3
O ambiente de desenvolvimento

3.1
Introdução

A idéia destes algoritmos surgiram de tratar de assemelhar ao robô com a pessoa humana; o robô conta com uma câmera que assemelha ao olho humano, os descritores SIFT assemelha à percepção do olho, as redes neurais ao aprendizagem do cérebro, a lógica fuzzy a toma de decisão da pessoa e algoritmos genéticos a idéia de otimizar a menor trajetória entre dois pontos, como se ilustra na figura Figura 3.1.

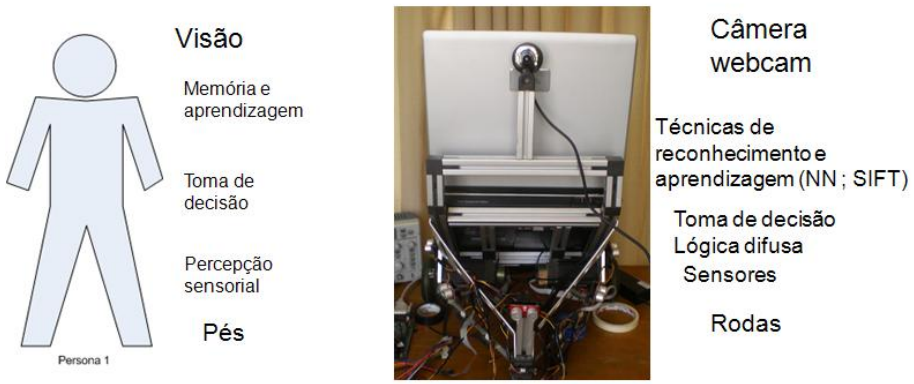

Figura 3.1: Semelhança robô e a pessoa humana

O robô ER1 é usado pela facilidade da estrutura mecânica y a facilidade de modificar nele outro mecanismo de mobilidade para facilitar nosso trabalho.

O uso do DSP e a notebook é para o seguinte: o DSP vai gerar a sinais de controle do motor usando o processamento embebido da lógica fuzzy além disso também faz a leitura dos sensores de proximidade para treina os na rede neural. Para a aquisição das imagens o sistema usa uma camera e para o processamento a notebook, devido à dificuldade do processamento o DSP não pode processar imagens, então a notebook ajuda ao DSP no processamento de imagens para ter informação dele e poder comunicar ao DSP para uma correta navegação.

Os sistemas de controle desenvolvidos neste trabalho visam ao tratamento dos dados provindos dos sensores, efetuando-se a navegação no ambiente desconhecido, O sistema de controle dever evitar que o veículo colida com os obstáculos, ao mesmo tempo que procura nós para poder mapear o ambiente desconhecido. Logo disso o robô tem que ser capaz de reconhecer cada nó do ambiente já explorado e poder navegar para qualquer ponto dele.

Para que o robô tenha sucesso com a tarefa de exploração é preciso que o robô tenha sucesso com a etapa da percepção sensorial e com a etapa de controle dos atuadores(motores) para isso se fiz um projeto mecânico, um projeto eletrônico, e o desenho do software.

3.2 Projeto mecânico do robô

3.2.1 Robô ER1

O robô móvel utilizado neste projeto foi construído pela empresa *Evolution Robotics*, e é por isso que se denomina $ER1$. O ER1 se locomove por direção diferencial tipo torque através de duas rodas ativas acionadas por motores de passo. Este robô foi extensivamente modificado neste trabalho para poder ter maior capacidade de mobilidade, trocando os motores de passo pelos motores *brushless* que são melhores no tempo de resposta e pode fazer movimentos muito mais precisos; além disso também o sistema de controle foi trocado por um processador digital de sinais (DSP) que com a ajuda de um *notebook* LG vão fazer as tarefas de controle usando algoritmos de navegação;o robô foi implementado com 11 sensores infravermelhos , 6 sensores ultra-sônicos e uma câmera como se mostra na Figura 3.2

O robô $ER1$ é composto de:

- Chassi: Vigas de perfil x de alumínio, conectores de plástico, porcas e placas laterais em alumínio.
- Mecanismo pré-montado sobre placas de alumínio que serve de suporte dos motores, tem dois rodas de 4 polegadas de diâmetro, correias e polias e também contém um rodízio de 360º.
- Bateria recarregável de $12V$ e $5,4A-h$ com fusível interno para proteção.

Figura 3.2: Robô ER1 modificado no momento de testes

3.2.2 Motores brushless

O robô foi implementado com dois motores *brushless DC (BLDC)* tipo $5143DO12$ da séries 5100 da família *Pittman-Elcom* como se ilustra na Figura 3.3; os motores BLDC são um dos tipos de motores que esta ganhando popularidade nas aplicações industriais pelas vantagens sobre os motores DC, como o nome deles diz, os motores BLDC não usam escovinha para sua comutação, no seu lugar usa comutação eletrônica; este motor tem as seguintes vantagens respeito aos motores DC com escovinha e os motores de indução:

- Melhor característica de velocidade versus torque.
- Alta resposta dinâmica.
- Alta eficiência.
- Largo tempo de vida.
- Operação sim ruídos.
- Altos ranges de velocidade.

A operação do motor *BLDC* é uma seqüencia de comutação numa das bobinas com voltagem positivo a segunda bobina é negativa e a terceira bobina não tem voltagem. O torque é produzido porque há interação entre o campo magnético gerado pelo estator e o ímã permanente. Idealmente o

Figura 3.3: Motor Brushless DC

torque mais alto ocorre quando estes dois campos estão a 90º um do outro e cai quando se movem juntos. Em disposição de manter o motor funcionando, o campo magnético produzido pelas bobinas deveriam trocar de posição. Como o movimento do rotor é capturado com o campo do estator, isto leva a definir uma seqüencia de comutação de 6 passos para ter voltagem nas bobinas, isto também leva a determinar a posição exata das bobinas no momento de giro, para isso o motor *brushless* esta implementado com sensores de efeito hall como se ilustra na Figura 3.4, para determinar as fases corretas e a forma de energiza-o seqüencialmente.

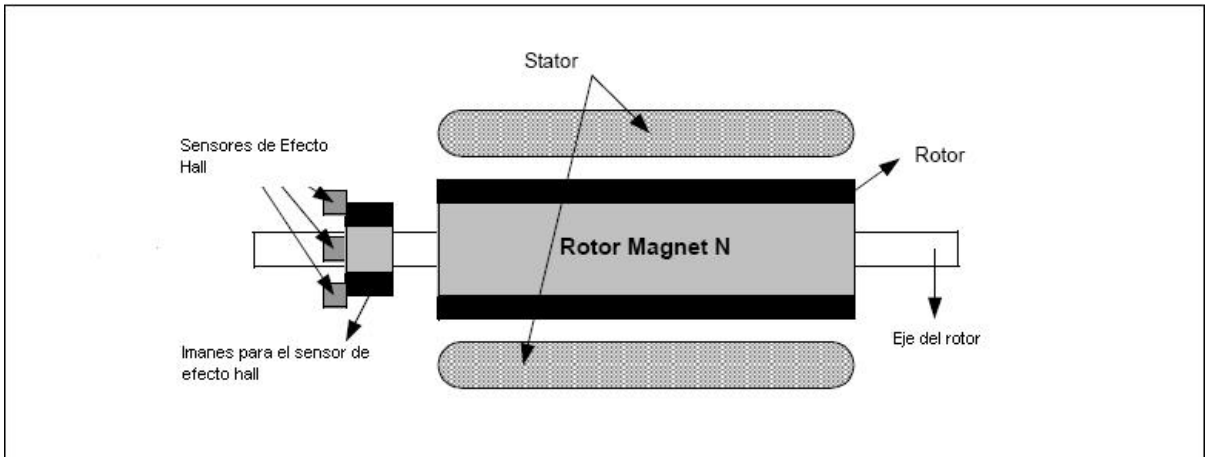

Figura 3.4: Vista transversal do motor brushless

3.2.3 Sensores

O robô $ER1$ foi implementado por 6 sensores ultra-som e 11 sensores infravermelhos os quais foram colocados simetricamente na estrutura do robô.

Sensor ultra-som

O sensor ultra-som que foi implementado no robô é o $SRF08$ (Figura 3.5), que tem um alcance de $3cm$ até $6m$ e a interface de comunicação é mediante o protocolo de comunicação IIC (*Inter-Integrated Circuit*); o $SRF08$ registra 17 subsequentes pulsos de ecos que correspondem ao objeto mais perto isto permite fazer uma medida mais acertada da distância ao objeto.

Figura 3.5: Sensor de Ultra-Som SRF08

A leitura destes sensores tem dois tipos de modalidades, a primeira é que o sensor pode armazenar o resultado da medida em cm, s ou $polegadas$ e a segunda modalidade é que pode armazenar o resultado da medida num registro de 32 bytes para o fácil uso se fosse o caso de precisar utilizar uma rede neuronal e ter como entrada a leitura do sensor; onde cada byte representa o espaço de $352mm$, é dizer se um objeto se encontra a $3m$ então o registro 8 seria ativado a 1 lógico e assim as entradas à rede neuronal estaria pronto para o treinamento.

Sensor infravermelho

O sensor que foi implementado é o $GP2D15$ da família $sharp$(Figura 3.6), este sensor tem um alcance de $10cm$ a $80cm$ e produz um voltagem proporcional à distância do objeto mas neste projeto ele é usado como um detector digital a uma distância de $25cm$, é dizer se existe um objeto a uma distância menor a $25cm$ o detector dará informação de 1 lógico.

Figura 3.6: Sensor Infravermelho GP2D15 da família Sharp

Encoder

O encoder utilizado para a estimar a velocidade dos motores é o Encoder $HEDL5540$ (Figura 3.7) de $500ppv$ com três canais e com um driver $RS422$ como se ilustra na Figura 3.8.

Figura 3.7: O Encoder HEDL5540

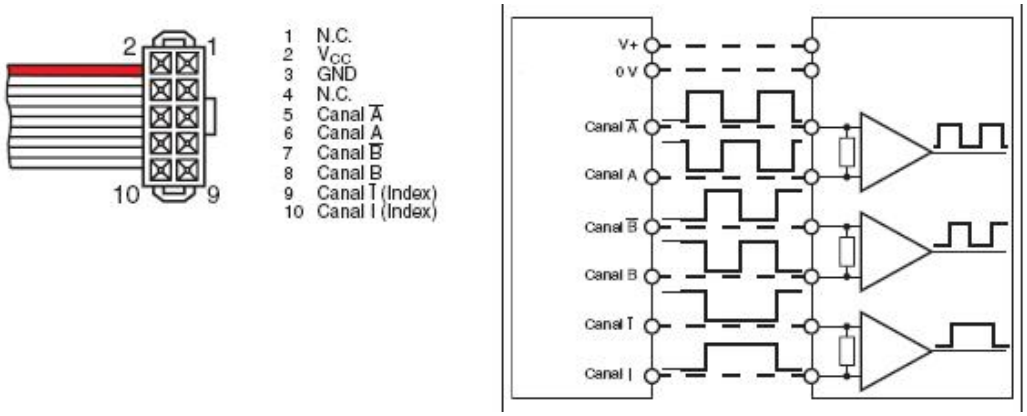

Figura 3.8: conexão e pulsos do encoder

Câmera

A câmera *webcam* utilizada é uma tipo *USB* da família *Clone* tão como se ilustra na Figura 3.9, ele tem uma resolução de 80×60 até 640×800 e é de 4 Megapixels.

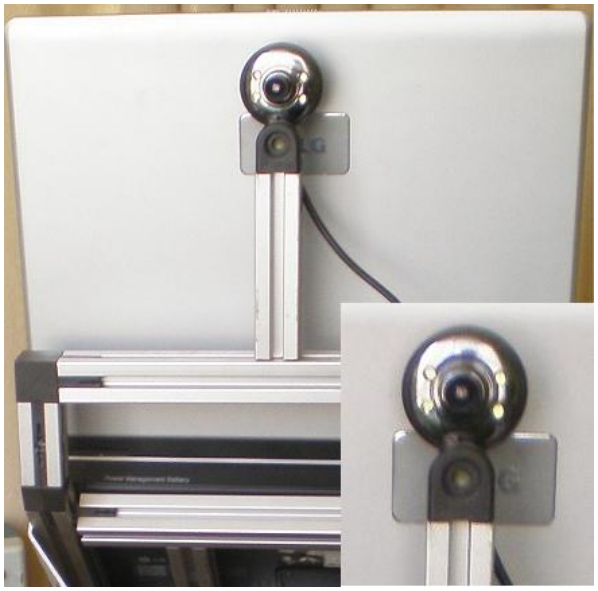

Figura 3.9: Câmera digital USB Clone

3.3 Projeto eletrônico do robô

No desenho eletrônico tivesse presente que o DSP é um processador de baixa potência, é dizer que ele trabalha a um voltagem de $3,3 voltios$, é por isso que a interface eletrônica com os outros dispositivos que trabalham a $5 voltios$ foi preciso.

Microprocessador TMS320F2812

O DSP (*Digital Signal Procesor*) é o dispositivo de controle e junto com o *notebook* é o cérebro do robô, é um processador de *Texas Instruments*, uma das maiores empresas fabricantes de ultimas tecnologias.

Um *DSP* é um processador de propósito particular (dedicado), com características especiais a nível estrutural, que permitem maximizar seu rendimento em términos de capacidades de memória e velocidade do processo; na Figura 3.10 se ilustra a arquitectura do *DSP*; atualmente este dispositivo é utilizado na maioria das tecnologias existentes no mundo, como são em sensores inteligentes, automóveis, aeronáutica, equipamentos médicos, etc.

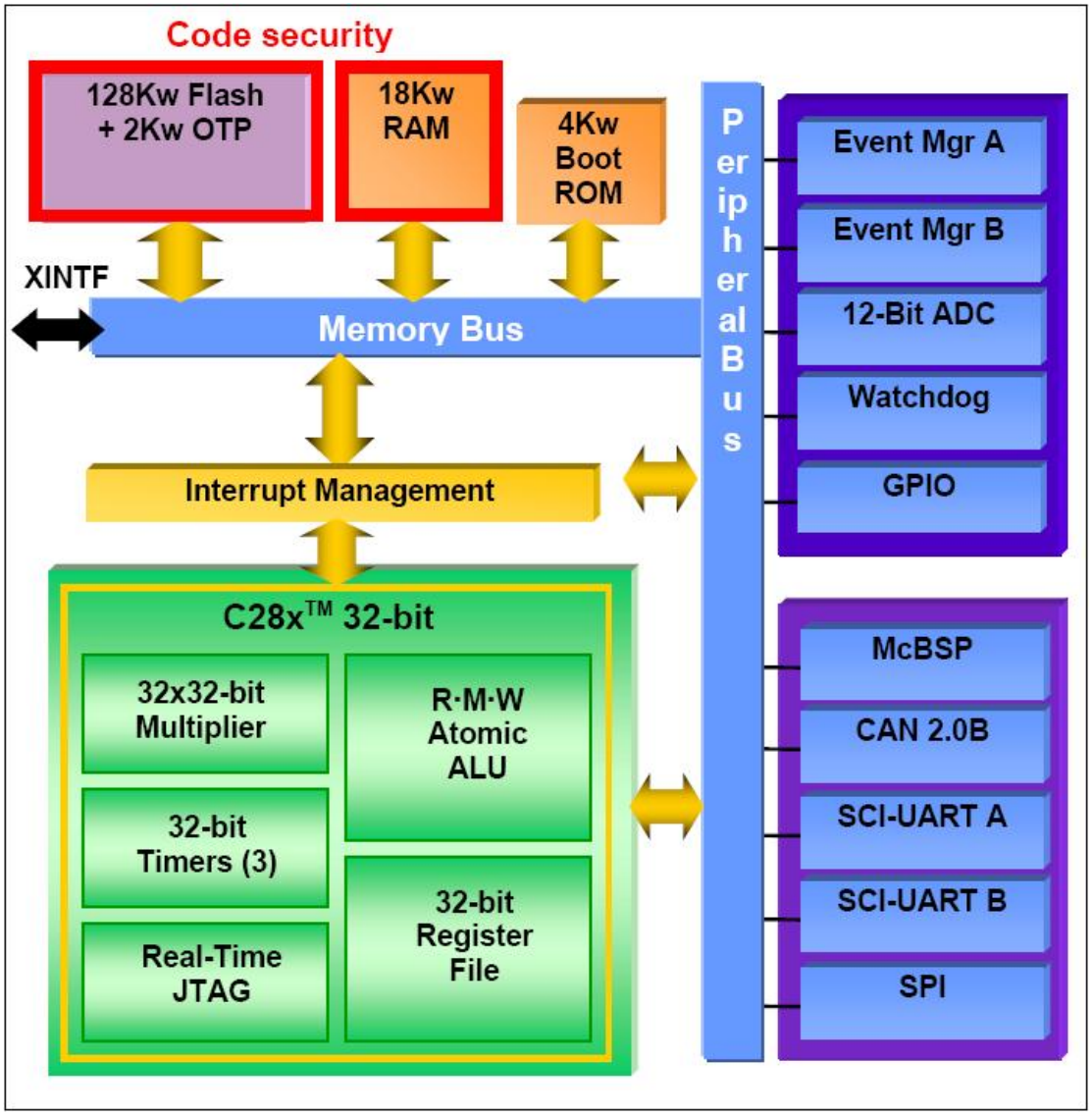

Figura 3.10: Arquitectura do TMS320F2812

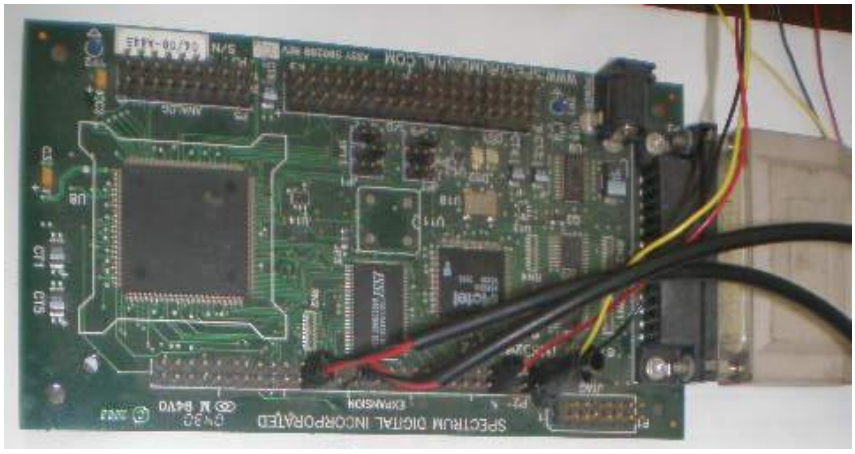

Figura 3.11: Processador de Sinais DSP TMS320F2812

El *DSP* $TMS320F2812$(Figura 3.11), do que pode se ver uma completa descrição em (sprs174), é um dispositivo mais potente da família $C2000$ de *Texas Instruments* dos todos os *DSPs* orientados ao controle de processos. Incorpora um núcleo de arquitetura Harvard de 32 bits de coma fixa a 150 MHz, capaz de executar funções MAC (multiplicar e acumular) num só ciclo de trabalho. Além disso, incorpora todos os periféricos necessários para o controle de motores: geradores de sinais PWM(*Pulse Width Modulation*) com tempo morto, conversores A/D (*Analogic to Digital Converter*), capturadores para o encoder, além disso tem algumas funções adicionais que ajudam no controle de processos, como se pode ver na Figura 3.10.

O *DSP* também incorpora memória *Flash* para o código e memória *RAM* para as variáveis. Também incorpora uma pequena memória ROM (*Read Only Memory*), onde se albergam os modos de arranque e umas tabelas para o cálculo matemático de algumas funções mediante interpolação numérica.

Como periféricos de comunicação incorpora controladores para distintos standards de comunicação, como pode ser o *UART Universal Asynchronous Receive and Transmit*), o protocolo *CAN* (*Controller Area Network*) e *McBSP* (*Multichannel Buffered Serial Port*). Neste dispositivo concreto (TMS320F2812), o bus de endereço e de dados, é acessível, com o que se pode ampliar seus recursos.

Os periféricos que interessam mais para este trabalho são três: os geradores de sinais *PWM*, os conversores A/D e os capturadores para o encoder, todos eles estão englobados no que se chama *Event Manager*(EV). O $TMS320F2812$ dispõe de dois EV, o A e o B. Cada um deles funciona como uma unidade independente capaz de gerar seus interrupções e administrar todos seus recursos. Tanto o A como o B funcionam da mesma forma, com o que com um só *DSP* é possível controlar dois motores *brushless* ao mesmo tempo. O EV dispõe de dois *Timers* de 16 bits com o que se gera as bases do tempo tanto para os registros de comparação como para o capturador da sinal do encoder. Três registros de entrada (CMPR1, CMPR2, CMPR3) geram as seis sinais complementarias dois a dois e se agrega um tempo morto totalmente programável desde $0us$ até $12us$.

Além, o *DSP* dispõe de uma sinal de entrada de falha do conversor que automaticamente e sim uso da *CPU* (*Central Process Unit*), pôr em alta impedância as seis saídas *PWM* e gera uma interrupção de software, que permite administrar a falha de forma rápida e segura para o entorno.

O capturador para o encoder funciona de forma muito fácil; ele se limita a medir o tempo entre dois pulsos consecutivos provenientes do encoder. Dispõe de uma lógica interna para decodificar o sentido de giro.

O conversor A/D é de 12 bits e dispõe de dois S/H (*Sample and Hold*; a entrada analógica tem um margem dinâmico de 0.0 V a 3.0 V, e é capaz de operar a $12,5$ *MSPS* (*Mega Samples Per Second*), velocidade suficiente para aplicações de controle de processos industriais. Tem vários modos de funcionamento e de sincronização com outros periféricos, em especial os *EV*, com o qual pode se fazer o amostragem das sinais no instante adequado sem necessidade de usar a *CPU*.

Circuito de potência dos motores

O projeto se fez tendo em conta as características de resposta em freqüência do robô e do controle, também se tomou em conta que era preciso ter uma interface entre os terminais do *PWM* que trabalha a $3,3V$ com os drivers dos *mosfet*, os *mosfet* são a parte importante do circuito de potência, eles são os que trabalham diretamente com o motor, e o tempo de resposta deles tem que ser muito maior que a velocidade de trabalho dos PWMs para ter um ótimo controle, na Figura 3.12 se ilustra o projeto feito no software *Orcad Capture*.

Interface dos sensores

Com o fim de reduzir o consumo da potência dos dispositivos na atualidade os dispositivos tem um menor consumo de energia, é por isso que os atuais dispositivos *DSP* são alimentados por uma fonte de voltagem de $3,3$ vóltios, isto é compatível com componentes e circuitos integrados (ICs) no mercado. Neste caso do *DSP* as entradas e saídas lógicas do controlador do motor trabalha a $3,3V$ então temos que ter em conta para o projeto da interface dos sensores com o *DSP* os seguintes casos (spra550):

A interface do sensor infravermelho é como se ilustra na Figura 3.13, o sensor quando percebe um objeto desativa o *gate* do *mosfet* então o *mosfet* deixa de conduzir corrente e a entrada do DSP é $3,3V$.

A interface com o sensor ultra-som é como se ilustra na Figura 3.14, o projeto é desse jeito porque o $TMS320F2812$ não tem o módulo de comunicação $I2C$ mas tem outro módulo com o qual pode se fazer o protocolo de comunicação $I2C$.

A interface com o encoder é como se ilustra na Figura 3.15, neste caso se fez uma interface para que o pulso de $5V$ seja de $3,3V$ à entrada do *DSP*.

3.4

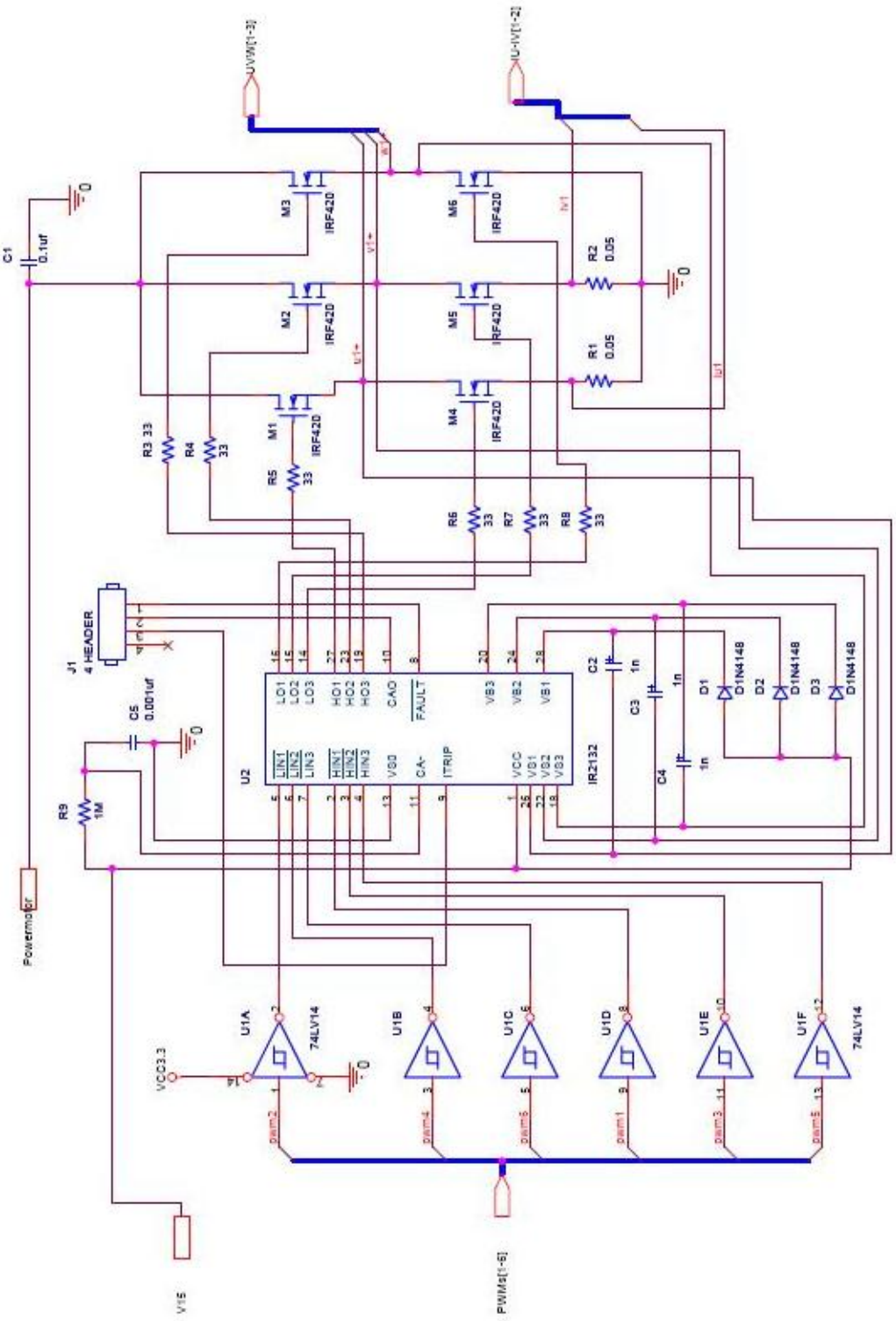

Figura 3.12: Circuito de potência do motor Brushless

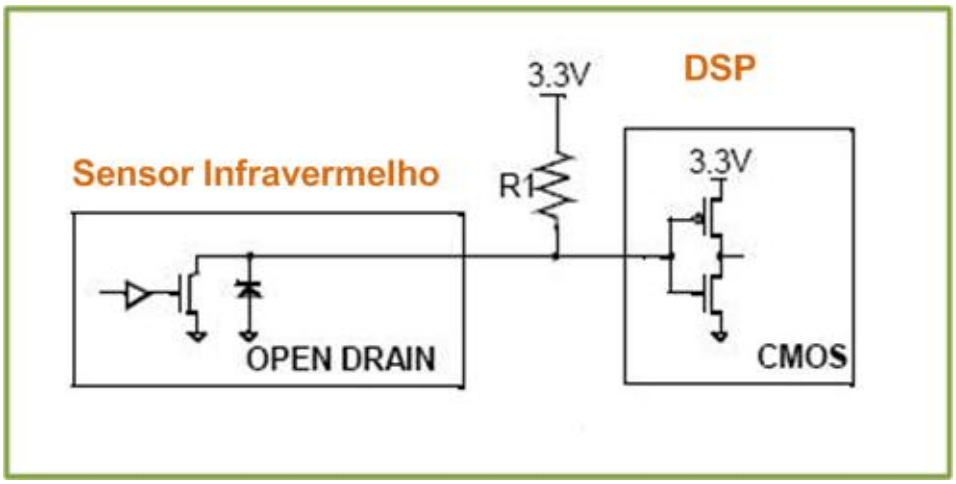

Figura 3.13: Interface do sensor infravermelho com o DSP

Projeto do software

Neste projeto se tem o desenvolvimento de duas etapas, o primeiro a etapa de simulação e o segundo a etapa de controle do robô que é o trabalho de dois software que estão em comunicação constante, a descrição de cada um deles se da a continuação:

3.4.1 Software de simulação

O software que se utilizou para a simulação foi o *Player-Stage* que é um simulador *open source* da plataforma *Linux*, que possibilitam simulações em $2D$ para robôs móveis com seus respectivos sensores e drivers comerciais já implementados no software tais como laser, sonares, gps, etc.; Possuem módulos para construção e localização de mapas *cross compiling*, etc.

O linguagem que se utiliza neste programa o C++ um linguagem muito fácil de usar, na Figura 3.16 se mostra um exemplo do ambiente de simulação com o *player-stage*.

3.4.2 Software de controle

O software de controle foi implementado com o uso de dois programas, eles se encarregam de todo o processo de exploração do robô, como descrito a seguir.

Code Composer Studio

O software que proporciona o fabricante para interatuar com o *DSP* é o *Code Composer Studio*(CCS), o mesmo programa serve para todas as famílias

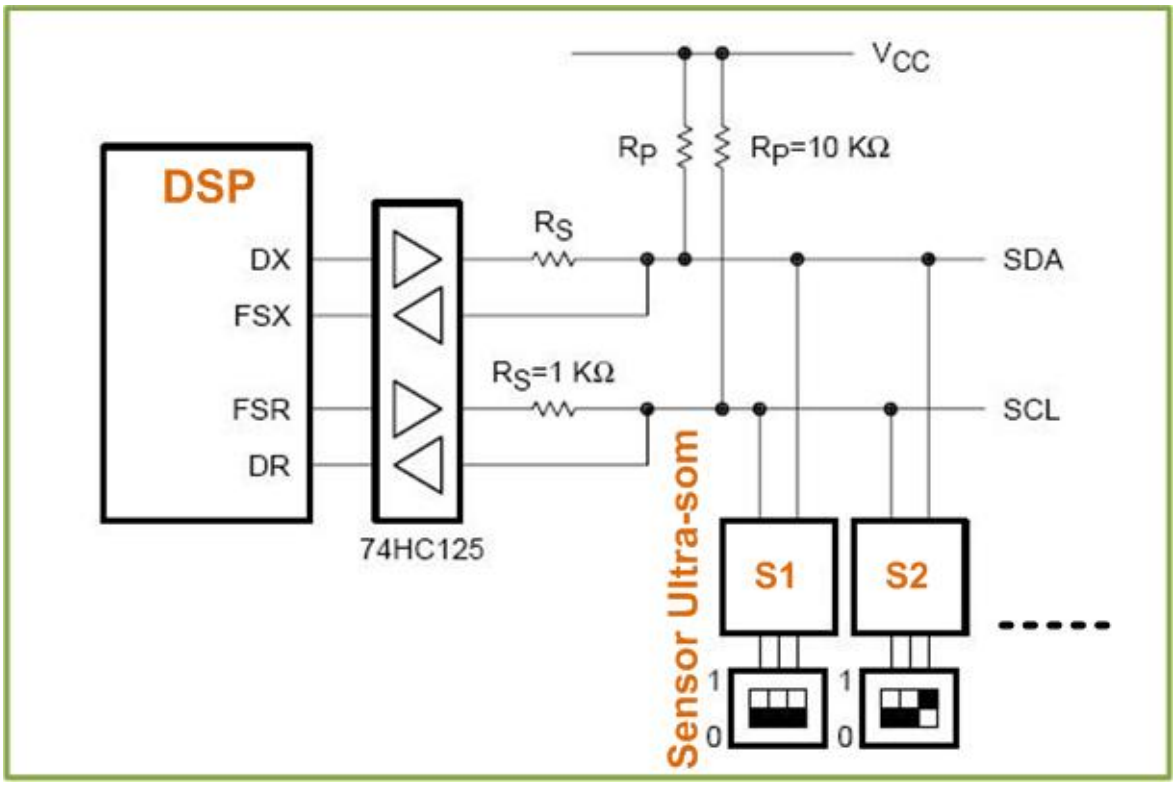

Figura 3.14: Interface do Sensor Ultra-som com o DSP

de *Texas Instruments*. O CCS forma o que se chama um sistema integrado de desarrolho, já que desde ele pode se escrever, compilar e carregar o código ao *DSP* em linguagem C e em Assembler; além disso é uma potente ferramenta para o análises e o depurado do código em tempo real mediante pontos de detecção (*breakpoints* ou interrupções). Permite também visualizar e modificar variáveis internas em tempo real sem deter a *CPU*.

As funções permitidas pelo entorno tem como objetivo facilitar a sintonização dos valores influentes no controle e o ajuste dos valores dos *PID* (controlador proporcional, integral e derivativo) em tempo real da forma mais eficaz. Na Figura 3.17 se ilustra uma imagem da janela principal do programa *CCS*, a qual à vez está dividida em diferentes janelas que amostram a informação necessária para um perfeito seguimento do controle. Como pode se notar apresenta varias janelas, algumas onde se desarrolha o programa e outras onde se amostra o código assembler e também o valor das variáveis e seu respectiva gráfica em tempo de execução.

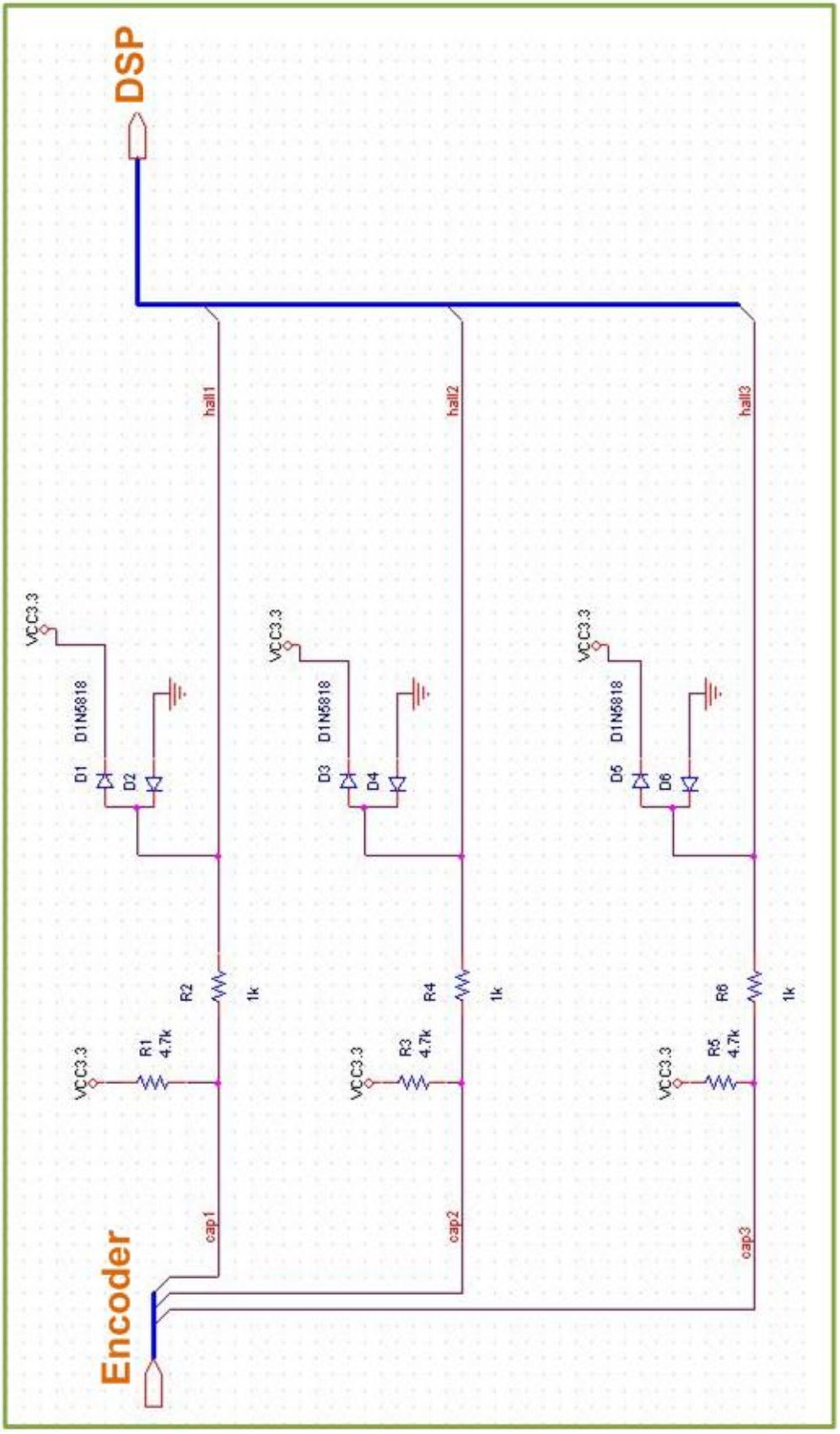

Figura 3.15: Interface do Encoder com o DSP

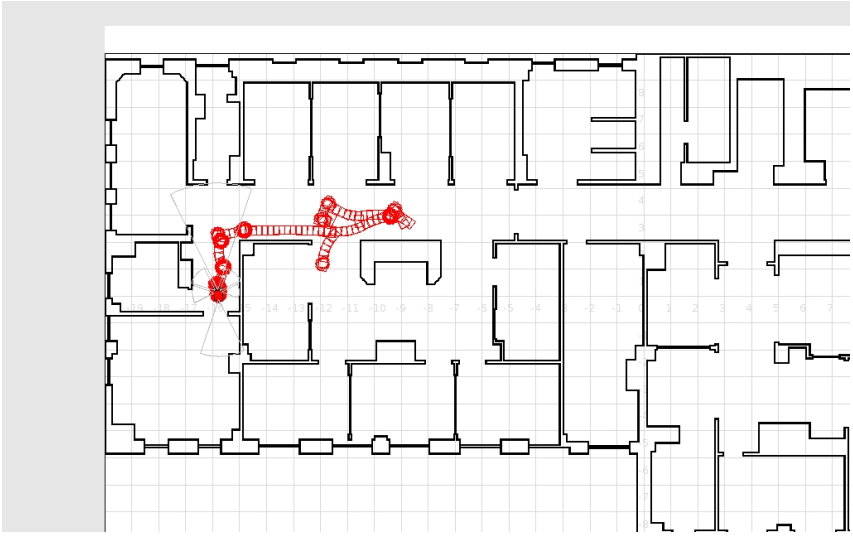

Figura 3.16: Ambiente de simulação do player-stage em 2-D

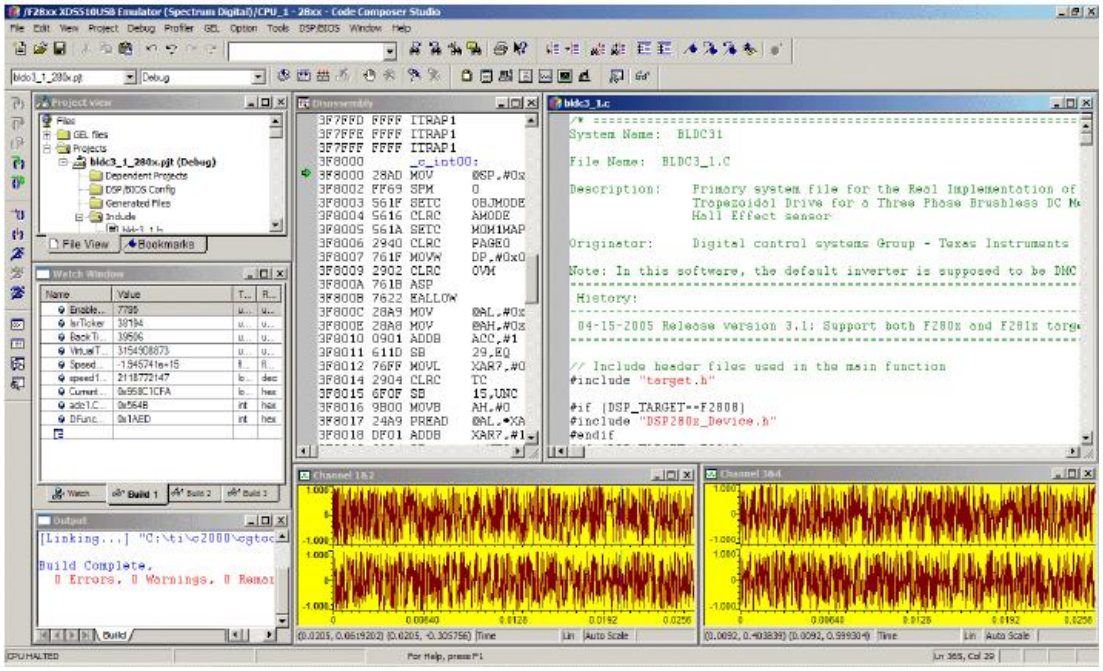

Figura 3.17: Aspecto da janela do Code Composer Studio

Matlab

Matlab é um software muito conhecido no mundo científico, e tem em suas livrarias um módulo que faz a interface entre o *DSP* em tempo real e o Matlab, isto é de grande ajuda porque neste trabalho a parte da comunicação entre o matlab e o *CCS* é importante para o desarrolho ótimo desde trabalho, o tutorial para aprender as funções pode ser encontrado escrevendo no promp do matlab o comando ccstutorial.

Controle do motor Brushless

Nesta seção vai se explicar qual é o tipo de técnica de controle utilizado e desarrolhado para o controle do motor *brushless*; existem várias técnicas de controle para este tipo de motores um dos mais simples é o controle baseado em comutação trapezoidal (Ham04), e que foi implementado para este trabalho mas posteriormente devido as deficiências de exatidão do controle foi trocado pelo controle vetorial o qual é o controle mais complexo e que requer maior potência de cálculo mas é a que melhor controle proporciona.

O controle vetorial ou *Field Oriented Control (FOC)* controla o vetor de correntes diretamente no espaço de referência ortogonal e rotacional, chamado espaço D-Q *(Direct-Quadrature)*. Dito espaço de referência está normalmente alinhado com o rotor de forma que permite que o controle do fluxo e o par do motor se realize de forma independente. A componente direta permite controlar o fluxo e a componente em quadratura o par.

Para poder realizar este controle é necessário transformar matematicamente as medidas das três correntes referidas ao espaço estático das bobinas do motor ao espaço rotacional D-Q. Embora esta transformação pode implementar se em um só passo, mas para o melhor entendimento se faz em dois transformações; a transformada de Clarke e a transformada de Park (Ham04).

O diagrama de controle se ilustra na figura 3.18, onde cada bloco representa programas feitos no linguagem C.

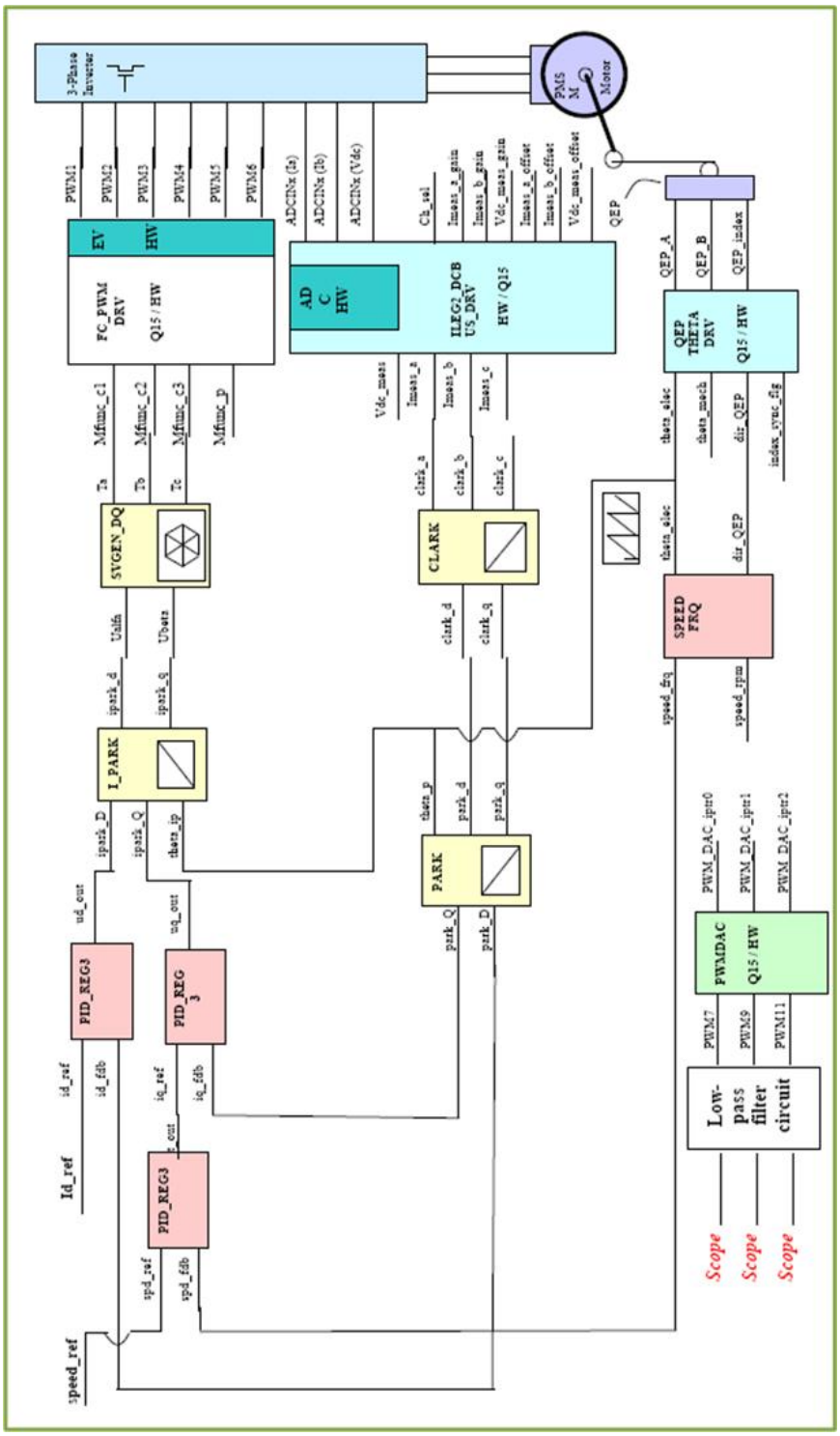

Figura 3.18: Diagrama de blocos do controle vetorial de velocidade para o robô

No próximo capitulo, a metodologia para auto localização e mapeamento do ambiente pelo robô é descrita.