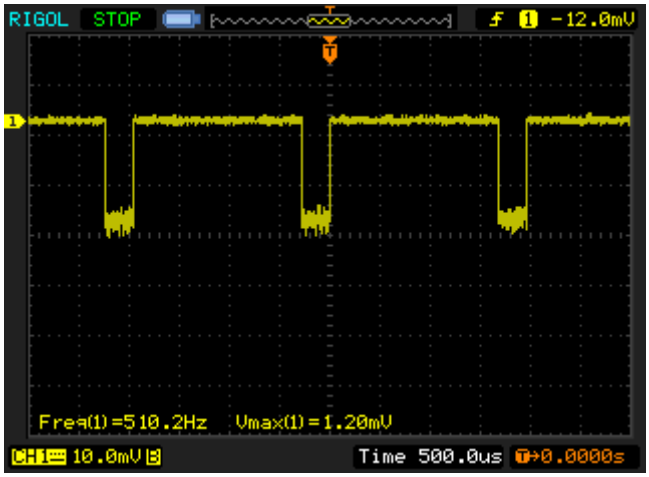

Figura 65: Forma de onda para frequência de 510 Hz com ciclo ativo de 80%

Fonte: Elaborado pelo autor

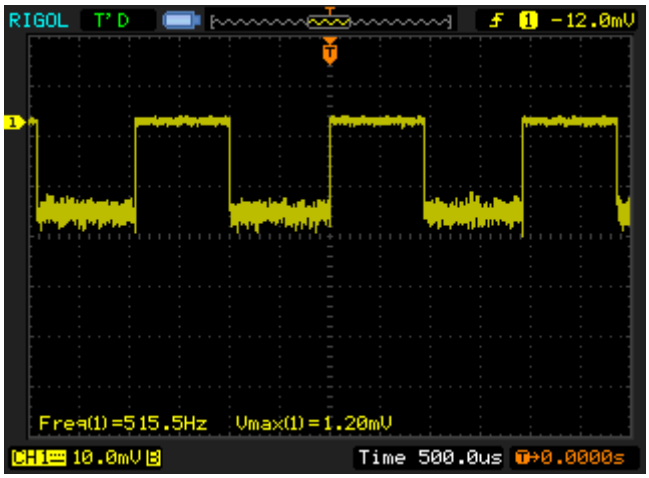

Figura 66: Forma de onda para frequência de 510 Hz com ciclo ativo de 50%

Fonte: Elaborado pelo autor

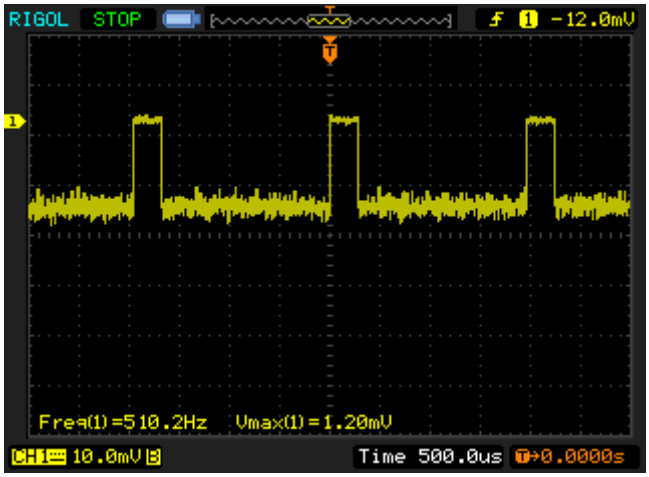

Figura 67: Forma de onda para frequência de 510 Hz com ciclo ativo de 20%

Fonte: Elaborado pelo autor

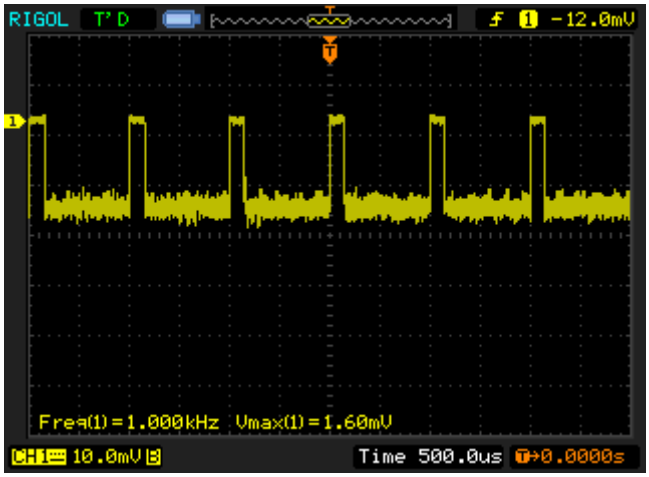

Figura 68: Forma de onda para frequência de 1kHz com ciclo ativo de 20%

Fonte: Elaborado pelo autor

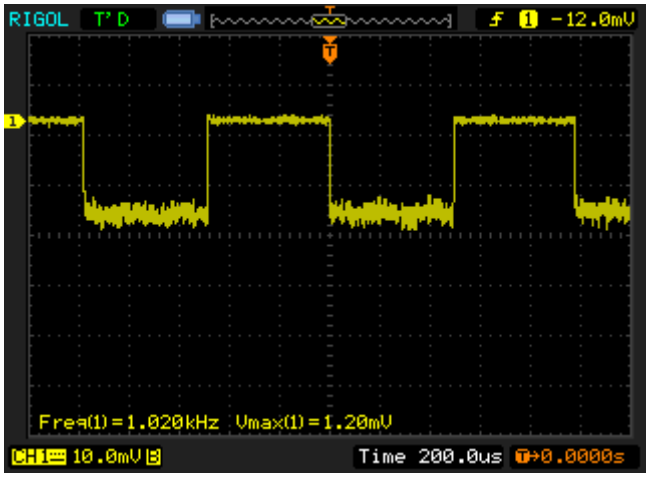

Figura 69: Forma de onda para frequência de 1kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

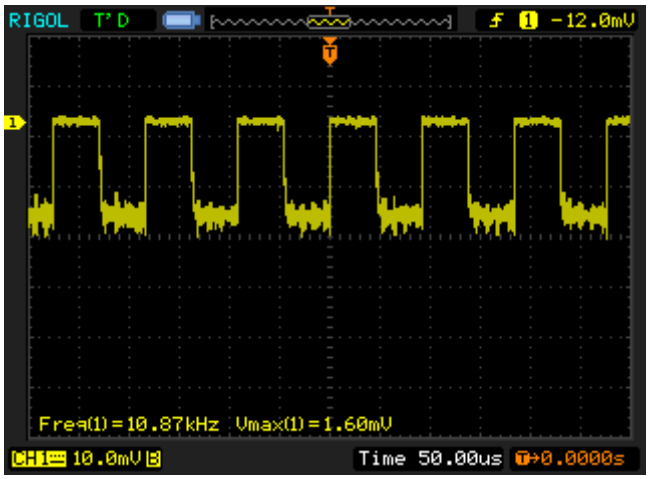

Figura 70: Forma de onda para frequência de 10,87kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

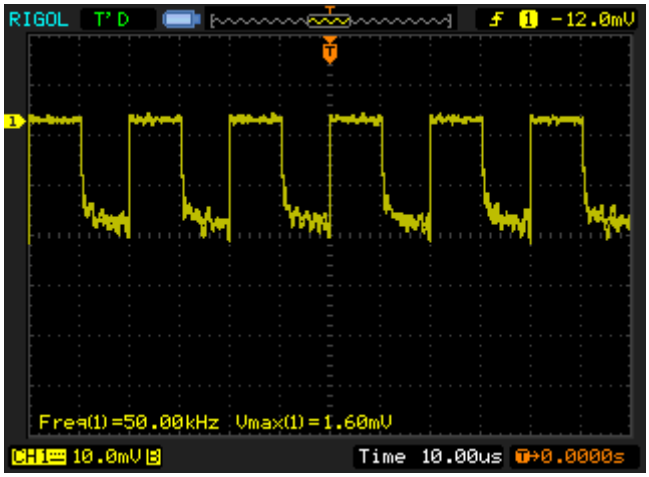

Figura 71: Forma de onda para frequência de 50kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

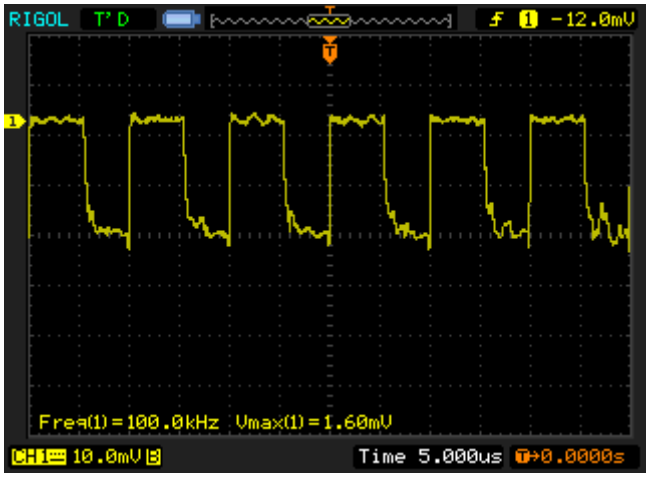

Figura 72: Forma de onda para frequência de 100kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

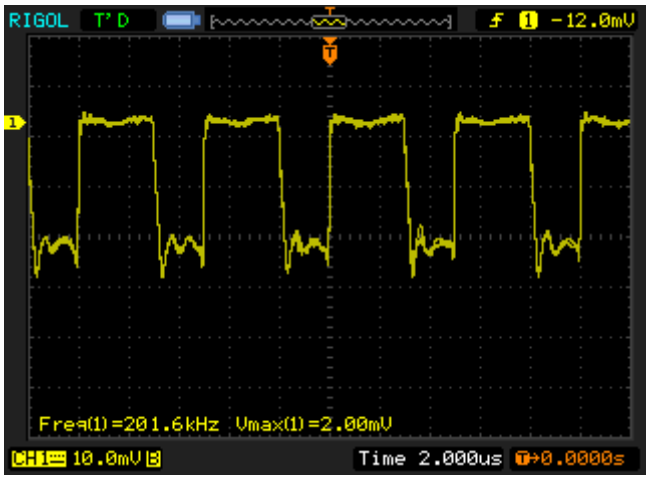

Figura 73: Forma de onda para frequência de 201kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

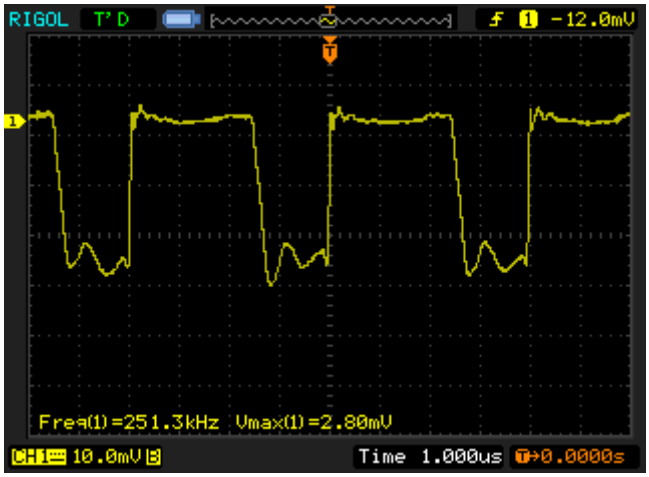

Figura 74: Forma de onda para frequência de 250kHz com ciclo ativo 50%

Fonte: Elaborado pelo autor

4. Controle PID

Um sistema de controle [45][46][47] é aquele que procura manter uma variável de determinado processo em um valor previamente configurado em que este é chamado de setpoint. O setpoint é o valor que se espera que uma variável tenha através da atuação do controlador PID. Digamos que, por exemplo, esperemos que a temperatura de um ambiente seja de 25º C, neste caso este seria o setpoint parametrizado. Para realizar tal tarefa encontramos o controle manual e o automático, em que o primeiro é aquele em que a ação de controle é realizada por meio de um observador que tem determinado grau de conhecimento necessário para o processo. Tal controle é feito por inspeção visual e em seguida são realizados os ajustes necessários em caso de divergência do valor esperado. A figura abaixo explana este conceito.

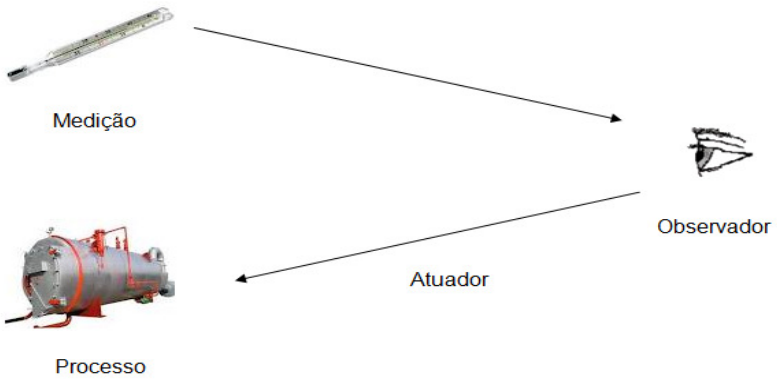

Figura 75: Controle Manual

Fonte: Elaborado pelo autor

Tal processo costuma ser caro além de trazer consigo o fator imprecisão já que dependerá da decisão e resposta do observador. Para evitar tal imprecisão, temos o controle automático em que o controle é realizado sem a participação direta do observador e a inspeção é feita através de instrumentos e as decisões tomadas baseadas nesta resposta e na configuração feita no controlador. Desta forma, a estrutura de controle permite manter automaticamente as variáveis de saídas dentro da vizinhança do valor esperado. As variáveis envolvidas no controle são aquelas que fornecem resultados provenientes de medidas realizadas no sistema, neste caso a temperatura. O sistema de controle manual apresentado na figura anterior ficaria da forma apresentada na figura abaixo com controle automático.

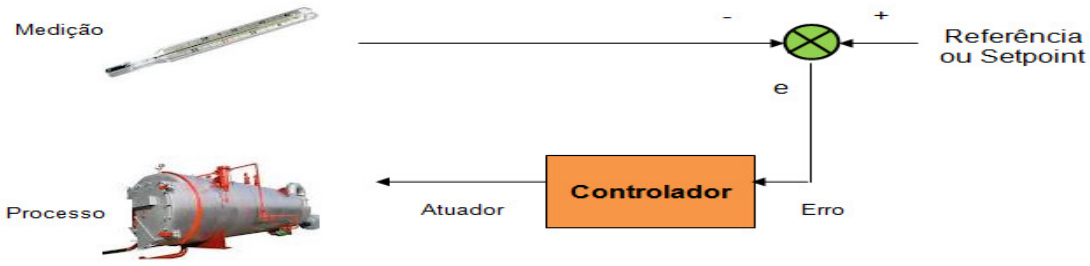

Figura 76: Controle Automático

Fonte: Elaborado pelo autor

Notamos a presença de um sinal de erro que é obtido através da diferença do setpoint pelo valor real de medição. Tal valor é fornecido ao controlador, que mediante ajustes parametrizados no mesmo, poderá ajustar o atuador para que a faixa de valor fique o mais próximo possível do valor de referência, diminuindo assim o erro. A representação básica de um controle realimentado pode ser visto como o apresentado na figura a seguir.

Figura 77: Exemplo de Controle Automático

Fonte: Elaborado pelo autor

4.1 Teoria de Controle PID

No controlador poderemos ajustar três constantes chamadas de Proporcional, Integral e Derivativo denominando assim o chamado controlador PID. Através destas combinações será possível controlar uma variável de forma a deixarmos mais próxima do setpoint esperado. Podemos representar um controlador PID através da equação apresentada na próxima tabela.

$$u(t) = k_p\left(e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \right)$$

Tabela 20: Equação do PID

Em que $u(t)$ é a resposta do sistema em que a saída será a velocidade do ventilador. O valor $e(t)$ é o erro, que será encontrado através da diferença do setpoint com a corrente real. As constantes k_p, k_i e k_d irão determinar a velocidade de resposta do controlador de acordo com os valores previamente configurados. Apesar das três ações disponíveis, não é necessário em si usar todas elas simultaneamente. Assim sendo, podemos ter as seguintes configurações para o controlador PID: Proporcional (P); Proporcional-Integral (PI); Proporcional-Derivativo (PD) e Proporcional-Integral-Derivativo (PID). Cada um destas configurações propiciará uma resposta diferente ao controlador em que seu emprego deverá ser avaliado de acordo com o problema a ser solucionado. A combinação das ações proporcional, integral e derivativa usadas para gerar um só sinal de controle, dá origem ao controlador proporcional-integral-derivativo ou simplesmente PID. O objetivo é aproveitar as características particulares de cada uma destas ações a fim de se obter uma melhora significativa do comportamento transitório e em regime permanente do sistema controlado. Desta forma, têm-se três parâmetros de sintonia no controlador, a saber: O ganho proporcional (ação proporcional); O tempo integral (ação integral) e o tempo derivativo (ação derivativa).

4.2 Controlador Proporcional

Diversos processos simples podem ser controlados facilmente somente com o ganho proporcional. A próxima equação demonstra a equação caso este tipo de controle seja feito.

$$u(t) = k_p\left(e(t)\right)$$

Tabela 21: Equação do Controlador P

Neste caso temos o erro (que é a diferença do valor atual medido do setpoint) multiplicado pela constante k_p (referindo-se ao ganho proporcional). O resultado é aplicado na saída do atuador, fazendo desta forma que o erro seja anulado ou diminuído.

4.3 Controlador Proporcional-Integral (PI)

O controlador PI é utilizado normalmente quando, por exemplo, a atuação do controlador proporcional não consegue diminuir o erro. Neste caso temos um erro que multiplicado pelo k_p, dá o quanto se deve atuar no ventilador para que o mesmo venha a arrefecer o resistor. Porém, se no passar do tempo, a atuação do ventilador não for suficiente para fazer com que o resistor venha a arrefecer somente neste caso utilizando o controle proporcional o controle integral poderá fazer com que o atuador fique mais potente e assim atue neste sentido. Utilizando neste caso o controlador PI, ao passar o tempo, o erro medido será integrado (somado) crescendo a cada verificação do erro. Neste caso, juntando o controle proporcional com o integral a resposta do sistema tenderá a aumentar em função do tempo já que cada erro é somado diferente do controle proporcional que depende apenas do erro momentâneo. A descrição da equação do controlador PI está apresentada na próxima tabela.

$$u(t) = k_p\left(e(t) + k_i\int_0^t e(t)dt\right)$$

Tabela 22: Equação do Controlador PI

4.4 Controlador Proporcional-Derivativo (PD)

A saída de um processo apresenta, intuitivamente, certa inércia com relação às modificações na variável de entrada. Este retardo explica-se pela dinâmica do processo que faz com que uma mudança na variável de controle provoque uma mudança considerável na

saída da planta somente após certo tempo. Outra interpretação é que, dependendo da dinâmica do processo, o sinal de controle estará em atraso para corrigir o erro. Este fato é responsável por transitórios com grande amplitude e período de oscilação, podendo, em um caso extremo, gerar respostas instáveis. A ação derivativa quando combinada com a ação proporcional tem justamente a função de antecipar a ação de controle a fim de que o processo reaja mais rápido. Neste caso, o sinal de controle a ser aplicado é proporcional a uma predição da saída do processo. A estrutura básica do controlador PD é dada pela expressão abaixo.

$$u(t) = k_p\left(e(t) + k_d \frac{de(t)}{dt} \right)$$

Tabela 23: Equação do Controlador PD

4.5 Controlador Proporcional-Integral-Derivativo (PID)

O controlador PID combina as vantagens do controlador PI e PD. A ação integral está diretamente ligada à precisão do sistema sendo responsável pelo erro nulo em regime permanente. O efeito desestabilizador do controlador PI é contrabalançado pela ação derivativa que tende a aumentar a estabilidade relativa do sistema ao mesmo tempo em que torna a resposta do sistema mais rápida devido ao seu efeito antecipatório. A função de transferência do controlador PID é dada pela equação da próxima tabela.

$$u(t) = k_p\left(e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \right)$$

Tabela 24: Equação do Controlador PID

Na etapa atual do projeto apenas o controle P está sendo usado com constante kp=1.

5. Fluxograma de Controle da Placa Driver Laser

O algoritmo da placa de controle[48] está apresentado nas próximas páginas, em que tal fluxo é baseado em uma máquina de estados que depende dos comandos enviados pelo computador via porta serial para dar início ao processo de rampa de subida, rampa de descida, feedback e controle para o funcionamento do laser. No bloco referente a rampa de subida a corrente parte do mínimo (0 mA) até o solicitado pelo software de controle via porta serial. Ao chegar neste ponto, o bloco de controle passa a operar mantendo a corrente dentro da faixa parametrizada. O bloco de rampa de descida faz o decaimento da corrente do valor configurado até o mínimo (0 mA). O bloco referente ao feedback reporta ao software de controle a corrente atual sendo consumida pelo driver, de modo que este possa plotar a curva de consumo do driver ao longo do tempo.

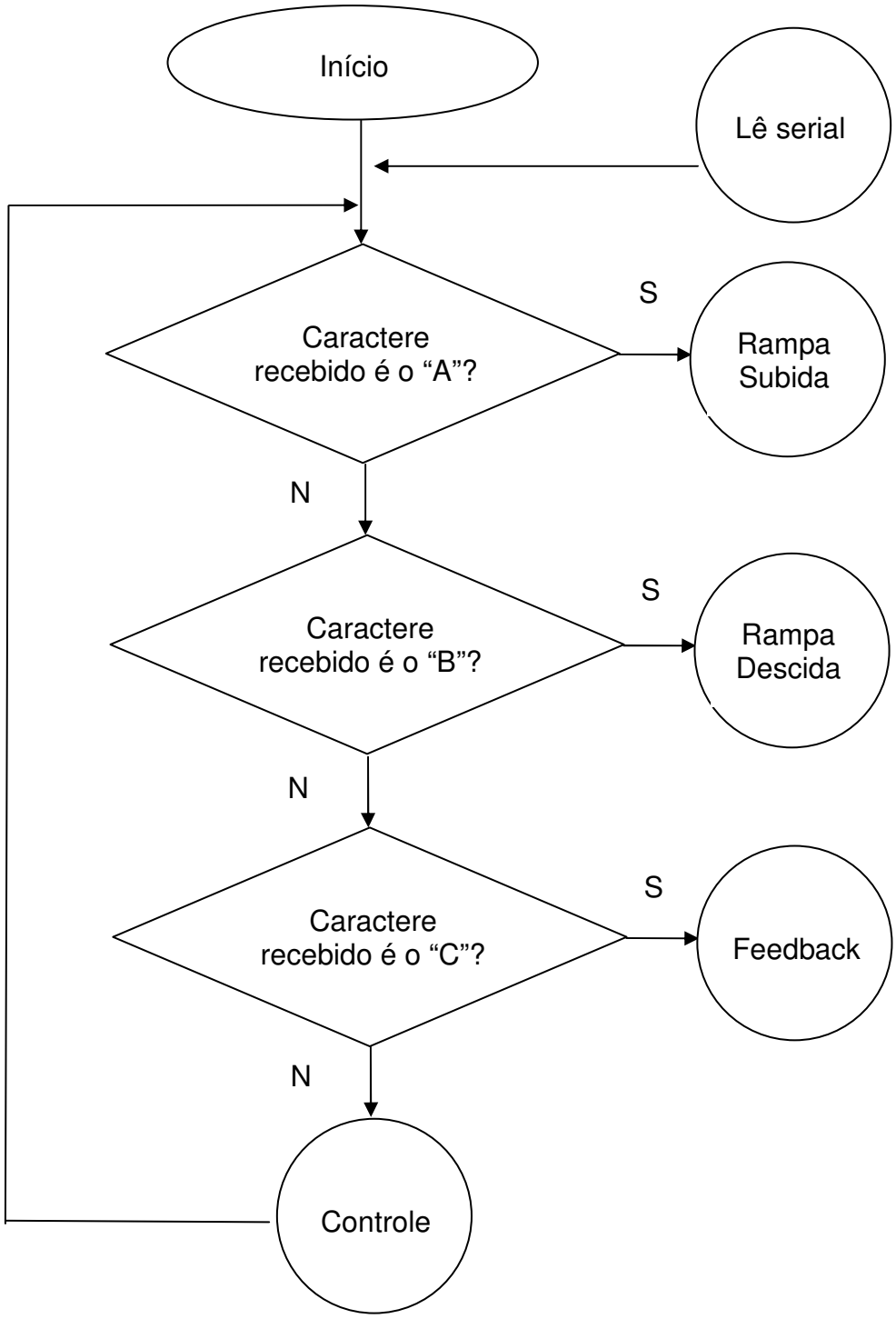

Figura 78: Fluxograma da placa de controle – Parte 1

Fonte: Elaborado pelo autor

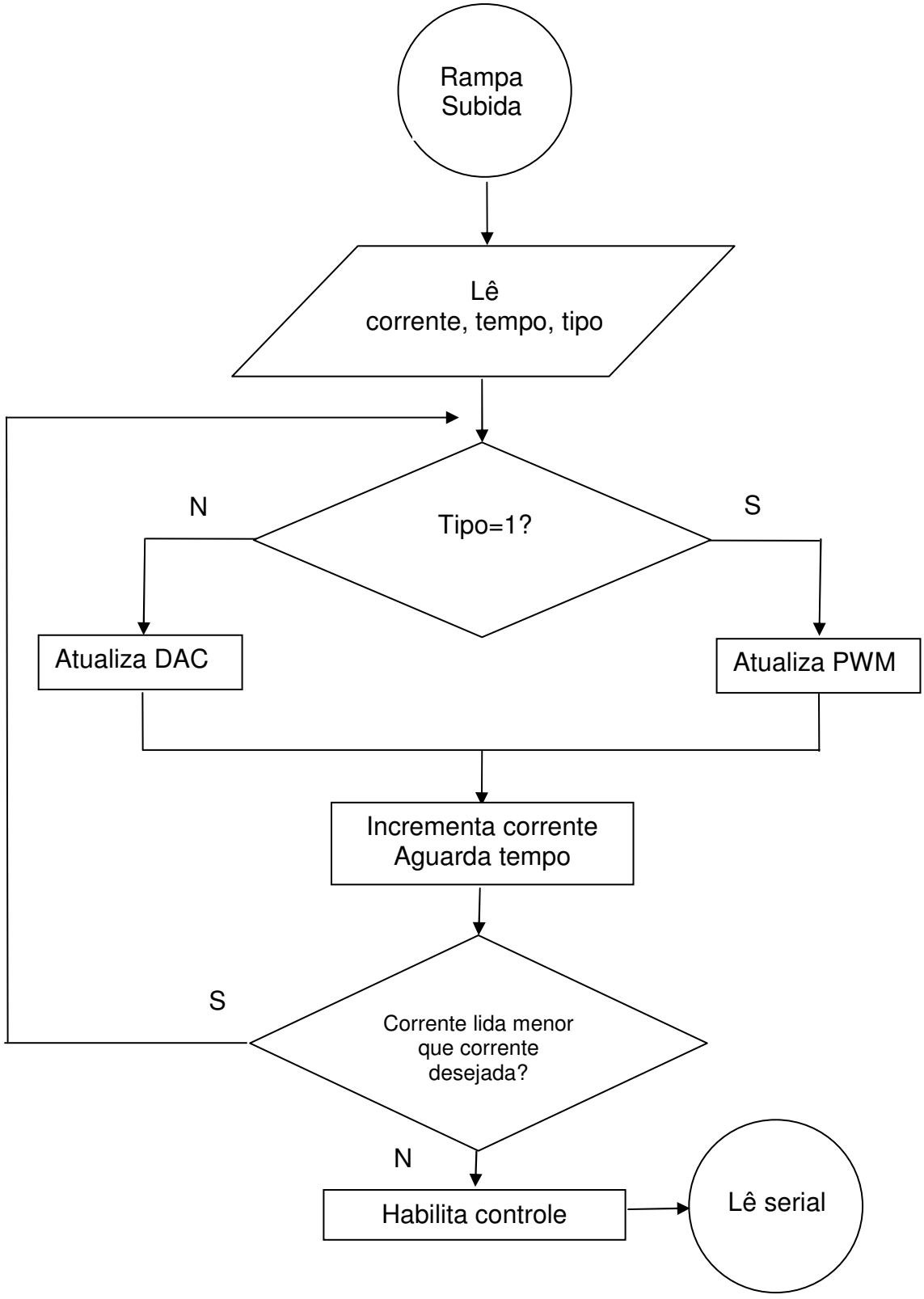

Figura 79: Fluxograma da placa de controle – Parte 2

Fonte: Elaborado pelo autor

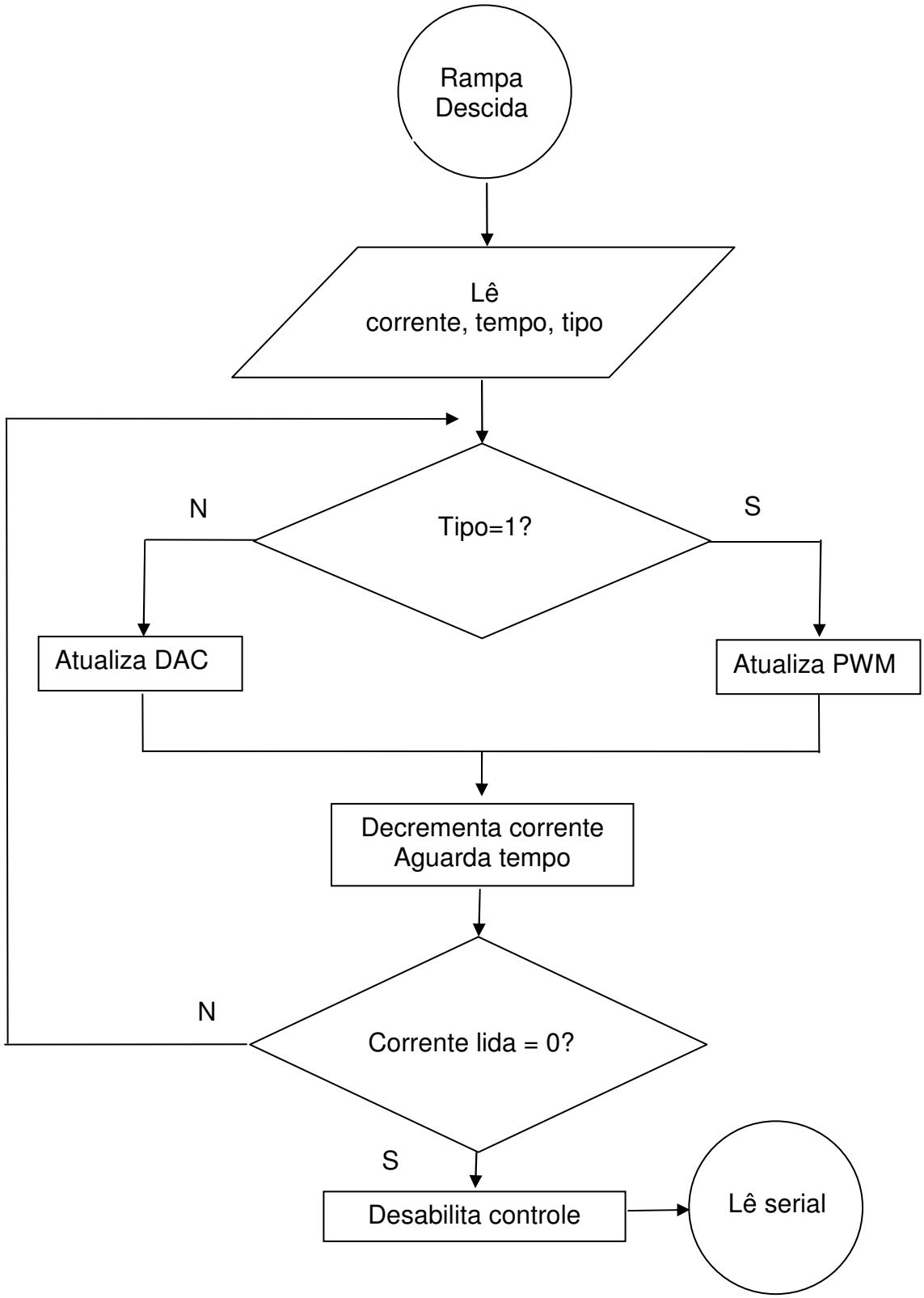

Figura 80: Fluxograma da placa de controle – Parte 3

Fonte: Elaborado pelo autor

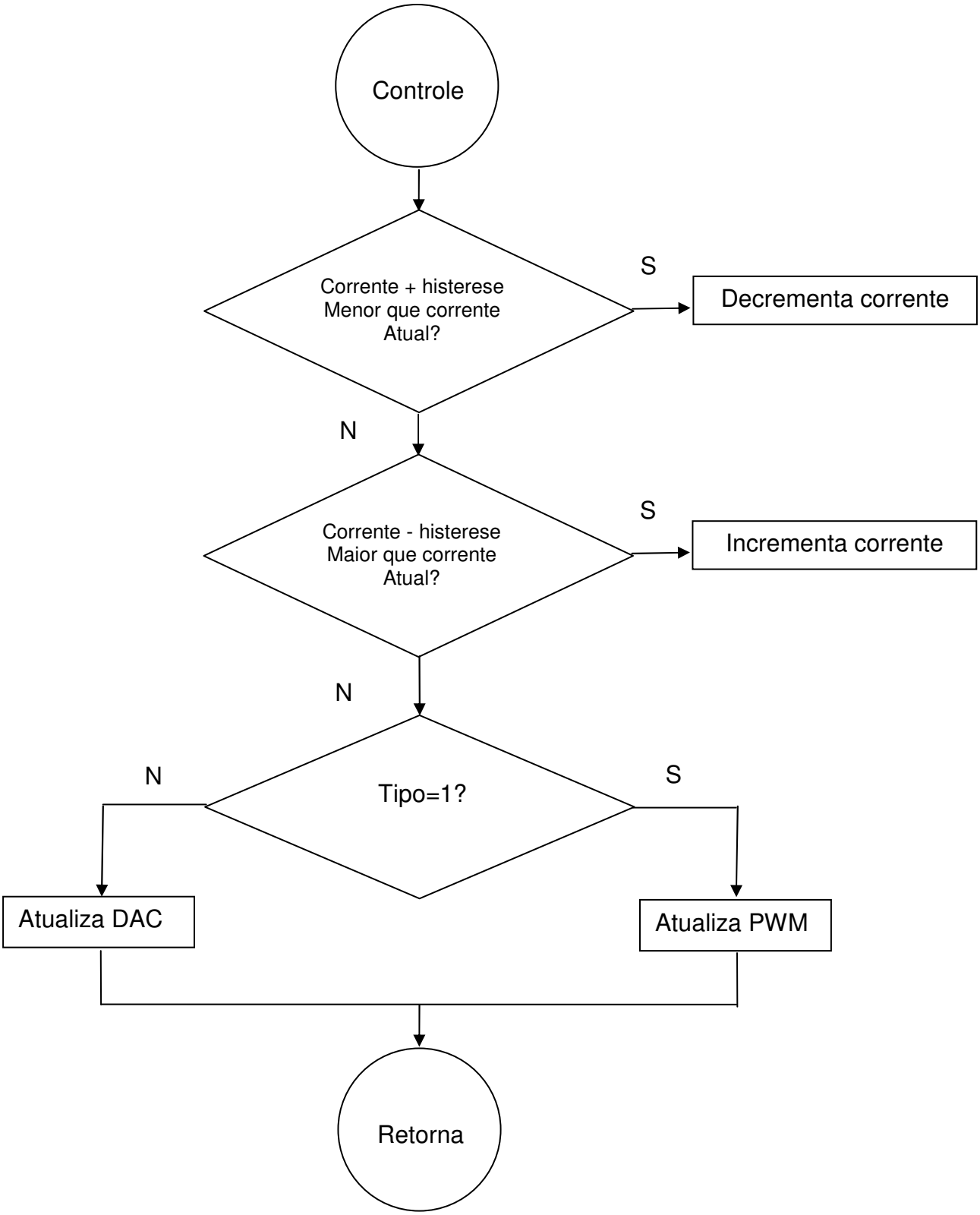

Figura 81: Fluxograma da placa de controle – Parte 4

Fonte: Elaborado pelo autor

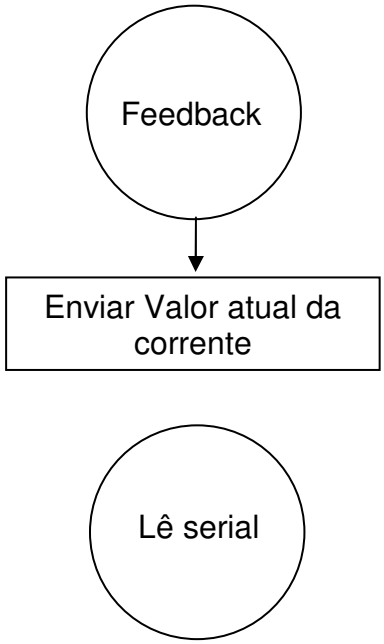

Figura 82: Fluxograma da placa de controle – Parte 5

Fonte: Elaborado pelo autor

Pelo fato da placa de potência ser composta apenas de hardware, não há algoritmo para a sua operação, já que todo o funcionamento é controlado pela placa de controle.

6. Software de Controle da Placa Driver Laser

O software de controle foi implementado na linguagem C [49][50][51][52][53][54][55][56][57] baseado no compilador C30 da Microchip. O mesmo segue o funcionamento apresentado no fluxograma anterior e está apresentado de forma segmentada na próxima tabela.

```
/*   *     *     *     *     *     *     *     *     *     *     *     *     *     *
 *               CBPF - Centro Brasileiro de Pesquisas Físicas                      *
 *                                Driver de Laser                                   *
 *                         Orientador: Pablo Diniz Batista                          *
 *                          Orientando: Vitor Amadeu Souza                          *
 *                              Início: 10/09/2012                                  *
 *   *     *     *     *     *     *     *     *     *     *     *     *     *     */

#include <P30f2020.h>             //Arquivo cabeçalho utilizado
#include <uart.h>
#include <string.h>
```

```
/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 *                          Bits de Configuração                              *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

_FOSCSEL(PRIOSC_PLL);
_FOSC(CSW_FSCM_ON & HS);                //Fonte de clock do tipo HS
_FWDT(FWDTEN_OFF);                      //Watchdog desligado
_FGS(CODE_PROT_OFF);                    //Código de proteção off
_FPOR(PWRT_128)
_FBS(BSS_NO_FLASH)

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 *                           Declaração de Saídas                             *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

#define          LED1        PORTBbits.RB1

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 *                          Definição de Constantes                           *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

#define HISTERESE     1

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 *                           Definição de Variáveis                           *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

unsigned char faz_controle=0,trata_comando=0;
unsigned char corrente_p,tempo_p,tipo_p;
unsigned int valor_pwm=0,valor_corrente;
unsigned char indice=0;
unsigned char buffer[]={0,0,0,0,0,0,0,0,0,0};
unsigned long timeout=0;

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 *                          Prototipagem das Funções                          *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

void delay_ms(unsigned int tempo);
void Rampa_Subida(unsigned char corrente, unsigned char tempo,unsigned char tipo);
void Controle_p(void);
void Rampa_Descida(unsigned char corrente, unsigned char tempo,unsigned char tipo);
void __attribute__ ((__interrupt__)) _U1RXInterrupt(void);

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   Principal                                                       *
 * Entradas: Nenhum                                                          *
 * Saídas:   Nenhum                                                          *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

void main(void)
{
     Init_Rs232();                          //Inicializa a RS232
     Init_Voltage();                        //Inicializa a SPI
     Init_Pwm();                            //Inicializa o PWM
     Init_Adc();                            //Inicializa o ADC

     Set_Pwm(0);                            //Seta um valor no PWM
     Set_Voltage(0,1);                      //Seta valor da tensão
```

```
            do
            {
              //Tratamento de controle de corrente

              if(faz_controle==1)               //É para fazer o controle?
              {
                    LED1=1;                     //Sim, então liga o led
                    Controle_p();               //Chama função de controle
              }
              else
                  LED1=0;                       //Não, então apaga o led

              //Tratamento de comandos da serial

              if(trata_comando==1)
              {
                   trata_comando=0;

                   switch(buffer[0])
                   {
                        case 'A':
                              corrente_p=buffer[1];
                              tempo_p=buffer[2];
                              tipo_p=buffer[3];
                              Rampa_Subida(corrente_p,tempo_p,tipo_p);
                              break;
                        case 'B':
                              tempo_p=buffer[2];
                              Rampa_Descida(corrente_p,tempo_p,tipo_p);

                              break;
                   }
              }

              //Tratamento de timeout de comunicação

              timeout++;
              if(timeout>1000000)
              {
                    timeout=0;
                    indice=0;
              }

            }while(1);                          //Fica em loop infinito

}


/*   *     *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   Interrupção serial                                                    *
 * Entradas: Nenhum                                                              *
 * Saídas:   Nenhum                                                              *
 *   *     *     *     *     *     *     *     *     *     *     *     *     *     */

void __attribute__ ((__interrupt__)) _U1RXInterrupt(void)
{
      Trata_rs232();
}
```

```
/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   trata_rs232                                                       *
 * Entradas: Nenhum                                                          *
 * Saídas:   Nenhum                                                          *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

void Trata_rs232(void)
{
     buffer[indice]=U1RXREG;
     IFS0bits.U1RXIF=0;
     indice++;
     timeout=0;

     if(indice==5)
     {
           indice=0;

           if (buffer[4]==10)
           {
                 if(buffer[0]=='C')
                 {
                       putcUART1(valor_pwm>>8);      //MSB
                       putcUART1(valor_pwm);         //LSB
                       putcUART1(13);                //Terminador
                       putcUART1(10);                //Terminador
                 }
                 else
                       trata_comando=1;
           }
     }
}

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   le_                                                             *
 * Entradas: corrente (float), tempo_rampa (int), tipo (DAC=0,PWM=1)         *
 * Saídas:   Nenhum                                                          *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

unsigned int le_ad(void)
{
     unsigned int media_ad=0;
     char contador;

     for(contador=1;contador<=16;contador++)
     {
                 while(!IFS0bits.ADIF);
                 IFS0bits.ADIF=0;
                 ADSTATbits.P0RDY = 0;
                 media_ad=media_ad+ADCBUF0;
     }

     return(media_ad/16);

}
```

```
/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   Rampa_Subida                                                      *
 * Entradas: corrente (float), tempo_rampa (int),                              *
 *               lacos (int), tipo (DAC=0,PWM=1)                               *
 * Saídas:   Nenhum                                                            *
 * Controla corrente de 20mA a 200 mA                                          *
 * Equação: Xadc=4.25(Ycor-20)+76                                              *
 *   *     *     *     *     *     *     *     *     *     *     *     *     *  */

void Rampa_Subida(unsigned char corrente, unsigned char tempo,unsigned char tipo)
{

     valor_pwm=500;
     valor_corrente=4.8*(corrente-20)+76;

     //Rampa de subida

     do
     {
           if(tipo)
                 Set_Pwm(valor_pwm);
           else
                 Set_Voltage(valor_pwm,1);

           valor_pwm++;
           delay_ms(tempo);
     }while(le_ad()<valor_corrente);

     //Controle

     faz_controle=1;                                  //Habilita rotina de controle

}

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   Controle_p                                                        *
 * Entradas: Nenhum                                                            *
 * Saídas:   Nenhum                                                            *
 *   *     *     *     *     *     *     *     *     *     *     *     *     *  */

void Controle_p(void)
{

           //Controle

           if((valor_corrente+HISTERESE)<le_ad())
           {
                 valor_pwm--;
           }

           if((valor_corrente-HISTERESE)>le_ad())
           {
                 valor_pwm++;
           }

           if(tipo_p)
                 Set_Pwm(valor_pwm);
           else
                 Set_Voltage(valor_pwm,1);

}
```

```
/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   Rampa_Descida                                                     *
 * Entradas: corrente (float), tempo_rampa (int),                              *
 *               lacos (int), tipo (DAC=0,PWM=1)                               *
 *                                                                             *
 * Saídas:   Nenhum                                                            *
 * Controla corrente de 20mA a 200 mA                                          *
 * Equação: Xadc=4.25(Ycor-20)+76                                              *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */

void Rampa_Descida(unsigned char corrente, unsigned char tempo,unsigned char tipo)
{

     do
     {
           if(tipo)
                 Set_Pwm(valor_pwm);
           else
                 Set_Voltage(valor_pwm,1);

           valor_pwm--;
           delay_ms(tempo);
     }while(valor_pwm>500);

     valor_pwm=0;

     Set_Pwm(0);
     Set_Voltage(0,1);

     faz_controle=0;                              //Desabilita rotina de controle

}

/*   *     *     *     *     *     *     *     *     *     *     *     *     *
 * Função:   delay_ms                                                          *
 * Entradas: tempo                                                             *
 * Saídas:   Nenhum                                                            *
 *   *     *     *     *     *     *     *     *     *     *     *     *     */
void delay_ms(unsigned int tempo)
{
PR1=4999;  //O registrador de período é carregado com //este valor pois 5000 * 200
ns
                                        //(C.M em 5 MIPS) é igual a 1 ms
     T1CON=0b1000000000000000;          //Liga o timer1
                                        //Prescaler 1:1
     if (!tempo) return;                //Caso tempo tenha vindo em 0 retorna
     do
      {
           while(!IFS0bits.T1IF);       //Aguarda o timer estourar
           IFS0bits.T1IF=0;             //Reinicializa a variável de estouro
           tempo--;                     //Decrementa a variável de tempo
      }while(tempo!=0);
}
```

Tabela 25: Programa fonte da placa de controle

Fonte: Elaborado pelo autor

7. Software de Controle do Computador

Este software foi desenvolvido com o intuito de permitir parametrizar e assim testar o funcionamento da placa de controle de driver. O mesmo foi desenvolvido no Borland Delphi 7 [58][59] [60][61] [62] e possui a interface apresentada a seguir.

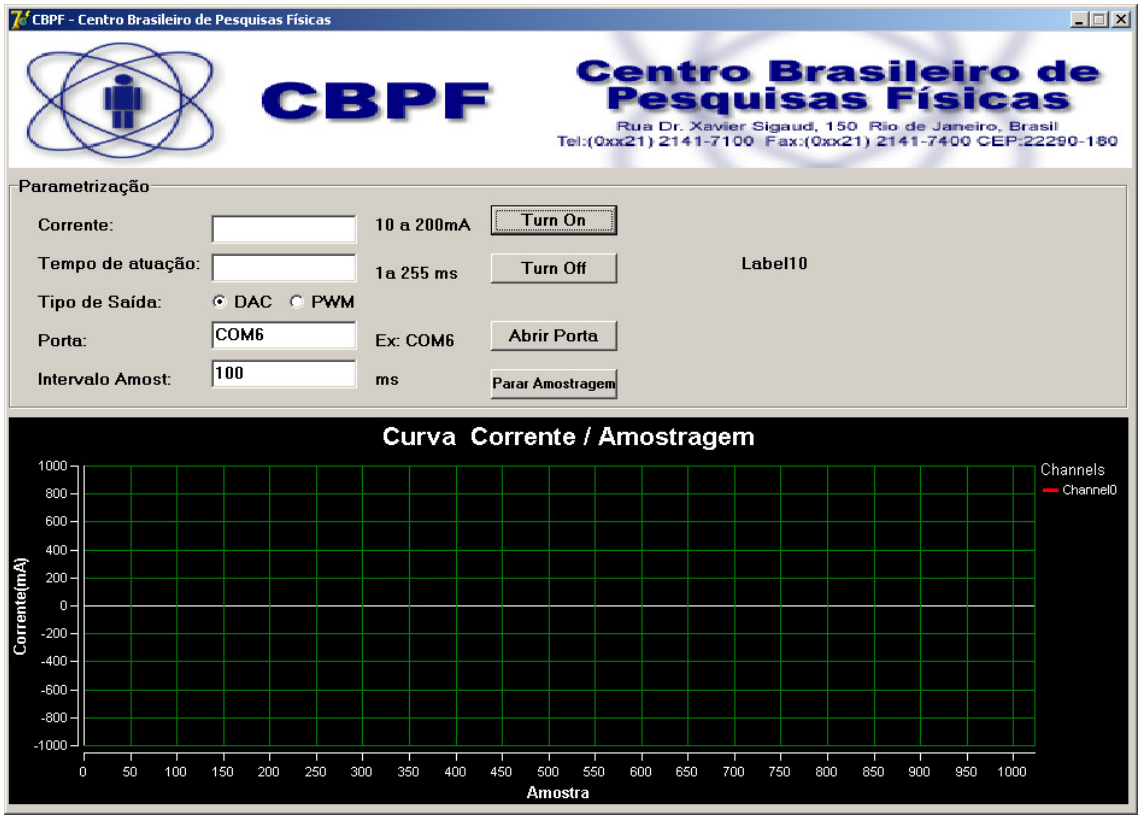

Figura 83: Tela gráfica do programa de testes

Fonte: Elaborado pelo autor

Através deste programa é possível selecionar a porta de comunicação RS232 que será utilizada para troca de informações com a placa de controle, a corrente especificada para o laser, que pode ser ajustada de 10 mA a 200 mA, o tempo de atuação da rampa de subida e descida que pode ser ajustada de 1 ms a 255 ms, o tipo de saída que irá controlar o FET de corrente onde neste caso pode ser selecionada a saída por DAC (Digital Analog Converter) ou PWM (Pulse Width Modulation). Finalmente o parâmetro Intervalo de Amostra permite configurar o tempo no qual o software ficará consultando o feedback de corrente para em seguida plotar o gráfico. A seguir está apresentado uma parametrização feita para uma corrente de 25 mA na placa de controle com tempo de atuação de 10 ms. Neste gráfico

observa-se a rampa de subida seguida da faixa estável de funcionamento até a rampa de descida de operação.

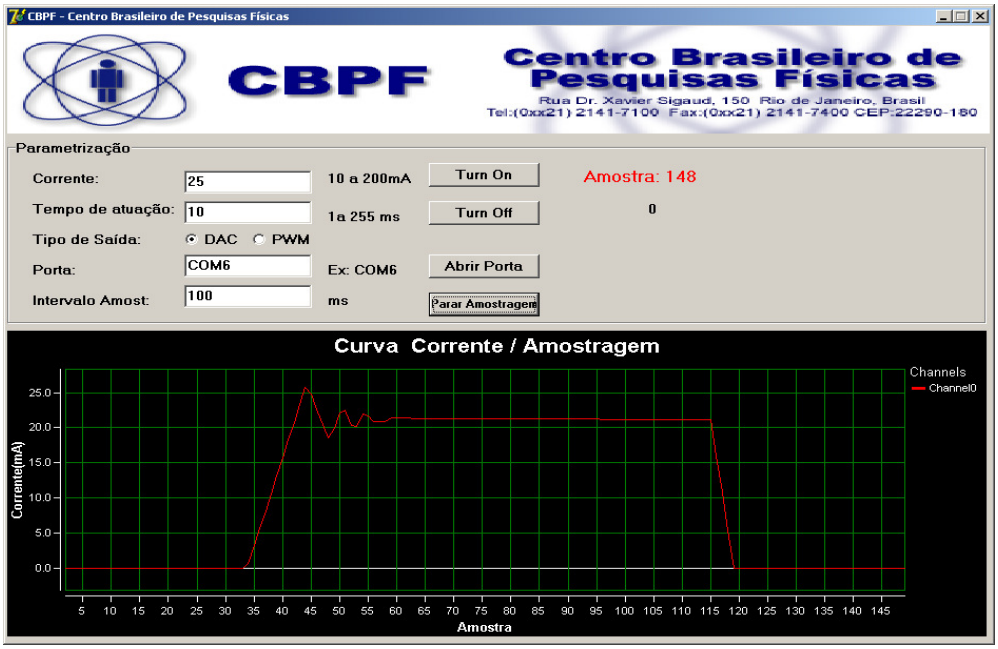

Figura 84: Simulação para uma corrente de 25 mA - 10 ms

Fonte: Elaborado pelo autor

Já a próxima figura apresenta a simulação para a mesma corrente, porém com tempo de atuação de 50 ms.

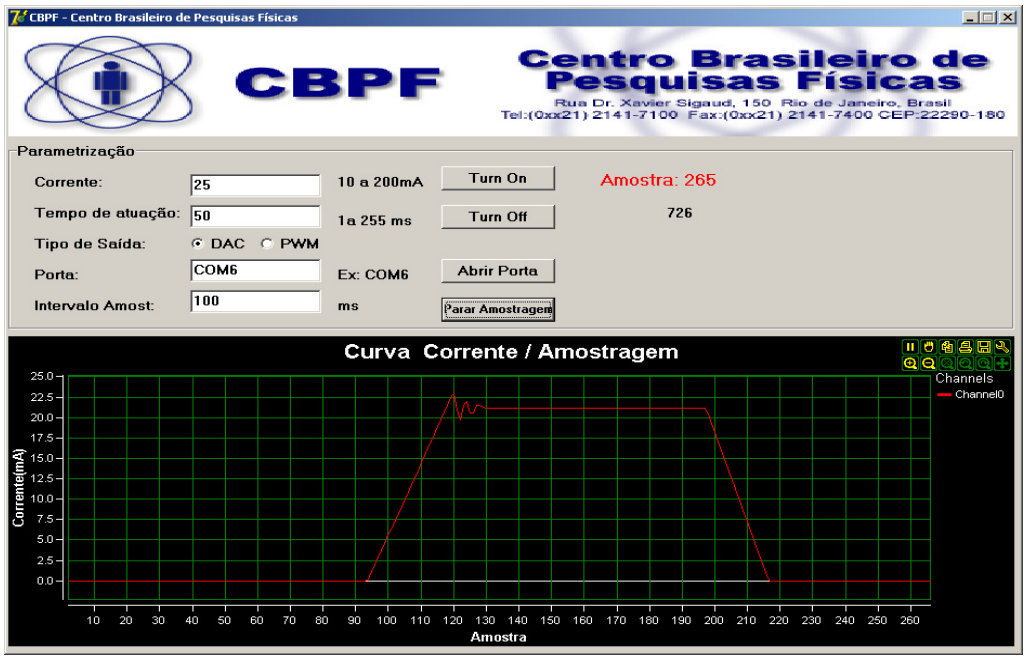

Figura 85: Simulação para uma corrente de 25 mA - 50 ms

Fonte: Elaborado pelo autor

O programa está baseado em eventos, onde cada botão pressionado gera uma ação diferente no programa. Por exemplo, ao pressionar o botão Abrir Porta é feita a abertura da porta serial de acordo com o parâmetro ajustado na caixa de texto. O código em Pascal a seguir apresenta tal procedimento no programa.

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  com.Port:=porta.Text;
  com.Open
end;
```

Tabela 26: Fonte do programa Delphi para abertura de porta serial

Fonte: Elaborado pelo autor

O botão Turn On envia um pacote de bytes para a placa de controle que permite a mesma dar início ao seu processamento. Este frame está organizado segundo o protocolo disponível na tabela abaixo.

“A”	CORRENTE	TEMPO	TIPO	10

Tabela 27: Protocolo de comunicação para iniciar o processo

Fonte: Elaborado pelo autor

O caracter “A” define o tipo de mensagem, neste caso de início de processo (turn on). O byte CORRENTE define a corrente que se deseja parametrizar na placa de controle, o byte TEMPO o tempo de atuação entre cada passo na placa de controle, o campo TIPO se a saída será para o DAC (0) ou PWM (1). Finalmente o valor final 10 refere-se ao término do protocolo, neste caso é o caracter Line Feed (LF) da tabela ASCII. A seguir o código em Delphi responsável por este trecho do programa.

```
procedure TForm1.Button2Click(Sender: TObject);
begin

        If ((strtoint(corrente.Text) < 10) Or (strtoint(corrente.Text) > 200))Then
        begin
            showmessage('Verifique a faixa correta de corrente!');
            exit;
        end;
```

```
        If ((strtoint(tempo.Text) < 1) Or (strtoint(tempo.Text) > 255)) Then
        begin
            showmessage('Verifique a faixa correta de tempo!');
            exit;
        end;

        If (porta.Text = '') Then
        begin
            showmessage('Preencha a porta de comunicação!');
            exit;
        end;

            //Prepara para fazer a amostragem

            graf.Channels[0].Data.clear;

            timer1.Interval :=strtoint(intervalo.text);
            timer1.Enabled:=true;     //Liga o timer para amostragem

            amostragem:=1;

            NomeDoLog:=extractfilepath(Application.ExeName)+
            '\Amostragem_Corrente_' + corrente.text + '_Tempo_'+ tempo.text + '.txt
            AssignFile(Arquivo, NomeDoLog);
            Rewrite(Arquivo);
            WriteLn(arquivo,'Amostragem'+chr(9)+'Corrente');

            //Protocolo: "A" + CORRENTE + TEMPO + TIPO (1 ->PWM, 0-> DAC) + CHR(10)

            COM.Send('A');
            COM.Send(Chr(strtoint(corrente.Text)));
            COM.Send(Chr(strtoint(tempo.Text)));

            If (dac.Checked = True) Then
                COM.Send(Chr(0))
            Else
                COM.Send(Chr(1));

            COM.Send(Chr(10));

end;
```

Tabela 28: Fonte do programa Delphi para inicializar o processo

Fonte: Elaborado pelo autor

O botão Turn Off possibilita encerrar o funcionamento do bloco de controle desligando o laser já que a corrente será ajustada para 0 mA. Da mesma forma que o botão de Turn On, há um protocolo estabelecido para este botão, que segue o apresentado na tabela a seguir.

“B”	0	TEMPO	TIPO	10

Tabela 29: Protocolo de comunicação para finalizar o processo

Fonte: Elaborado pelo autor

O caracter de início “B” define que a mensagem especificada é para encerrar o funcionamento do controle e assim gerar a rampa de descida no laser. O parâmetro TEMPO define o tempo de decaimento de cada passo na rampa de descida. O parâmetro TIPO se o desligamento será feito pelo DAC (0) ou PWM (1) e finalmente o parâmetro 10 é LF (Line Feed) que encerra o protocolo de comunicação. A seguir o código em Delphi responsável por este trecho do programa.

```
        COM.Send('B');
        COM.Send(Chr(0));
        COM.Send(Chr(strtoint(tempo.Text)));
        If (dac.Checked = True) Then
                COM.Send(Chr(0))
        Else
                COM.Send(Chr(1));
        COM.Send(Chr(10));
```

Tabela 30: Fonte do programa Delphi para finalizar o processo

Fonte: Elaborado pelo autor

Há um bloco de timer que periodicamente interrompe o programa de forma a solicitar a placa de controle o valor atual da corrente consumida de modo a plotar o gráfico da curva de consumo de corrente x amostragem. O valor deste intervalo de tempo é feito através da caixa de texto Intervalo de Amostragem. Da mesma forma que o botão de início e término de processo, há um protocolo estabelecido para este timer, que segue o apresentado na tabela a seguir.

“C”	0	0	0	10

Tabela 31: Protocolo de comunicação para feedback de corrente

Fonte: Elaborado pelo autor

Periodicamente, o timer fica enviando este frame de modo que a placa driver retorne com a corrente mensurada naquele instante. A seguir está apresentado o trecho de código deste bloco de código em Delphi.

```
   //Protocolo: "C" + X + X+ X + CHR(10)
        COM.Send('C');
        COM.Send(Chr(0));
        COM.Send(Chr(0));
        COM.Send(Chr(0));
        COM.Send(Chr(10));
```

Tabela 32: Fonte do programa Delphi para feedback de corrente

Fonte: Elaborado pelo autor

Ao receber a resposta da placa via RS232, é gerado um evento de recepção na porta de comunicação serial que faz a captura da informação e plota em seguida no gráfico a mesma, de modo a podermos acompanhar em modo contínuo o consumo de corrente. A seguir o trecho de código responsável pela recepção serial no Delphi.

```
procedure TForm1.comReceiveCallBack(Data: String);
var
  corrente:real;

begin

  corrente:=ord(data[1]);
  corrente:=corrente*256;
  corrente:=corrente+ord(data[2]);
  label10.Caption:=floattostr(corrente);

  corrente:=(corrente-1159)/8;       //Obtém o valor em corrente
                                     //corrente:=(corrente-2335)/1.122;
  if(corrente<0)then
    corrente:=0;

  WriteLn(arquivo,inttostr(amostragem)+chr(9)+chr(9)+FormatFloat('000.00', corrente
  amostra.Caption :='Amostra: ' + inttostr(amostragem);
  amostragem:=amostragem+1;

  graf.Channels[0].Data.addxypoint(amostragem,corrente);

  if(parar_amostragem=true) then
  begin
        timer1.Enabled:=false;     //Liga o timer para amostragem
        CloseFile(arquivo);        //Fecha o arquivo de amostragem
        parar_amostragem:=false;
  end;

end;
```

Tabela 33: Fonte do programa Delphi para recepção serial

Fonte: Elaborado pelo autor

O botão Parar Amostragem tem a incumbência de desligar o controle Timer de forma que o mesmo pare de solicitar o feedback de corrente da placa de controle. O trecho de código a seguir está implementado neste botão.

```
 parar_amostragem:=true;
```

Tabela 34: Fonte do programa Delphi para encerrar a amostragem

Fonte: Elaborado pelo autor

CONCLUSÃO

O interferômetro é um equipamento que possui grande utilidade em laboratórios que fazem uso de microscopia e até o momento não dispomos de tecnologia nacional que atenda esta demanda. Através deste projeto, espera-se que tal carência seja sanada, já que tal recurso estará disponível a nível nacional, suprindo a necessidade de laboratórios que dependam hoje de importação de peças e componentes para se equiparem. Espera-se também que outras pesquisas se iniciem com este projeto como, por exemplo, na área de ótica já que o driver e detector que serão desenvolvidos serão genéricos, podendo ser empregados de outro modo de acordo com a aplicação a ser empregada.

Neste primeiro relatório, uma das partes que compõem o interferômetro foi implementada, neste caso o driver de laser, porém para se chegar no objeto em estudo falta outras partes como a recepção do laser, mesa XY e o software de tratamento de imagens.

REFERÊNCIAS

[01] Antônio Francisco Gomes Furtado Filho, **Estudo do desempenho do Multiplexador/Demultiplexador add/drop baseado na configuração do Interferômetro de Michelson de fibras ópticas para aplicações em sistemas OTDMA e OCDMA** - Disponível em: < http://www.repositorio.ufc.br:8080/ri/bitstream/123456789/3820/1/2012_tese_afgfurtadofilho.pdf>. Acesso: 23 jan 2013.

[02] FERRIER, NICOLAS. **Le laser et son application en interférométrie pour des measures dimensionnelles**. Bulletin de la Société des Enseignants Neuchãtelois de Sciences, nº 29, octobre 2005, Physique. Disponível em:< http://www.sens-neuchatel.ch/bulletin/no29/art1.pdf>. Acesso em: 14 maio. 2011.

[03] McAleese, Frank G. **The laser experimenter's handbook**. s/l: Tab Boos Inc, 1980.

[04] MELISSINOS, Adrian C.; NAPOLITANO, Jim. **Experiments in modern physics.** s/l: Academic Press, 2003.

[05] C. C. Bradley, J. Chen, and Randall G. Hulet,**Instrumentation for the stable operation of laser diodes** - AIP, Rev. Sci. Instrum. 61, 2097 (1990).

[06] Young, Hugh D., Freedman, Roger A., **Física IV: ótica e física moderna**. 12 ed. São Paulo: Addison Wesley, 2009. 100 p. vol. 4.

[07] A.G. Olszak, J.Schmit, M. G. Heaton, **Interferometry: Technology and Applications –** Bruker, (2000).

[08] Carl E. Wieman, Leo Hollberg, **Using diode lasers for atomic physics** - AIP, Rev. Sci. Instrum. **62**, 1 (1991).

[09] Eric P. Rudd*,* **Laser Diode Driver with 5-Decade Range**- IEEE, Transactions on Instrumentations and Measurement, Vol. 49, Nº. 1, (2000).

[10] Josef Lazar, Petr Jedlika, Ondej íp, and Bohdan Ržika, **Laser diode current controller with a high level of protection against electromagnetic interference** - AIP, Rev. Sci. Instrum. 74, 3816 (2003).

[11] Yukihiro Ishii, Jun Chen, Kazumi Murata, **Digital phase-measuring interferometry with a tunable laser diode** - Optics Letters, Vol. 12, No. 4 (1987).

[12] S. M. Jordan, J. S. S. Whiting, **Detecting two components of magnetization in magnetic layer structures by use of a photoelastic modulator**- AIP, Rev. Sci. Instrum. 67, 4286 (1996).

[13] Byeong Ha Lee, Young Ho Kim, Kwan Seob Park, Joo Beom Eom, Myoung Jin Kim, Byung Sup Rho, Hae Young Choi, **Interferometric Fiber Optic Sensors** - *Sensors*, *12*, 2467-2486 (2012).

[14] Jun MIYAZAKI, Shuichi KINOSHITA, **A Stabilized Michelson Interferometer for Active Phase Locking of Nanosecond Pulse Pair**- Journal of the Physical Society of Japan, Vol. 79, No. 8, August, (2010).

[15] V. Oliveira, N.I.Polushkin, O.Conde, R.Vilar, **Laser surface patterning using a Michelson interferometer and femtosecond laser radiation** - Optics & Laser Technology, No. 44, pp. 2072-2075 (2012).

[16] Nirmala Sanikommu, Nitin P. Wasekar, A. S. Joshi G. Sudararajan, **A virtual instrument for pulsed electrodeposition: A novel technique for obtaining graded coatings**- Journal of the Scientific & Industrial Research, Vol. 70, pp. 1026-1028, December, (2011).

[17]**CPS 180 Datasheet**. Disponível em: <http://www.thorlabs.com/thorProduct.cfm?partNumber=CPS180>. Acesso: 22 jan 2013.

[18] Marc T. Thompson; Martin F. Schlecht, **High Power Laser Diode Driver Based on Power Converter Technology** - IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 12, NO. 1, JANUARY 1997

[19] MALZAHN, Uwe. **Controle de diodos laser**. Tradução: William Salomão, IC-BR Microelectronics. Disponível em:< http://ic-br.com/Diodos_Laser_icbr.pdf>. Acesso em:17 maio 2011.

[20] Altium Designer. **Getting Started with PCB Design**. Disponível em:< http http://www.altium.com/files/Altiumdesigner6/LearningGuides/TU0117%20Getting%20Started%20with%20PCB%20Design.PDF>. Acesso em:25 jan 2013.

[21] Texas Instruments. **Using PWM Output as a Digital-to-analog Converter on a TMS320C240 DSP**. Disponível em:< http://www.ti.com/lit/an/spra490/spra490.pdf>. Acesso em:25 jan 2013.

[22]**SPI Overview and Use of the PICmicro Serial Peripheral Interface**. Disponível em: < http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf>. Acesso: 25 jan 2013.

[23] **Synchronous Serial Peripheral Interface (SPI)**. Disponível em: < http://www.eng.auburn.edu/~nelson/courses/elec2220/Chapter15%20SPI.pdf>. Acesso: 25 jan 2013.

[24] University of California. **Capacitors and Inductors Lab Guide**. Disponível em: < http://www-inst.eecs.berkeley.edu/~ee100/su08/lab/ee100su08Guide3.pdf>. Acesso: 25 jan 2013.

[25]**dsPIC30F2020 Datasheet**. Disponível em: < http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026339>. Acesso: 29 out 2012.

[26]**PIC18F14K50 Datasheet**. Disponível em: < http://www.microchip.com /wwwproducts/Devices.aspx?dDocName=en533924>. Acesso: 29 out 2012.

[27]**USB Framework for PIC18, PIC24 & PIC32**. Disponível em: < http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en537044>. Acesso: 25 jan 2013.

[28]**What is Serial communication?**. A.Awadhi. Disponível em: < http://www.efarabi.com/en/resarchnarticles/rs232%20serial%20communication.pdf4>. Acesso: 25 jan 2013.

[29]**The RS-232 Specification**. Smart-AVI. Disponível em: < http://www.smartavi.com/assets/files/RS-232_Specification.pdf>. Acesso: 25 jan 2013.

[30]**MAX232 Datasheet**. Disponível em: < http://www.ti.com/lit/ds/symlink/max232.pdf>. Acesso: 25 jan 2013.

[31]Silicon Labs. **Serial Communications**. Disponível em: < http://www.silabs.com/Support%20Documents/Software/Serial_Communications.pdf>. Acesso: 25 jan 2013.

[32]SELCO. **SIGMA**/**HyperTerminal**. Disponível em: < http://www.dsf-technologies.com/upload/proddocspdf/proddocspdf_2_109.pdf?PHPSESSID=486e8afe8fca69bc2a32d4070505705d>. Acesso: 25 jan 2013.

[33]Viva o Linux. **GkTerm, uma saída para o Hyperterminal**. Disponível em: < http://www.vivaolinux.com.br/dica/Gtkterm-um-saida-para-o-Hyperterminal>. Acesso: 25 jan 2013.

[34]Ubuntu. **Official Ubuntu Documentation**. Disponível em: < https://help.ubuntu.com/>. Acesso: 25 jan 2013.

[35]**USB 2.0 Specification**. Disponível em: < http://www.usb.org/developers/docs/>. Acesso: 22 jan 2013.

[36]**IRF540 Datasheet**. Disponível em: < http://www.datasheetcatalog.org/datasheet/stmicroelectronics/9387.pdf>. Acesso: 23 jan 2013.

[37] JÚNIOR, Antônio Pereira. **Amplificadores Operacionais e Filtros Ativos**. 6º Edição. São Paulo: Érica, 2010.

[38]**LM358 Datasheet**. Disponível em: < http://www.ti.com/lit/ds/symlink/lm158-n.pdf >. Acesso: 25 jan 2013.

[39]**IRF730 Datasheet**. Disponível em: <https://ec.irf.com/v6/en/US /adirect/ir?cmd=catProductDetailFrame&productID=IRF730>. Acesso: 29 out 2012.

[40]**MCP4822 Datasheet**. Disponível em: < http://www.microchip.com /wwwproducts/Devices.aspx?dDocName=en024016>. Acesso: 29 out 2012.

[41]**TL084 Datasheet**. Disponível em: < http://www.ti.com/lit/ds/symlink/tl084.pdf>. Acesso: 29 out 2012.

[42]**INA121 Datasheet**. Disponível em: < http://www.ti.com/lit/ds/symlink/ina121.pdf>. Acesso: 29 out 2012.

[43]**I²C Specification**. Disponível em: < http://www.nxp.com/documents/other/39340011.pdf>. Acesso: 23 jan 2013.

[44]**IRLZ14N Datasheet**. Disponível em: < http://www.irf.com/product-info/datasheets/data/irlz44n.pdf>. Acesso: 5 nov 2012.

[45] Katsuhiko Ogata. **Engenharia de Controle Moderno**, 5º Edição. Pearson (2010).

[46] Richard C. Dorf. **Sistemas de Controle Moderno**, 11º Edição. Prentice Hall (2008).

[47] Farid Golnarachi; Benjamin C. Kuo**. Sistemas de Controle Automático**, 1º Edição. LTC (2012).

[48]Guimarães, Ângelo Moura. **Algoritmos e Estrutura de Dados**. 50º Edição. São Paulo: LTC, 1994.

[49]PEREIRA, Fábio**. PIC18 Detalhado**. 1º Edição. São Paulo: Érica, 2009.

[50]PEREIRA, Fábio. **PIC: Técnicas Avançadas**. 1º Edição. São Paulo: Érica, 2002.

[51]PEREIRA, Fábio. **PIC em** C. 1º Edição. São Paulo: Érica, 2010.

[52]ZANCO, Wagner da Silva. **Microcontroladores PIC18 com linguagem C**. 1º Edição. São Paulo: Érica, 2010.

[53]ZANCO, Wagner da Silva. **Microcontroladores PIC Técnica Avançadas**.1º Edição. São

[54]Fábio Pereira. **Microcontroladores PIC: Técnicas Avançadas, 1º Edição.** Érica (2002).

[55]Fábio Pereira. **Microcontroladores PIC18 Detalhado – Hardware e Software, 1º Edição.** Érica (2010).

[56] Mazidi, Muhamaad Ali, Mckinlay, Rolin D, Causey, Danny ,**PIC MICROCONTROLLER and EMBEDDED SYSTEM** using assembly and C for PIC 18.

[57] Di Jasio, Lucio;Wilmshurst,Tim;Ibrahim, Dogan; Morton,John; Bates ,Martin P.;Smith, Jack; Smith, D.W.;Hallebucyck,Chuck, **PIC Microcontrollers**, Newnes Know it al series, Elsevier Inc. , 2008

[58]DIAS, Adilson de Souza. **Delphi: Para todas as versões**. 1º Edição. Rio de Janeiro: Ciência Morderna, 1º Edição, 1999.

[59]KIMMEL, Paul. **Desenvolvendo Aplicações em Delphi 6**. 1º Edição. Rio de Janeiro: Ciência Moderna, 2001.

[60]LOBO, Rodrigo. **Delphi 5: Dicas e Truques**. 1º Edição. São Paulo: Relativa, 2000.

[61]MECENAS, Ivan. **Delphi 6: Desenvolvendo Projetos**. 1º Edição. São Paulo: Alta Books, 2001.

[62]Marcelo Leão. **Delphi 7 – Curso Completo, 1º Edição.** Axcel (2003).

[63] LM340/LM78XX Series, **3-Terminal Positive Regulators**, 2001 National Semiconductor Corporation

[64] TL082, **Wide Bandwidth Dual JFET Input Operational Amplifier**, National Semiconductor Corporation,2004

[65] MCP1525/41**, 2,5V and 4.096V Voltage References**, Microchip Technology, 2005

[66] AN737, **Using Digital Potentiometers to Design Low-Pass Adjustable Filters,** Microchip Technology Inc., 2004