UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
ENGENHARIA INDUSTRIAL ELÉTRICA – ELETRÔNICA/TELECOMUNICAÇÕES

ERIKA KLITZKE
GABRIEL DE CARVALHO BUENO
ISABELA BARLETA JAVORSKY

UNIDADE SEGUIDORA DE SOLDA (U.S.S. AIR-12)

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2013

ERIKA KLITZKE
GABRIEL DE CARVALHO BUENO
ISABELA BARLETA JAVORSKY

UNIDADE SEGUIDORA DE SOLDA DO ROBÔ AIR-12 (U.S.S. AIR-12)

TRABALHO DE CONCLUSÃO DE CURSO

Trabalho de Conclusão de Curso de graduação do Curso de Engenharia Elétrica – Eletrônica/Telecomunicações da Universidade Tecnológica Federal do Paraná como requisito parcial para o título de Engenheiro Eletricista.

Orientadora: Prof.ª Dr.ª Lucia Valéria Ramos de Arruda

CURITIBA
2013

AGRADECIMENTOS

Apoio financeiro da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis – ANP –, da Financiadora de Estudos e Projetos – FINEP –, do Ministério da Ciência e Tecnologia – MCT – por meio do Programa de Recursos Humanos da ANP para o Setor Petróleo e Gás – PRH-ANP/MCT – e do Programa de Formação de Recursos Humanos da PETROBRAS - PRH10-UTFPR.

RESUMO

KLITZKE, Erika; BUENO, Gabriel de C.; JAVORSKY, Isabela B. Unidade Seguidora de Solda do Robô AIR-12 (U.S.S. AIR-12). Trabalho de Conclusão de Curso – Engenharia Industrial Elétrica – ênfase Eletrônica/Telecom., Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

A indústria de Petróleo e Gás Natural (P&G), devido à competição de mercado e buscando baratear custos e aumentar sua produtividade, está sempre em constante desenvolvimento tecnológico em áreas como extração e processamento de petróleo e seus derivados, manutenção e inspeção de equipamentos, entre outras atividades técnicas. Junto a esses esforços estão os programas de incentivo à formação de recursos humanos para a indústria, como é o caso dos Programas de Formação de Recursos Humanos (PRH) da ANP e da Petrobras. A UTFPR é a sede do PRH-10, precisamente o Laboratório de Automação e Sistemas de Controle Avançado (LASCA) desenvolve projetos em parceria com a Petrobras e concede bolsas a estudantes, como os integrantes deste grupo de trabalho, para formar recursos humanos para a indústria de Petróleo e Gás Natural através da realização de projetos de Pesquisa e Desenvolvimento. Dentre os projetos do laboratório LASCA está o desenvolvimento de ferramentas de controle e automação de processos de P&G, que são aplicáveis principalmente às áreas relacionadas à inspeção e segurança de equipamentos, destacando-se entre estes equipamentos os tanques e esferas de armazenamento de combustíveis e derivados, que são instalações de grande porte necessitando constantes manutenções. Este projeto se caracteriza pelo desenvolvimento de um módulo de processamento de imagem em tempo real que, através de um algoritmo desenvolvido, define a trajetória de um robô de inspeção, sobre um tanque ou esfera de armazenamento. Este processamento é realizado por hardware e faz com que o robô se movimente autonomamente ao longo de um cordão de solda. Foi fabricada uma estrutura que permite que feixes de laser incidam sobre um cordão de solda e uma câmera realiza a captura da imagem do padrão luminoso formado pelas faixas de laser sobre o cordão. Esta imagem é processada por um hardware codificado em *VHDL (Very High Speed Integrated Circuit Hardware Description Language)*, que será embarcado em um *FPGA (Field-programmable gate array)*. Este módulo é parte da malha de controle dos motores que realizam a movimentação do robô. A partir deste módulo, outros grupos de bolsistas do PRH-10 poderão desenvolver módulos específicos de inspeção ou localização para que o robô inteiro se torne um produto aplicável à indústria de P&G.

Palavras-chave: Robótica Móvel. Inspeção de Solda. Visão Computacional. Processamento Digital de Imagem por *hardware*.

ABSTRACT

KLITZKE, Erika; BUENO, Gabriel de C.; JAVORSKY, Isabela B. Unidade Seguidora de Solda do Robô AIR-12 (U.S.S. AIR-12). Trabalho de Conclusão de Curso – Engenharia Industrial Elétrica – ênfase Eletrônica/Telecom., Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

The Industry of Oil and Gas (O&G), due to market competition, seeking to minimize its costs and to increase its productivity, is always in constant technological development in areas such as extraction of oil and production of oil products, maintenance and inspection of equipment, among other technical activities. In order to promote training of human resources for this industry there were created special educational programs, such as the PRHs, which are Training Programs for Human Resources that are sponsored by ANP and Petrobras. UTFPR is the seat of PRH-10, precisely the Laboratory of Automation and Advanced Control Systems (LASCA) develops projects in partnership with Petrobras and provides scholarships for students, as the members of this working group, to form human resources for industry Oil and Gas by conducting research and development projects. Among these projects tools are being developed to control and automate processes for O&G, which apply primarily to fields related to surveillance and security of equipment, foremost among these equipment there are tanks and spheres for fuel storage. These large facilities require constant maintenance. This project is characterized by the development of a module for image processing in real time, using an algorithm that defines the trajectory of a robot inspection on a tank or storage sphere. This processing is realized by hardware, and causes the robot to autonomously move along a weld bead. It was assembled a structure that allows laser to beam a weld bead and a camera performs the image capture of the light pattern formed by laser stripes on the weld bead. This image is processed by a circuit coded in VHDL (Very High Speed Integrated Circuit Hardware Description Language), which is embedded in an FPGA (Field-programmable gate array). This module is part of the control loop of the engines that perform the movement of the robot. From this module, other groups of PRH-10 scholars can develop specific inspection or location modules, so the entire robot shall become a product applicable to O&G industry.

Key-words: Mobile Robotics. Weld Inspection. Computational Vision. Digital Image Processing by hardware.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Para o sucesso de qualquer indústria é fundamental a busca por redução de custos e aumento da produtividade, por este motivo a indústria de Petróleo e Gás Natural (P&G) está sempre em constante desenvolvimento tecnológico em áreas como extração e processamento de petróleo e seus derivados, manutenção e inspeção de equipamentos e instalações, entre outras atividades técnicas. A inspeção externa de tanques de armazenamento de petróleo e seus derivados é uma prática necessária para garantir a segurança e a confiabilidade desses ambientes, influenciando as demais etapas da cadeia de abastecimento. Porém, este é um processo complexo, que demanda tempo devido à extensão das instalações e na maioria das vezes é um trabalho altamente insalubre e com riscos ambientais.

A automatização dos processos de inspeção por meio de robôs autônomos reduz a necessidade de operadores, garantindo uma melhor segurança às instalações, otimizando o tempo de inspeção, e podendo até mesmo reduzir os custos deste processo. Um robô voltado à inspeção desses ambientes necessita ser totalmente autônomo, de operação flexível, capaz de movimentar-se por todo o tanque e possibilitar a supervisão remota de todo o processo de inspeção. Visando a solução deste problema de engenharia, está sendo desenvolvido no laboratório LASCA da UTFPR, em parceira com a Petrobras e com apoio do Programa de Formação de Recursos Humanos (PRH-10) da ANP, o projeto *Autonomous Inspection Robot* (AIR-12).

O projeto AIR-12 aborda um sistema de inspeção composto por um robô de inspeção, cuja parte mecânica foi desenvolvida em um trabalho anterior, controlado através de um computador remoto. Durante o desenvolvimento do robô um dos objetivos do projeto é a implementação de um sistema de movimentação e geração de trajetória para o robô, baseado na necessidade de se controlar o movimento de maneira autônoma e precisa, de acordo com as configurações do ambiente, por exemplo, seguir um cordão de solda e desviar obstáculos. Além disso, o sistema deve permitir que o robô execute uma trajetória arbitrária pré-programada. O robô deve funcionar em modo autônomo ou controlado remotamente por um operador. Para o controle remoto do AIR-12 foi necessária a definição de um protocolo de

comunicação entre o robô e o computador remoto. Esta etapa do projeto já foi desenvolvida resultando em um sistema flexível o suficiente para que possam ser adicionadas novas funcionalidades ao robô como, por exemplo, o sistema de inspeção e localização. Uma dessas funcionalidades seria o módulo que permitirá que o robô se movimente ao longo de um cordão de solda. Este módulo, que será o escopo deste projeto, é um projeto de *hardware*, codificado em *VHDL*, para processamento digital de imagens (PDI) que será sintetizado em um *Field-programmable gate array* (*FPGA*). Uma câmera captura a imagem do cordão de solda, localizado à frente e na direção de movimento do robô. Sobre este cordão incidem vários feixes de luz estruturada, com faixas emitidas por *lasers*. Essas imagens, após serem processadas, permitirão que um algoritmo desenvolvido neste trabalho atue controlando os motores a fim de ajustar a direção de movimento do robô.

1.1 JUSTIFICATIVA

A soldagem é um processo importante para a indústria de petróleo e gás natural particularmente por ser parte dos processos de fabricação de dutos, tanques de armazenamento, navios e plataformas petrolíferas. Justamente por estarem expostos a um ambiente hostil, os cordões de solda das instalações de uma refinaria ou de um navio, por exemplo, podem ao longo do tempo apresentar trincas, bolhas ou demais defeitos como os que são ocasionados por corrosão. A falta de manutenção na indústria petrolífera pode levar a acidentes, às vezes de grandes proporções, como vazamentos e incêndios, dentre outros, causando poluição ambiental e prejuízos financeiros. Por outro lado, a parada de equipamentos para manutenção por longos períodos, também gera perdas financeiras.

Exemplos recentes de acidentes na indústria de petróleo são os vazamentos de óleo na baía de Guanabara, no Rio de Janeiro, e na Refinaria Getúlio Vargas (REPAR) em Araucária (PR), ambos no ano 2000. Em Janeiro de 2000, ocorreu o acidente provocado por uma falha no duto PE II da Refinaria Duque de Caxias, da Petrobras ocasionando vazamento de 1,3 mil toneladas de óleo. Seis meses após isso, cerca de 4 milhões de litros de óleo cru vazaram do oleoduto OSPAR na REPAR causando o maior acidente ambiental envolvendo a Petrobras nos últimos

26 anos. Neste episódio a empresa foi penalizada com uma multa de R$ 168 milhões. Em maio de 2001, o duto OPASA que interliga Paulínia a São Paulo, na altura de Barueri, próximo a capital paulista, apresentou um vazamento de 200m^3 de RAT, produto derivado de petróleo. Na investigação constatou-se que a causa da ruptura fora a corrosão externa [TERZIAN, 2005]. Sendo que este último acidente está comprovadamente ligado ao processo de inspeção.

Não somente a soldagem antiga, mas também juntas de solda recém-fabricadas, por erro do operador humano ou de sistemas automatizados de soldagem, podem apresentar defeitos e, neste caso, precisam ser refeitas. Portanto, os procedimentos de inspeção são de fundamental importância no setor para identificar defeitos e possibilitar a devida correção, evitando acidentes e seus efeitos, que podem comprometer as demais etapas da cadeia produtiva. O AIR-12 está sendo desenvolvido com intuito de facilitar os procedimentos de inspeção nos setores de armazenamento de produtos da indústria do petróleo. Este projeto, como parte de um trabalho de conclusão de curso, tem por objetivo a pesquisa tecnológica e o desenvolvimento de um produto aplicável à indústria. Porém, o intuito é também desenvolver e capacitar tecnicamente os integrantes do grupo de trabalho do projeto, conferindo a estes algumas noções nas áreas de eletrônica, robótica, automação e controle. No momento do desenvolvimento do módulo de inspeção é necessário que o robô já esteja apto a realizar uma trajetória pré-definida e também que se movimente ao longo de um cordão de solda. O módulo do projeto AIR-12, desenvolvido neste trabalho, trata do desenvolvimento de um módulo de processamento digital de imagem para controlar a trajetória do robô ao longo do cordão de solda. Isto permite que grupos de trabalho futuros adicionem e refinem ainda mais as funcionalidades de inspeção e localização inercial, entre outras funcionalidades.

1.2 OBJETIVOS

O objetivo principal deste trabalho é projetar, implementar, testar e validar experimentalmente um módulo de processamento de imagem que auxilie na movimentação de um robô de inspeção. Este módulo é composto por soluções de

hardware e *software* embarcados que assegurem a movimentação autônoma do robô ao longo de um cordão de solda.

Especificamente, os objetivos do trabalho podem ser descritos como a seguir:

- Adquirir feixes de *laser* paralelos incidindo sobre uma superfície plana ou curva, desde que esta última possua um grande raio, contendo um cordão de solda.
- Identificar e capturar, por meio de uma câmera de vídeo, a imagem do cordão de solda sobre o qual incidem os feixes luminosos.
- Codificar em *VHDL* um circuito para processar as imagens obtidas pela câmera e embarcar este código em um *FPGA*, visando um processamento em tempo real.
- Atuar no controle dos motores ajustando a direção de movimento do robô e fazendo com que este percorra o cordão de solda.

A partir da imagem capturada pela câmera é gerada uma palavra digital de comando que permite controlar os motores para que o robô percorra o cordão de solda de maneira autônoma.

1.3 DIAGRAMA DO SISTEMA

A seguir é apresentado um diagrama de blocos que representa funcionalmente a estrutura do sistema projetado e implementado neste trabalho. O bloco mais acima representa o módulo de interface do sistema do robô com o operador, que é feita através de um computador. A partir deste módulo são enviados os comandos para que o robô inicie o seu deslocamento. O bloco mais elaborado, no centro da Figura 1 - Diagrama de blocos do projeto, é a representação do robô em si. Nele estão contidos os módulos de processamento, acionamento dos motores e o módulo de processamento de imagem, que é o escopo principal deste trabalho. Este último é expandido e pode ser visto no canto inferior direito da figura. O módulo de PDI, como pode ser visto no detalhe, é composto por feixes *laser*, que geram uma faixa de luz vermelha sobre a chapa e o cordão de solda, por uma câmera, que

captura a imagem do padrão luminoso formado pelo *laser*, e por uma FPGA, na qual está sintetizado o circuito que faz o processamento desta imagem.

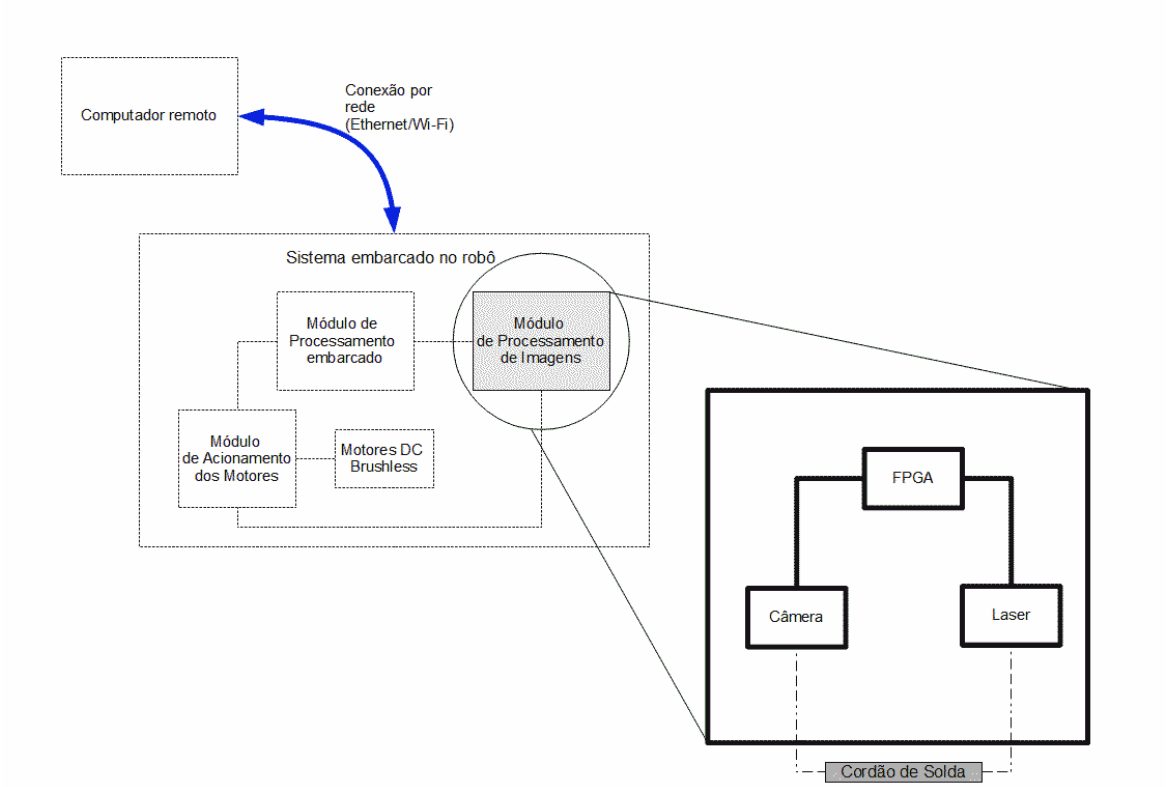

Figura 1 - Diagrama de blocos do projeto

1.4 ORGANIZAÇÃO DO DOCUMENTO

No capítulo 2 deste documento estão descritas as pesquisas tecnológicas realizadas para a proposta de uma solução do problema descrito. Neste capítulo são indicadas ainda as definições de tecnologias utilizadas para cada parte do desenvolvimento inclusive com descrição das especificações componentes e equipamentos utilizados.

No capítulo 3 é apresentado o desenvolvimento, com a modelagem do sistema, bem como com as descrições complementares de equipamentos, *software* e algoritmos utilizados para a implementação de cada funcionalidade. Ainda no capítulo 3 há o relato das dificuldades encontradas durante o desenvolvimento deste trabalho de pesquisa, bem como a descrição da soluções encontradas para a resolução desses problemas de implementação.

Em seguida, no capítulo 4, são apresentados os principais resultado atingidos.

No capítulo 5 é apresentada a gestão do projeto, com análise de custos, de riscos e de pessoal.

Finalmente, as considerações finais são expostas no capítulo 6 com um resumo dos resultados encontrados. Neste capítulo são também explicitadas as lições aprendidas durante a realização do projeto e são também feitas sugestões para trabalhos futuros.

2 DESCRIÇÃO DO SISTEMA E TECNOLOGIAS ENVOLVIDAS

O sistema de locomoção do robô AIR-12 é basicamente composto pelo módulo de controle e acionamento dos motores, que estão centralizados na CPU embarcada, e pelo módulo de processamento digital de imagem, que foi desenvolvido neste trabalho em um kit de desenvolvimento da Altera, enviando os dados resultantes desse processamento para o módulo de controle.

Definidos os módulos de trabalho, foram realizadas pesquisas para verificar quais as tecnologias existentes e qual seria a mais apropriada para aplicação neste projeto. Além disso, foram pesquisados sistemas semelhantes ao sistema proposto. Alguns exemplos são o sistema de visão computacional proposto por (WEI, YING e BAOQUAN, 2009) e o método de controle para robôs móveis apresentado em (SAIDONR, et al. 2011).

Em etapas anteriores do projeto foi implementado um sistema similar, também para análise e deslocamento para inspeção de solda (SANTOS,CASAROTTO,BRESSAM, 2010). Neste trabalho anterior o processamento de imagem foi realizado por software e sobre uma plataforma cedida pela Petrobrás. O diferencial do sistema proposto aqui é o PDI por hardware, além do fato de este ser realizado em uma estrutura mecânica (ROVANI, 2013) e com componentes eletrônicos especificados neste mesmo projeto.

Ainda dentre as tecnologias pesquisadas estão as relacionadas ao sistema de processamento de imagem, ao sistema operacional e ao compilador para permitir o controle remoto do robô. Estas serão descritas a seguir com mais detalhes.

Além disso, existem algumas limitações de hardware determinantes de características do projeto, já que componentes como os motores, *drivers* e a CPU industrial foram especificados em etapas anteriores do projeto. Assim, esses componentes e soluções adotadas também foram estudados para que a equipe dominasse toda a estrutura do robô. Entre os tópicos estudados podem-se citar os seguintes: as tecnologias envolvidas no sistema de comunicação entre os motores e *drivers*, as especificações e limitações da CPU industrial, as especificações da câmera e dos *lasers,* entre outras.

2.1 PROTOCOLOS DE COMUNICAÇÃO

Este projeto envolve a conexão de vários dispositivos como os motores, *drivers*, *lasers* e câmera, em um módulo de desenvolvimento. Para conectá-los à CPU ou à placa DE2, que centralizam o controle do robô e o processamento de imagem respectivamente, é necessário utilizar os tipos de conexões disponíveis em cada *hardware*, ou seja, a USB, a CAN ou a RS232. Para cada uma dessas conexões foi estudado um protocolo apropriado.

2.1.1 Protocolo USB

O protocolo *USB* (*Universal Serial Bus*) fornece um padrão de conexão entre dispositivos. Este protocolo define que o *host* (aquele que recebe as conexões) deve emitir um sinal para verificar quais dispositivos estão conectados, especificando um endereço para cada um. Em seguida o *host* verifica qual tipo de conexão o dispositivo utiliza (Quadro 1) e o dispositivo estará pronto para utilização (ALECRIM, 2013).

Bulk	Para dispositivos com grande volume de dados, tendo recursos para detecção de erro (impressoras e scanners).
Control	Transmissão de parâmetro de controle e configuração de dispositivo.
Interrupt	Para dispositivos com pequeno volume de dados (mouses e teclados).
Isochronous	Para transmissões contínuas, ou seja, não há recurso de detecção de erros (caixas de som).

Quadro 1 - Tipos de conexão de um dispositivo conectado por USB

As versões do protocolo USB foram sendo desenvolvidas à medida que a necessidade de velocidade de transmissão foi aumentando. A primeira versão amplamente utilizada foi a USB 1.1, com velocidade entre 1,5 Mb/s e 12 Mb/s. A segunda versão é a USB 2.0, com 480 Mb/s. Mais recentemente foi lançado o USB 3.0, que permite o tráfego de informação nos dois sentidos e ao mesmo tempo, além de ter a velocidade de transmissão aumentada para até 4,8 Gb/s. No projeto AIR-12 está sendo utilizada a *USB* 2.0.

2.1.2 Protocolo CAN

O protocolo CAN foi desenvolvido inicialmente para aplicações na indústria automotiva, mas, com suas características favoráveis em aplicações de controle e automação, foram aperfeiçoados e desenvolvidos novos padrões baseados em CAN para aplicações próprias em diversos setores industriais. As vantagens do protocolo CAN são (SOUSA; INAMASU; TORRE NETO, 2001):

- possibilidade de configurações para operar com taxas de comunicação de poucos kb/s até 1 mb/s;
- comunicação de dados utilizando dois fios que garante redução de custo e complexidade;
- tamanhos de quadros por quadro otimizado gerando transmissão de dados comuns a dispositivos com redução da ociosidade;
- método de acesso ao meio de transmissão o que evita colisões e permite uma resposta rápida à necessidade de transmissão;
- permite comunicação ponto a ponto, por multidifusão ou por difusão;
- mecanismo de identificação de erros e de tolerância a faltas gerando redes robustas;
- flexibilidade para alteração dos dispositivos.

Este protocolo tem como base o protocolo de transmissão de mensagem orientada, no qual cada mensagem possui um identificador único, que define o conteúdo e a prioridade da mensagem. Com isso, a flexibilidade do sistema aumenta, permitindo adicionar novas estações em uma rede CAN existente sem realizar modificações no software ou hardware de estações preexistentes. O barramento CAN pode ser visualizado na Figura 2 - Esquema barramento CAN, onde é possível verificar o par trançado de fios (*High* e *Low*) com as terminações conectadas a um resistor de 120Ω.

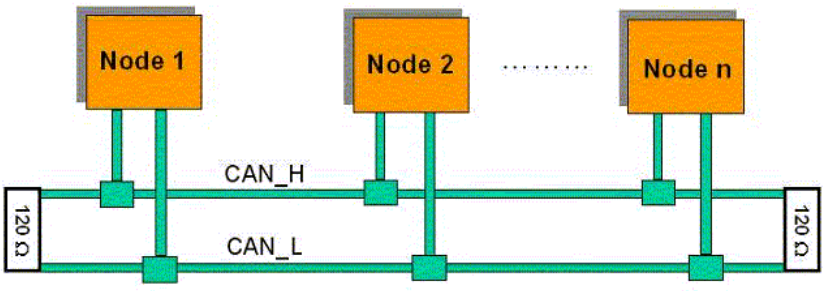

Figura 2 - Esquema barramento CAN

2.1.3 Protocolo RS232

O protocolo RS232 é um padrão de comunicação assíncrona serial, desenvolvido há muito tempo e que continua sendo largamente utilizado na indústria, principalmente por sua confiabilidade. Para implementar esta transmissão de dados serial é utilizado um chip de interface, o *UART* (*Universal Asynchronous Receiver/Transmiter*).

A transmissão de dados utilizando este padrão é assíncrona, sendo transmitido um bit por vez. Com isso, é necessário definir quando a mensagem (*data* bits) inicia e quando termina, sendo utilizados um *start* bit, um ou dois *stop* bit e um bit de paridade (opcional) para cada byte transmitido (Figura 3). Além disso, podem ser definidas diferentes velocidades de transmissão, desde que o transmissor e o receptor estejam configurados com a mesma velocidade.

Figura 3 - Palavra transmitida no protocolo RS232

2.2 ESCOLHA DA CPU

A CPU industrial embarcada ao robô é do fabricante MPL, sendo escolhido o modelo MIP11, Pentium-M com 1.8 GHz de processamento, como mostra a Figura 4. A escolha foi feita em etapas anteriores do projeto (WATANABE et al., 2010). Este modelo de CPU possui entrada serial, USB, Ethernet e PS/2. Para a escolha deste componente não houve estudo comparativo entre modelos e fabricantes, visto que já estava disponível para a equipe no laboratório.

Figura 4 - CPU industrial MIP11

2.3 COMUNICAÇÃO ENTRE A CPU E O COMPUTADOR REMOTO

Como solução para testes em laboratório, foi definida a comunicação através de um cabo Ethernet, entre a CPU embarcada e o computador remoto, conectando o sistema do robô a uma rede pré-definida. Assim, para realizar a comunicação com a rede do laboratório em que os testes são realizados, foi necessário incluir um *switch* junto ao corpo do robô.

2.4 SISTEMA OPERACIONAL

A partir das especificações de projeto foram analisados três possíveis sistemas operacionais (SO) a serem utilizados para o desenvolvimento do robô: Windows, Linux-Ubuntu e Linux-Debian.

O Windows é o sistema operacional que possui uma interface gráfica que facilita a utilização de ferramentas e aplicações no computador. Porém, não costuma ser utilizado nas aplicações industriais, pois este sistema não apresenta a confiabilidade e a segurança necessárias para este tipo de ambiente (NET TUTORIAIS, 2013).

Dentre os sistemas Linux estudados estão o Debian e o Ubuntu, que possuem características semelhantes em relação à segurança, confiabilidade e compatibilidade de *software*. Ao verificar quais aplicações utilizam cada um destes SOs, verificou-se que o Ubuntu é utilizado por usuários para aplicações em

computadores *Desktop*, enquanto o Debian é mais utilizado por desenvolvedores, sendo o mais aplicado em ambientes industriais. Optou-se então pelo Debian, por apresentar características necessárias ao desenvolvimento do projeto.

Ao configurar o processador MIP11 para trabalhar em sua configuração original, percebeu-se que o hardware esquentava até o sistema reiniciar quando atingia uma determinada temperatura máxima. Então, para evitar o aquecimento, a velocidade de processamento foi reduzida para 600 MHz.

2.5 COMPILADOR ECLIPSE

Aplicações em ambientes industriais geralmente trabalham com o SO Linux e com a plataforma de desenvolvimento Eclipse IDE. Para este ambiente de desenvolvimento foi escolhido a versão *Indigo Service Release 2 (1.4.2.201202013-0)*, na qual a linguagem de programação escolhida foi o C++ e o debug remoto foi devidamente configurado para rodar na CPU embarcada.

2.6 PROJETO MECÂNICO

Em projeto de robôs uma das primeiras e principais etapas diz respeito à estrutura mecânica do robô, que, além de permitir a movimentação do sistema como um todo, deve garantir o funcionamento e segurança dos equipamentos eletrônicos que o robô transporta. O robô AIR-12 possui uma estrutura capaz de receber dois motores que movimentam, de maneira independente, cada lado do robô, como evidenciado na Figura 5. As rodas laterais são ligadas através de uma correia dentada, permitindo que cada motor garanta a força para a movimentação de cada par de rodas. Além disso, foi desenvolvido um sistema de cambagem, que permite mais liberdade ao robô principalmente ao fazer curvas e ao passar por obstáculos, como buracos ou pequenas saliências. O diferencial da estrutura mecânica deste robô são as rodas magnéticas, que são fortes o suficiente para a adesão e movimentação em estruturas metálicas não sendo necessária muita força para retirá-lo do ambiente de teste. Mais detalhes do projeto mecânico do robô AIR-12 podem se encontrados em (ROVANI, 2013).

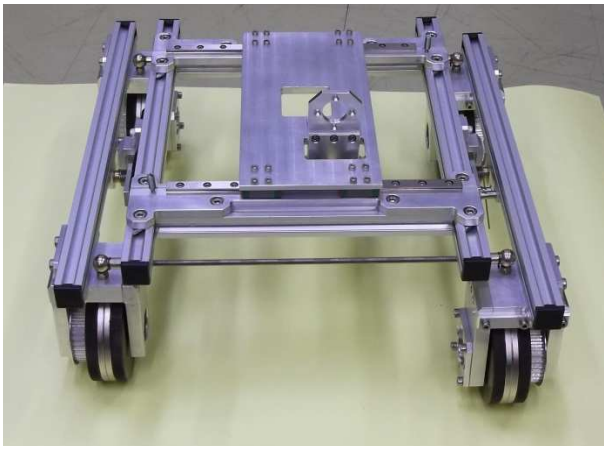

Figura 5 - Estrutura mecânica do robô

2.7 MOTORES E DRIVERS

A partir do projeto mecânico definiu-se que o robô utilizaria dois motores para realizar a movimentação das quatro rodas. Assim, calculando-se a potência necessária para movimentar o robô, foram especificados motores e *drivers* da marca Maxon Motors (Figura 6). O *driver* é do modelo EPOS2 70/10. O conjunto do motor é composto por um motor DC *brushless*, acoplado a um *encoder*, a um sensor *hall* e a uma caixa de redução, sendo o controle desses componentes realizado por meio do *driver* EPOS2 70/10. Este componente suporta a comunicação com um computador através dos protocolos USB, CAN ou RS232. Além disso, possui entradas e saídas digitais, que podem ser utilizadas para a troca de dados com um sensor, por exemplo.

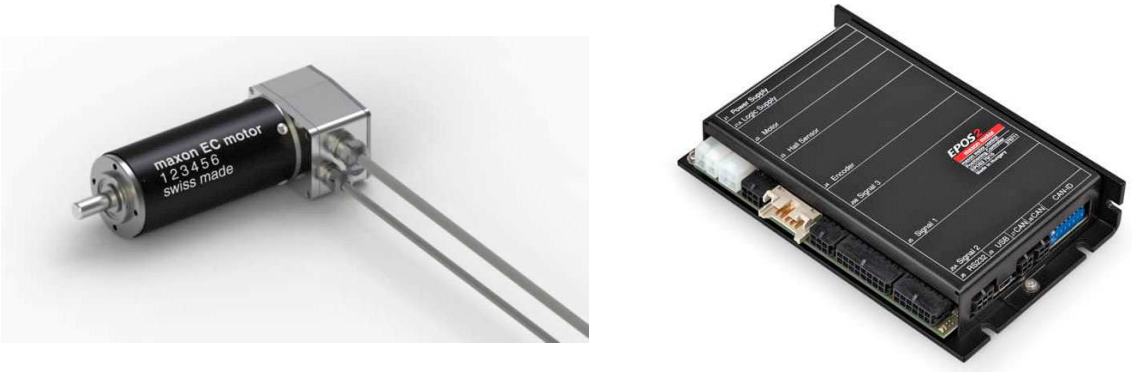

Figura 6 - Motor e *Driver*

O acesso a essas configurações iniciais e aos comandos dos motores é realizado através das funções codificadas em C++ de uma biblioteca (*Definitions.h*) disponibilizada pelo fabricante desses equipamentos (NATIONAL INSTRUMENTS, 2013). Para a alimentação dos motores e dos *drivers* é utilizada uma fonte externa de 48V/480W.

Para o robô AIR-12 definiu-se a entrada USB como meio de comunicação entre os *drivers* dos motores e a CPU, pois esta porta está disponível nos dois dispositivos e também por possuir uma interface que não necessita de muitas configurações. Além disso, foi definido o barramento CAN como a comunicação entre os dois *drivers*, sendo o mestre conectado diretamente na CPU através da USB e os escravos definidos através de uma DIP *Switch*, como mostra a Figura 7.

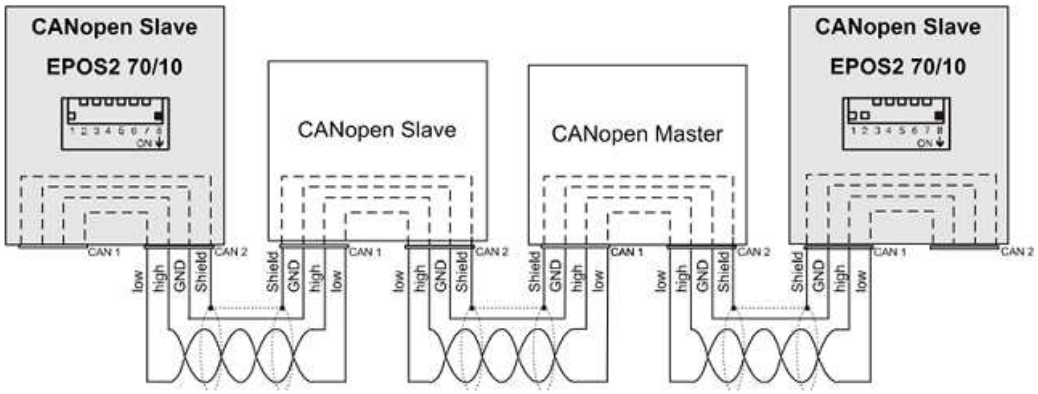

Figura 7 - Modelo do barramento CAN Motor/Driver

Antes de colocar os motores na estrutura do robô, foram realizados testes de desempenho com o auxílio software EPOS Studio, verificando as limitações da configuração em relação à velocidade e à aceleração. Assim, limitou-se a velocidade máxima do robô em 8000 rpm (com caixa de redução) e a aceleração e a desaceleração máxima em 5000 rpm/s (com caixa de redução).

2.8 CONTROLE DOS MOTORES

Durante o desenvolvimento do controle do robô verificou-se que, apenas com a manipulação das grandezas dos motores, como velocidade e aceleração, o controle da movimentação não era suave como desejado, principalmente porque o

robô deve transportar equipamentos de inspeção baseados em ultrassom que são sensíveis a vibração e a paradas e acelerações bruscas. Com isso, foi iniciado um estudo mais aprofundado sobre o controle dos motores.

Para simular a situação de controle do robô em relação a uma posição determinada, como é o caso do controle da movimentação do robô ao longo do cordão de solda, foi adicionado à estrutura do robô o sensor SRF02 da Devantec. A abordagem utilizada foi a de fazer com que o robô seguisse uma trajetória paralela a uma parede, com uma distância inicialmente especificada. Esta abordagem foi utilizada para acelerar o estudo da estratégia de controle do robô, nesta situação, enquanto a interface de aquisição de imagem ainda estava em desenvolvimento. O SRF02 (Figura 8) é um sensor de ultra-som utilizado para verificar distâncias de 0.2 à 3m. Este sensor pode se comunicar com o módulo de controle através do modo I2C ou do modo Serial.

Figura 8 - Sensor SRF02

Para esta aplicação, optou-se pela comunicação serial, que utiliza o pino receptor para receber os sinais de comando e o pino transmissor para enviar a distância captada. Para conectar este sensor na RS232 da CPU é necessário realizar a conversão de TTL para o padrão EIA-232, utilizando um circuito integrado MAX232 (Figura 9).

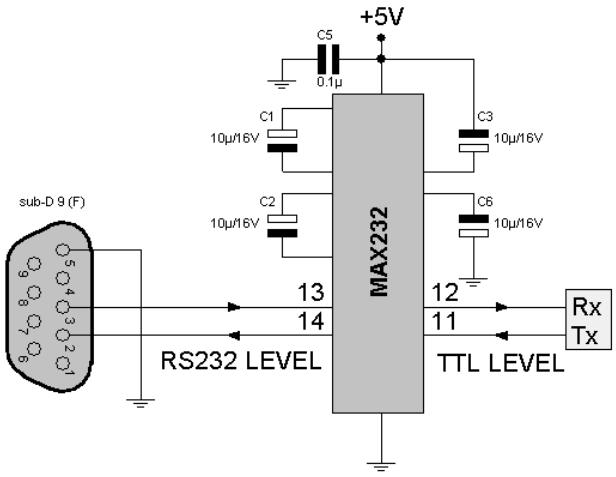

Figura 9 - Circuito utilizado para converter TTL no padrão EIA-232

Configurada a conexão entre o sensor e a CPU, o desenvolvimento do *software* foi realizado na plataforma do Eclipse, utilizando o mesmo programa de acionamento e controle dos motores em C++. Assim, foi criada uma função para recepção dos dados vindos do sensor e para a transmissão de comandos de iniciar ou terminar as medidas de distância.

2.9 SISTEMA DE PROCESSAMENTO DE IMAGEM

Em uma câmera analógica, as lentes focalizam a luz de uma cena sobre um filme tratado com produtos químicos que possuem uma reação controlada pela intensidade luminosa incidente. Desta forma, o filme fotográfico grava a cena à sua frente, ou seja, absorve maiores quantidades de luz das partes mais claras da cena e menores quantidades de luz das partes mais escuras. Em uma câmera digital este processo é feito através de um sensor fotossensível. As câmeras digitais podem ter sensores de luz construídos com duas tecnologias diferentes: CCD (Charge-Coupled Devices) e CMOS (Complementary Metal-Oxide Semiconductor). Ambas as tecnologias de sensores de luz, CCD e CMOS, tem como objetivo a conversão de imagens em sinais elétricos. Estes dispositivos medem a luz com um painel formado por pequenos diodos fotossensíveis chamados pixels.

O princípio de funcionamento dos sensores CCD, utilizados neste trabalho, é descrito a seguir: trata-se de um material semicondutor, de modo que, quando exposto à luz, cargas livres são geradas, as quais crescem em quantidade de

acordo com a intensidade da luz e o tempo de exposição. As cargas em cada pixel são coletadas em cada leitura, dando origem a uma tensão analógica proporcional à quantidade de cargas, a qual é posteriormente convertida em uma tensão digital por um conversor analógico-digital, instalado no próprio CCD ou fora dele. A diferença entre esta tecnologia e a CMOS é que na última são utilizados pixels ativos, os quais têm acesso individual a seu próprio amplificador sensor, como evidencia a Figura 10.

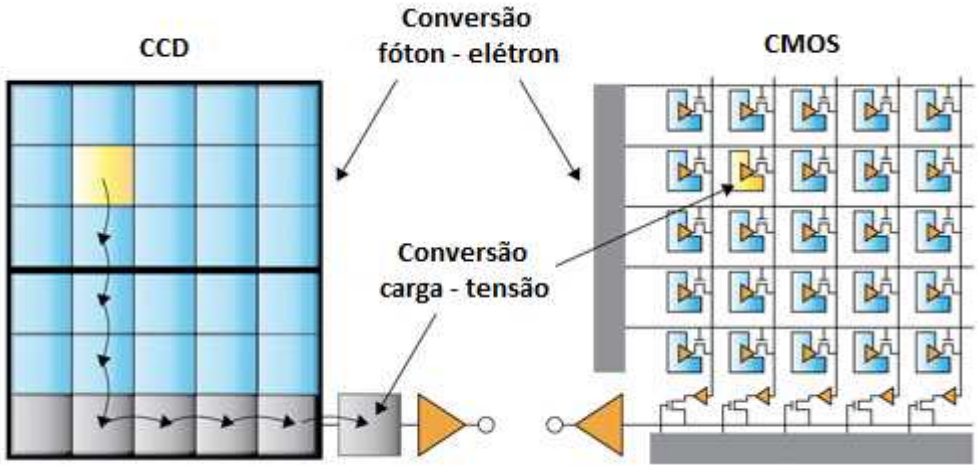

Figura 10 - Diferença construtiva entre sensores CCD e CMOS

Os sensores CCD fornecem imagens de alta qualidade e com baixo ruído. Os sensores CMOS, embora sejam mais rápidos, são mais suscetíveis ao ruído. Como cada pixel em um sensor CMOS tem vários transistores localizados próximos a ele, a sensibilidade à luz de um chip CMOS tende a ser menor. Muitos dos fótons que atingem o chip colidem com os transistores em vez do fotodiodo. Sensores CMOS tradicionalmente consomem pouca energia já aqueles do tipo CCD consomem mais energia. CCDs consomem até 100 vezes mais energia do que o sensor CMOS equivalente. Os chips CMOS podem ser fabricados em praticamente qualquer linha de produção de silício padrão, então eles tendem a ser extremamente baratos em comparação com os sensores CCD. Por outro lado, sensores CCD têm sido produzidos em massa por um longo período de tempo, por isso eles são mais estabelecidos no mercado e tendem a ter maior qualidade e mais pixels, o que proporciona uma melhor resolução.

A câmera utilizada neste projeto é do tipo CCD, devido ao baixo custo e à alta disponibilidade no mercado. Para este projeto foi especificada uma câmera CCD (especificação: NS-C3901N), mostrada na Figura 11, com resolução de 640x480,

mesma resolução do padrão VGA, fabricado pela empresa Sony. A câmera disponibiliza imagens no padrão NTSC. O NTSC (National Television System Committee) é um padrão de vídeo composto que é um formato de TV analógica.

Figura 11 - Câmera com CCD modelo NS-C3901N da Sony

2.10 LASER

O Laser (Light Amplification by Stimulated Emission of Radiation), sigla cuja tradução é “Amplificação da Luz por Emissão Estimulada de Radiação”, é um dispositivo que produz radiação eletromagnética com características específica. A luz emitida é monocromática, ou seja, possui comprimento de onda muito bem definido, é coerente, pois todas as ondas dos fótons que compõe o feixe estão em fase, e é colimada, ou seja, propaga-se como um feixe de ondas praticamente paralelas. Por essas características que o laser apresenta uma potencia maior que fontes de luz visível comuns.

Para a realização deste projeto foram especificados emissores de feixe laser de baixo custo e também de baixa potência. O gerador de feixes especificado, ilustrado na Figura 12, funciona com um valor de tensão de 5V e potência de 5mW.

Figura 12 - Gerador de feixe laser de cor vermelha

Através de lentes, que acompanham o emissor laser, o feixe emitido é transformado em uma linha. Sistemas assim são chamados de sistemas estruturados de luz. Em visão artificial, utiliza-se a luz estruturada para iluminar um objeto, a partir de um ângulo conhecido, com um padrão de luz específico, no caso, uma linha. Observando a forma como se altera este padrão sobre o objeto, pode-se obter informação sobre a terceira dimensão do objeto iluminado. Esta é a técnica empregada para a detecção do cordão de solda.

2.11 PLACA DE DESENVOLVIMENTO DE2

Como já especificado, este projeto se caracteriza pelo desenvolvimento de um módulo de processamento de imagem em tempo real que, através de um algoritmo desenvolvido, define a trajetória do robô AIR-12 ao longo de um cordão de solda em um tanque de armazenamento. Este processamento é realizado por hardware. Esta abordagem foi escolhida, devido ao fato de o processamento por hardware oferecer velocidades superiores ao processamento de imagem por software. A diferença principal entre essas duas opções de processamento é que em *software* o processador é compartilhado por diversas tarefas, enquanto que em *hardware* o circuito é dedicado exclusivamente ao PDI.

A câmera, descrita acima, realiza a captura da imagem do padrão luminoso formado pelas faixas de laser sobre o cordão. Esta imagem é então processada por um circuito codificado em *VHDL (Very High Speed Integrated Circuit Hardware Description Language)*, que é sintetizado em um *FPGA (Field-programmable gate array)*. Para tanto foi especificada uma placa de desenvolvimento, que já inclui vários chips de interface para diversos protocolos de trocas de dado, descritos na tabela abaixo. A placa escolhida foi a DE2 da empresa TerasIC, representada na Figura 13 e no Quadro 2, que utiliza a FPGA Cyclone II da empresa Altera. A escolha desta placa foi escolhida pelo fato de contar com um conversor analógico-digital e outro digital-analógico específicos para aplicações de vídeo. Outra vantagem é que todos os pinos destes componentes já se encontram roteados na placa com a FPGA e possuem conectores prontos e convenientemente disponíveis nas laterais da DE2.

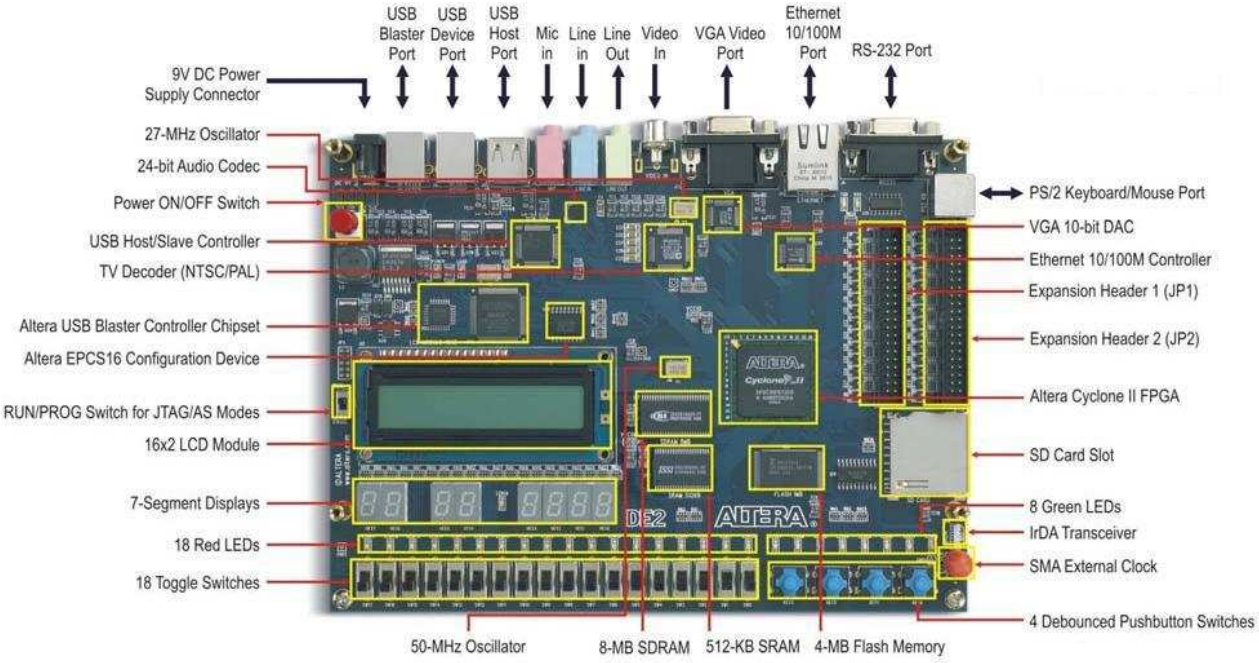

Figura 13 - Kit de desenvolvimento DE2

Funcionalidade	Descrição
FPGA	Cyclone II EP2C35F672C6
I/O Interfaces	Interface USB-Blaster para configuração da FPGA
	Entrada para Microfone (24-bit Audio CODEC)
	Saída de Vídeo (VGA 10-bit DAC)
	Entrada de Vídeo (NTSC/PAL/Multi-format)
	Serial RS232
	Porta IrDA
	Porta PS/2 para *mouse* ou teclado
	10/100 Ethernet
	USB 2.0
	80 pinos de entrada e saída para uso genérico
Memory	8 MB SDRAM, 512 KB SRAM, 4 MB Flash
	Slot para cartão de memória SD
Displays	8 displays de 7 segmentos
	display LCD 16 x 2
Switches e LEDs	18 *toggle switches*
	18 LEDs vermelhos
	9 LEDs verdes
	4 *pushbuttons*
Clocks	50 MHz *clock*
	27 MHz *clock*
	Entrada SMA para *clock* externo

Quadro 2 - Descrição das funcionalidades oferecidas pela placa DE2

Mais especificamente quanto à FPGA, tem-se que é do modelo Cyclone II EP2C35F672C6N da marca Altera, como já citado. Esta contém 33 mil componentes lógicos, com 475 pinos de entrada/saída disponíveis para o usuário. Estes são distribuídos em um encapsulamento FBGA (*Fine pitch Ball Grid Array*). A FPGA é utilizada para sintetizar todos os circuitos necessários para o processamento digital de imagens, excluindo-se os conversores de vídeo já citados.

2.12 INTERFACE DE VÍDEO VGA

Embora a interface VGA não fosse um requisito do projeto, esta foi estudada e adicionou-se ao projeto um módulo que permite enviar a imagem processada a um monitor de computador. Assim foi possível visualizar a imagem de saída do sistema. Esta interface tornou mais eficiente o processo de *debug* durante o desenvolvimento do módulo de aquisição e processamento de imagem.

VGA (Video Graphics Array) é um padrão de interface de vídeo analógico. A resolução mais comum aceita pelos monitores com entrada analógica VGA é de 640 x 480 x 60 Hz, que consiste de 640 colunas, 480 linhas e 60 quadros por segundo. Este padrão emprega três cores por pixel: vermelho, verde e azul. Este padrão de cores é conhecido como RGB (R = *red*, G = *green*, B = *blue*). A intensidade de cada cor é definida por um valor determinado tensão que varia entre 0 V e 0,7 V.

O sinal digital resultante do processamento de imagem deve ser enviado para o chip que controla o sinal que sairá para o monitor VGA. Esse chip, o conversor ADV123 que está disponível na placa da FPGA, recebe o sinal digital e o converte para analógico. Este sinal é então enviado para o monitor.

Este conversor recebe um pixel digital por vez, com tamanho de 10 bits, e também sinais de sincronismo. Então o sinal é convertido para analógico e enviado para o conector VGA. O conector VGA recebe os sinais analógicos dos pixels (três componentes, RGB) e os sinais digitais de sincronia. Estes sinais são recebidos pelo controlador do monitor, o qual irá imprimir os pixels na tela, na sequencia determinada por sinais de sincronismo.

Toda a sequencia é cadenciada pelo sinal de *clock*. São gerados no código cinco sinais de sincronismo, os quais podem ser observados na Figura 14 e na Figura 15. Os sinais são denominados *Hsync* para o sincronismo horizontal, *Vsync*

para o sincronismo vertical, *Hactive* que corresponde a parte do sinal *Hsync* em que os pixels são enviados, *Vactive* que corresponde a parte do sinal *Vsync* em que linhas de pixels são enviados e o sinal *dena* que habilita o envio para o display.

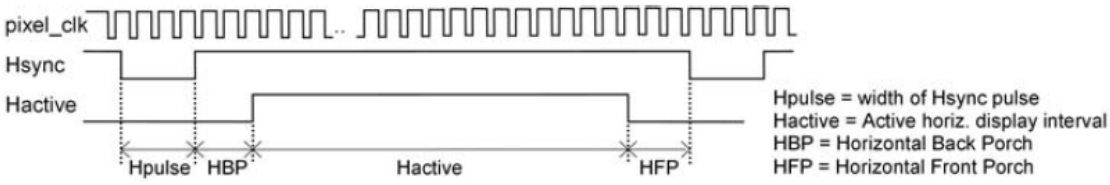

Figura 14 - Sinais de sincronismo horizontal

Fonte – (PEDRONI, 2010, p. 428)

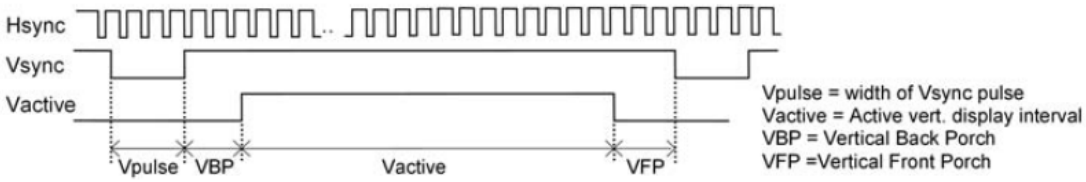

Figura 15 - Sinais de sincronismo vertical

Fonte – (PEDRONI, 2010, p. 429)

Os sinais *Hsync* e *Vsync* determinam quando uma nova linha ou quadro devem começar. *Hactive* e *Vactive* representam o intervalo de tempo que os pixels vão realmente para o monitor. E o *dena* é responsável por manter o sinal de pixels desabilitado no intervalo de tempo entre o final de uma linha e o começo da próxima ou entre o fim de um quadro e o começo do próximo (PEDRONI, 2010).

Os valores dos sinais de sincronismo são valores tabelados e a escolha depende do padrão adotado pelo projeto.

O conversor D/A utilizado apresenta uma característica especial, ele possui a possibilidade de fazer o sincronismo por meio da combinação da dos pulsos de sincronização horizontal e vertical com o sinal do verde (sync-on-green), eliminando a necessidade de receber os sinais Hsync e Vsync. A distinção entre esses três sinais é facilmente detectada pela diferença de tensão.

Portanto os sinais enviados para o DA são os três componentes de cor, um sinal de sincronismo e um sinal que representa o mesmo que o dena.

2.13 DEFINIÇÃO DO SISTEMA FINAL

Esta seção visa recuperar e reunir as principais tecnologias pesquisadas e envolvidas no desenvolvimento do robô, para que assim o sistema possa ser caracterizado de forma clara e sucinta.

A parte mecânica utilizada, proposta em (ROVANI, 2013), é a estrutura primordial do robô e foi um projeto realizado para superar os problemas apresentados pela plataforma cedida pela Petrobrás, que foi utilizada por (SANTOS, CASAROTTO e BRESSAM, 2010).

Outro componente fundamental para a construção do robô é a sua Unidade Central de Processamento. Definido o modelo MIP11 de empresa MPL, este componente foi especificado no trabalho realizado por (WATANABE, BORBA e OLEINIK, 2010). Este módulo de processamento está conectado a um switch que permite que o sistema seja conectado a qualquer rede, para permitir o seu acionamento remoto. Outra conexão importante é entre a CPU e os *drivers* dos motores, feita com o barramento USB. Quanto à parte de locomoção propriamente dita, os *drivers* especificados se comunicam entre si através do protocolo CAN.

A FPGA recebe da câmera, por meio de uma entrada RCA, as imagens, no formato NTSC, do padrão luminoso formado pelos feixes de laser. A imagem é processada e a comunicação entre o módulo de processamento de imagem e o MIP11 é realizada pela porta serial RS-232. Para facilitar o processo de debug foi também incluído um módulo de saída de vídeo para um monitor VGA.

Com a estrutura do projeto definida, o próximo capítulo trás o relato de como o projeto foi de fato implementado e quais foram as principais dificuldades técnicas encontradas.

3 DESENVOLVIMENTO

3.1 PROCESSAMENTO DIGITAL DE IMAGEM

Durante a definição do escopo deste projeto foram realizadas pesquisas bibliográficas envolvendo temas como: processamento de imagem em FPGA (PEDRONI, 2011), simulações de processamento de imagem em MATLAB (GONZALES, 2004), sistemas autônomos e de controle de movimento (SAIDORN, et al., 2011) desenvolvidos na FPGA, dentre outros. Assim, com os objetivos definidos, foi montado um plano de trabalho com as abordagens consideradas adequadas para resolver determinados problemas.

É clássico de projetos da área de controle analisar a malha na qual se deseja intervir e tomar decisões baseadas em um sinal de erro realimentado de alguma variável sensível do sistema (OGATA, 2012). Este sinal de erro é calculado a partir da medida de um sensor comparado com um sinal de referência. Neste projeto o robô representa a planta de controle e se deseja controlar a sua trajetória, fazendo com que o equipamento siga o cordão de solda. É necessário então gerar o sinal de erro baseado na localização do cordão de solda em relação a posição assumida pelo robô. A abordagem utilizada para a obtenção deste sinal foi a de processar digitalmente a imagem do cordão de solda, a partir de um circuito codificado em *VHDL* e implementado através de um *kit* de desenvolvimento com uma *FPGA*. Assim tem-se uma aplicação em tempo real que pode ser considerada, do ponto de vista de controle, como um sensor gerando um sinal de erro.

A ferramenta de software utilizada como plataforma de estudos que auxiliou na compreensão do problema foi o MATLAB, no qual foram realizados testes com uma câmera e com as imagens capturadas. Foi desenvolvido inicialmente, um código base no MATLAB para o algoritmo de processamento de imagem que auxiliou na sua implementação posterior em VHDL. Portanto, para evitar problemas de portabilidade, foi utilizado o mínimo de funções específicas de PDI do MATLAB, sendo a maior parte das rotinas de desenvolvimento próprio. Paralelo a este estudo, foram realizados testes na FPGA, com exemplos disponibilizados pela Altera, visando à familiarização com este ambiente de desenvolvimento.

No código em MATLAB são definidas as técnicas de PDI fundamentais para a filtragem e aprimoramento da imagem, de maneira que o algoritmo a ser

desenvolvido na FPGA tornou-se mais facilmente estruturado. Assim o código implementado em VHDL, realiza a aquisição e processamento da imagem a partir da entrada RCA da DE2. Também foi desenvolvido o código que analisa geometricamente a imagem, utilizando artifícios de programação em álgebra, cálculo numérico e probabilidade e estatística.

A partir de conceitos de processamento digital de imagem e de programação em hardware utilizando uma FPGA, o algoritmo foi estruturado em forma de fluxograma e/ou máquina de estados. Com isto, utilizando o MATLAB como software auxiliar, foram realizadas simulações do módulo de processamento de imagem, sendo possível verificar e propor soluções aos problemas que foram encontrados durante o desenvolvimento do código em VHDL na FPGA, como por exemplo, ruído luminoso e saturação da imagem.

3.1.1 Filtro por cor

A primeira tentativa de filtragem, para extrair apenas o perfil luminoso formado pelo laser sobre o cordão de solda foi uma abordagem por filtragem de cor. As imagens analisadas estavam no formato *raster*. O armazenamento de imagens exige uma grande quantidade de bytes, por isso existem diversas formas de armazenar os arquivos de imagem em função do propósito de utilização. Dois modos de representação básicos podem ser utilizados para compor imagens:

- *raster*;
- vetorial.

Na representação vetorial a imagem é descrita através dos parâmetros de suas formas geométricas. Neste formato armazenam-se apenas as coordenadas de pontos, linhas e polígonos que compõe a imagem. Este formato exige menos memória, porém não é o padrão disponibilizado pelas câmeras comerciais.

Apesar de o formato *raster* ocupar mais espaço do que o formato vetorial, esse formato é mais simples que o anterior. No formato *raster* a imagem é representada por uma matriz de duas dimensões, sendo uma a largura e a outra a altura, cujos valores são chamados de *pixels* (Figura 16). Cada um destes *pixels* é composto por três valores que representam uma cor no sistema de cores RGB.

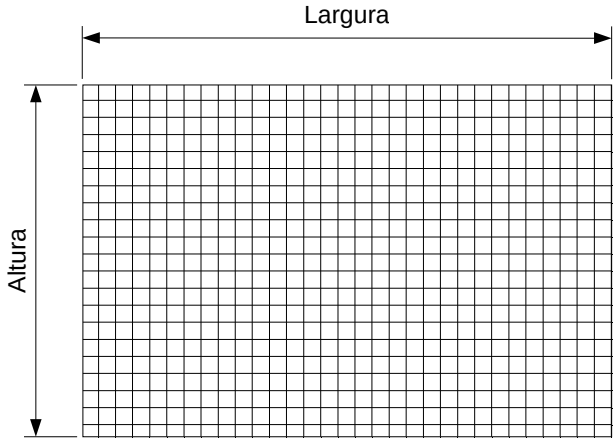

Figura 16 - Estrutura de uma imagem digital

RGB é a abreviatura do sistema de cores aditivas formado por Vermelho (Red), Verde (Green) e Azul (Blue). O propósito principal do sistema RGB é a reprodução de cores em dispositivos eletrônicos como monitores de TV e computador, *datashows*, *scanners* e câmeras digitais, assim como na fotografia tradicional.

Uma cor no modelo de cores RGB pode ser descrita pela indicação da quantidade de vermelho, verde e azul que contém. Cada uma pode variar entre o mínimo (completamente escuro) e máximo (completamente intenso). Quando todas as cores estão no mínimo, o resultado é preto. Se todas estão no máximo, o resultado é branco.

Uma das representações mais usuais para as cores é a utilização da escala de 0 a 255, que possibilita a representação de cada valor de cor em 1 byte (8 bits). Assim, o vermelho completamente intenso é representado por (255, 0, 0), por exemplo conforme mostra o Quadro 3.

	Componentes		
Cor	RED	GREEN	BLUE
Branco	255	255	255
Vermelho	255	0	0
Verde	0	255	0
Azul	0	0	255
Amarelo	255	255	
Magenta	255	0	255
Ciano	0	255	255
Preto	0	0	0

Quadro 3 - Distribuição de valores na escala RGB

Apesar de os feixes *laser* analisados nesse trabalho serem vermelhos, devido à variação de luminosidade no ambiente e do próprio espalhamento causado pelas características construtivas do emissor de feixes laser, os pixels correspondentes ao laser raramente tem valores exatos iguais a [255,0,0], que seria o vermelho puro no sistema RGB. Então é preciso fazer a filtragem de cor com a utilização de *thresholds* (Figura 17) para limitar as componentes de cor em faixas específicas da imagem que foi considerado vermelho. Estes níveis de *threshold* foram determinados através da análise das imagens obtidas em laboratório. Deseja-se descobrir quais *pixels* são vermelhos para poder-se binarizar a imagem. Aquilo que é vermelho é jogado para o valor de *foreground*, por exemplo, 1 (um) ou 255; caso contrário atribui-se o valor de *background*, que complementaria os valores 1 ou 255, como o valor 0 (zero).

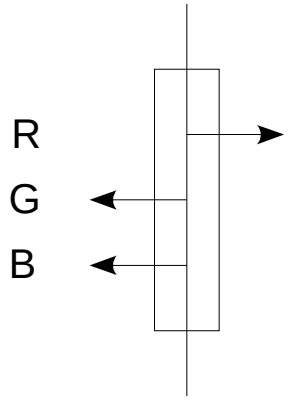

Figura 17 - Esquema de thresholding das cores em RGB

Considera-se um ponto válido como vermelho um *pixel* de componente R acima de um certo nível e componentes G e B abaixo de outros níveis específicos.

O resultado esperado é uma imagem preta e branca com o padrão formado pelo laser sobre o cordão de solda. Abaixo, na Figura 18, tem-se um exemplo bem sucedido desta abordagem.

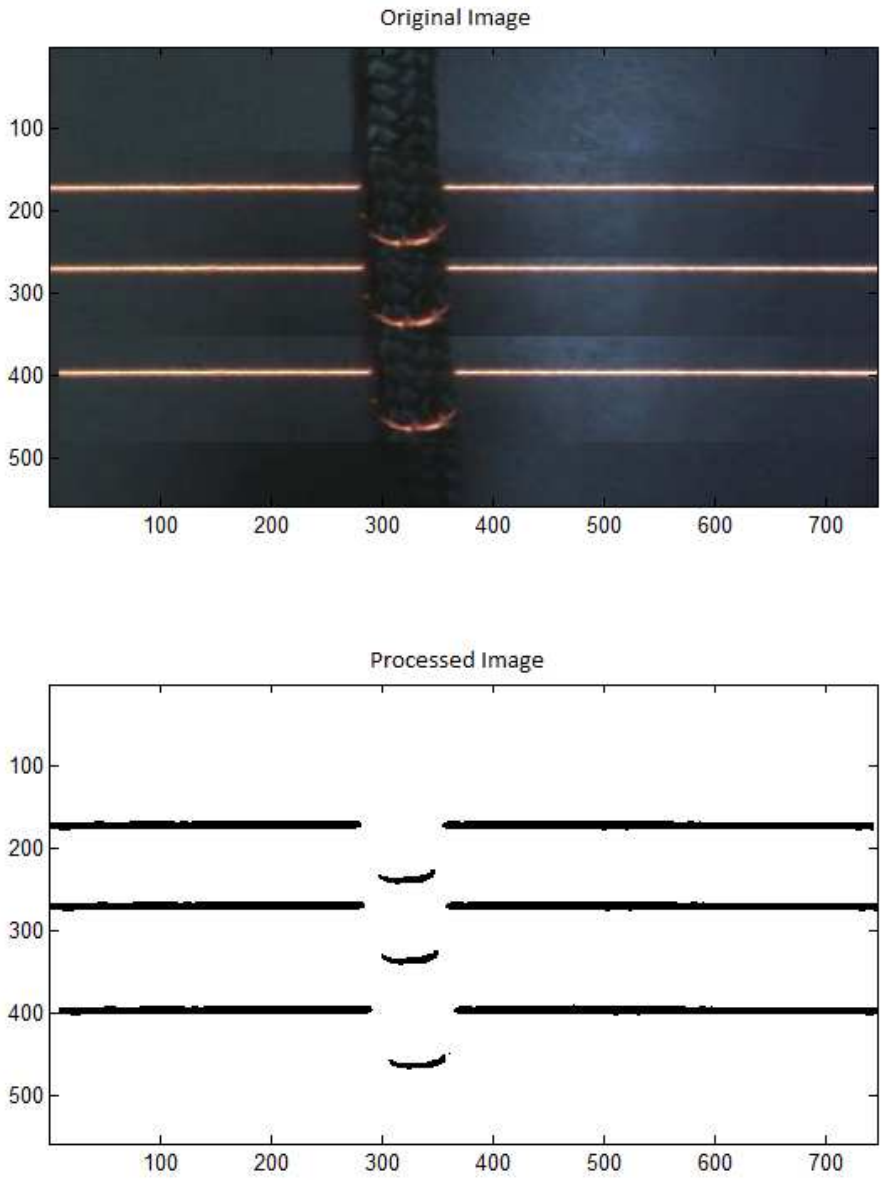

Figura 18 - Tentativa bem sucedida da filtragem por cor

A imagem original, encontrada na metade de cima da figura, foi retirada do trabalho de [SANTOS, CASAROTTO e BRESSAM, 2010] e auxiliou no desenvolvimento inicial do algoritmo de filtragem. Porém, quando se foi aplicar este mesmo processo às imagens obtidas nos primeiros testes do projeto verificou-se que a filtragem por cor seria insuficiente para uma solução mais robusta.

Como já citado anteriormente, embora o objetivo do trabalho seja uma solução embarcada, optou-se por definir o algoritmo de processamento de imagem em ambiente MATLAB. Pois este é uma ferramenta de software que facilita a verificação de resultados, o que é altamente desejável em um ambiente de

pesquisa. A metodologia utilizada foi utilizar uma *webcam* para capturar a imagem de um feixe laser incidindo sobre a bancada do laboratório. Esta câmera, que estava disponível no laboratório, foi escolhida devido à facilidade de integração com o MATLAB. O próprio software oferece uma *toolbox* para aquisição de imagens, que tem uma interface com dispositivos do tipo *webcam*. Logo ao início dos testes foi verificado que a luminosidade ambiente interferia no padrão observado, tornando impossível tomar qualquer decisão, mesmo visualmente, em relação à localização do feixe *laser*.

Este problema foi parcialmente resolvido com a alteração dos níveis de *thresholds*. Porém foi identificado outro problema. Só era possível identificar o contorno do feixe luminoso. Na imagem abaixo, Figura 19, é possível ver que o *laser* capturado pela câmera apresenta saturação na parte central do feixe, aproximando-se da cor branca, Esta saturação é confundida pela filtragem com a saturação causada pela iluminação ambiente. Portanto tornou-se necessária uma abordagem mais refinada para filtrar o perfil formado pelo laser.

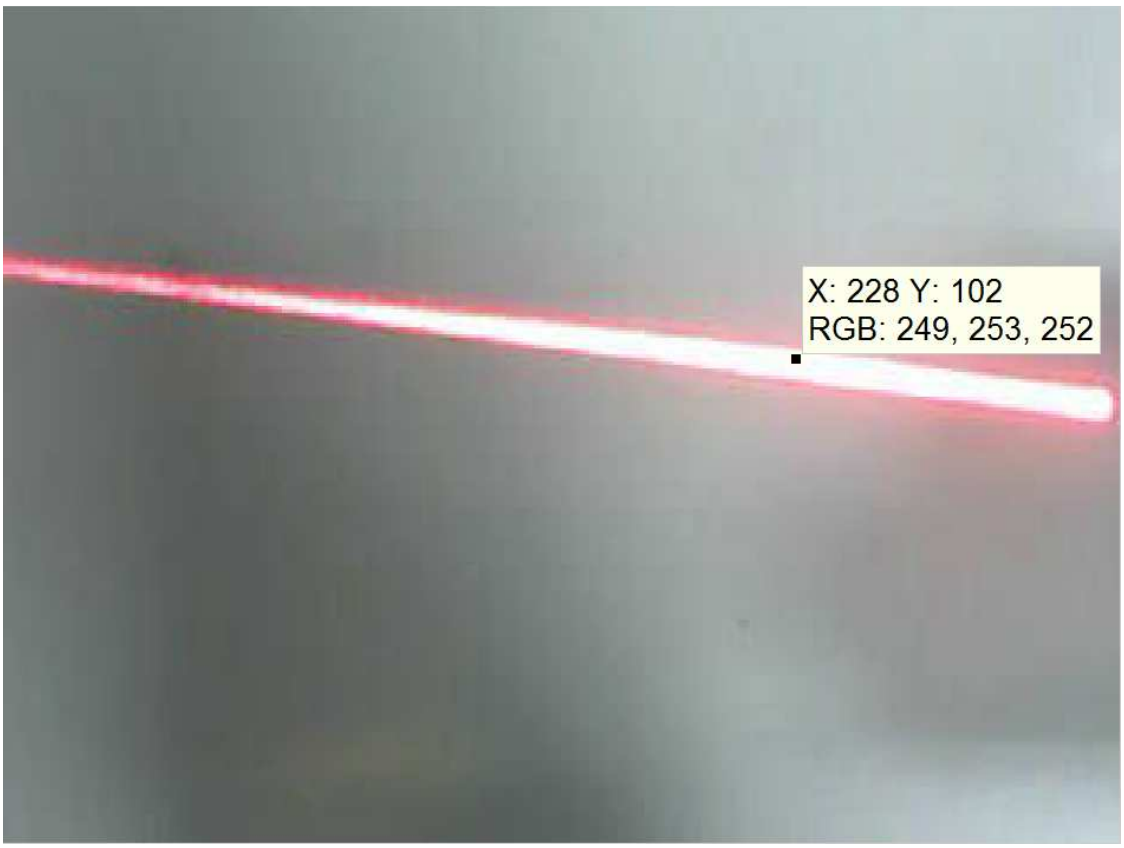

Figura 19 - Saturação da parte interna do feixe de laser

3.1.2 Filtragem Morfológica

Observando a Figura 20 pode-se notar que há uma saturação para o branco no meio do feixe luminoso, impossibilitando a identificação do padrão formado pelo *laser* através da filtragem por *thresholding* de cor. Na tentativa de aplicar-se esta filtragem na Figura 20 tem-se como resultado a Figura 21 onde se tem a noção visual do contorno desse feixe luminoso.

Figura 20 – Imagem do laser com saturação no meio do feixe luminoso.

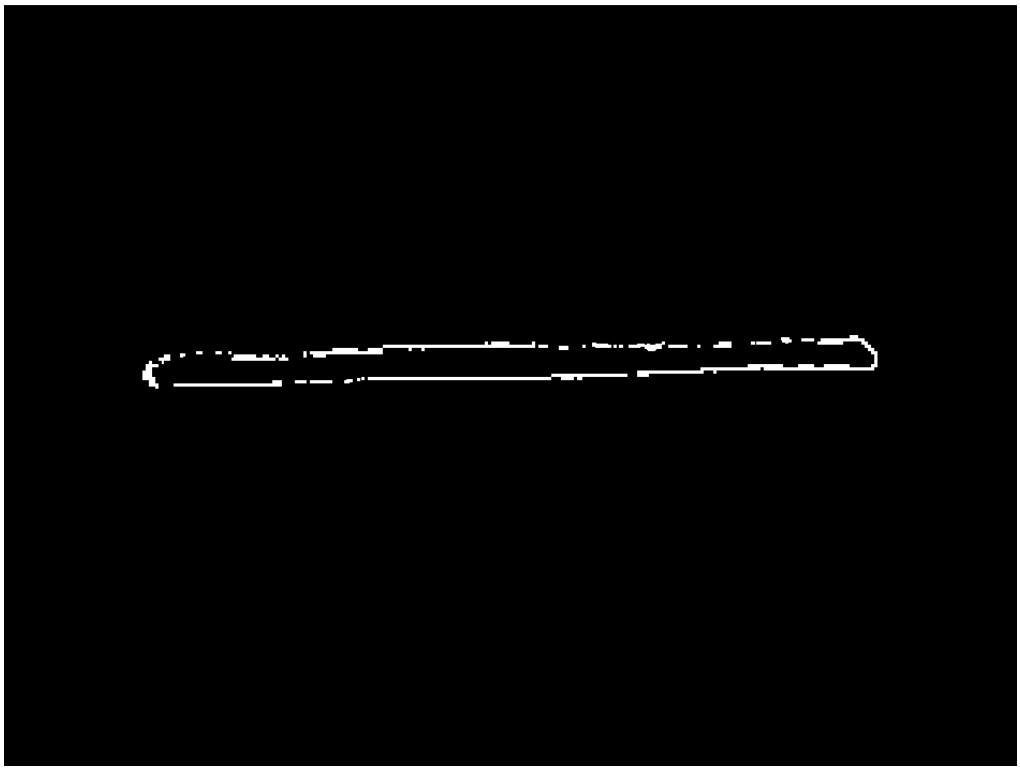

Figura 21 – Resultado do thresholding por cor

Para poder aproveitar este contorno e extrair a informação visual do padrão luminoso formado pelo *laser*, foram estudadas técnicas de processamento morfológico de imagem. O primeiro passo consiste na aplicação da operação de fechamento. Esta operação torna o contorno visto na imagem anterior em uma região conexa.

O fechamento morfológico (GONZALES, 1992) da imagem consiste de duas operações primárias em processamento morfológico de imagem, a saber: Dilatação e Erosão. Erosão e dilatação são operações elementares de processamento morfológico e formam a base para a construção de transformações mais complexas, como abertura e fechamento. Assim, numa cadeia morfológica de processamento de imagens, encontra-se um grande número de operadores encadeados, todos definidos a partir destas funções elementares definidas para os conjuntos A e B. O conjunto B é normalmente chamado de elemento estruturante.

Erosão: a erosão de A por B pode ser definida pelo conjunto de todos os pontos *z*, os quais B, transladados por *z*, são contidos em A. Esta operação pode ser evidenciada pela expressão a seguir.

$$A \ominus B = \{z \epsilon E | Bz \subseteq A\} \quad (1)$$

A Figura 22 evidencia a operação de erosão.

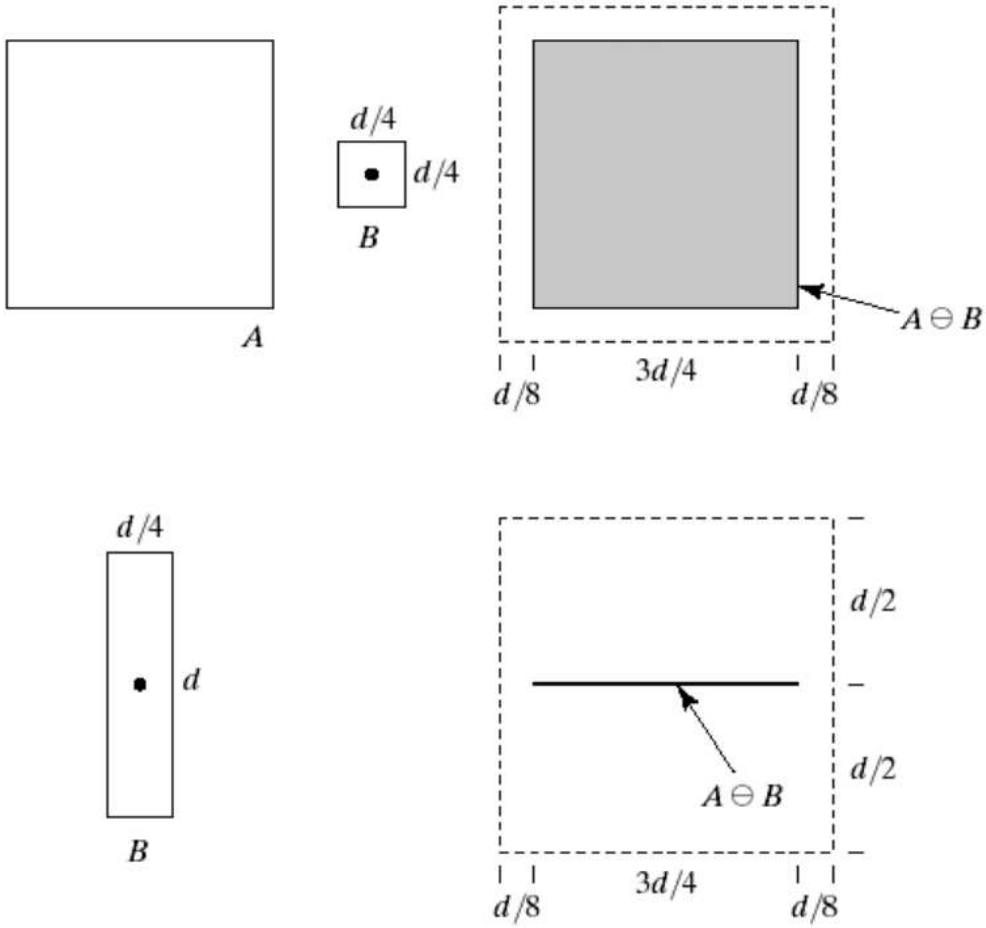

Figura 22 - Operação morfológica de erosão

Fonte – (GONZALES, 1992)

Dilatação: a operação de dilatação calcula a reflexão de B em relação a sua origem, alem de deslocar esta reflexão por *z*. A dilatação de A por B é o conjunto de todos os deslocamentos *z*, tais que B e A sobreponham-se em pelo menos um elemento não nulo.

$$A \oplus B = \{z \epsilon E | (B^s)z \cap A \neq \emptyset\} \quad (2)$$

A Figura 23 evidencia a operação de dilatação.

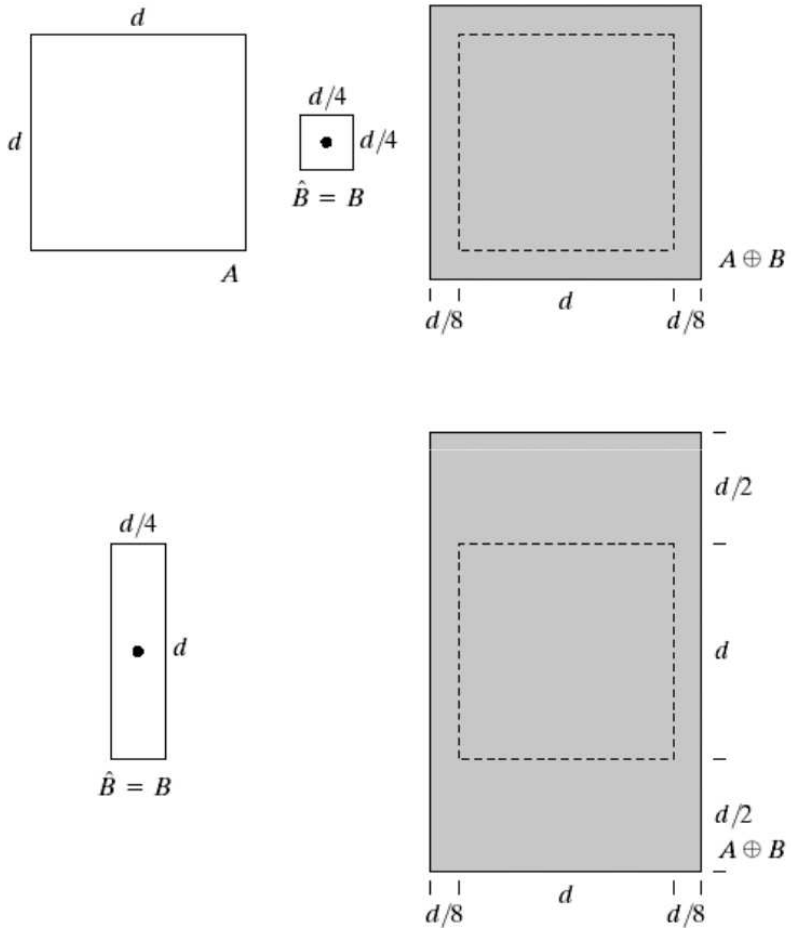

Figura 23 - Operação morfológica de dilatação

Fonte – (GONZALES, 1992)

A operação de fechamento suaviza o contorno da imagem, elimina pequenos buracos e preenche fendas em um contorno. O fechamento consiste da dilatação de A por B seguida da erosão do resultado por B e pode ser representada pela expressão a seguir:

$$A \cdot B = (A \oplus B) \ominus B \qquad (3)$$

A partir disso pode-se aplicar a operação de fechamento de buracos. A Figura 24 mostra o resultado dessa filtragem morfológica no MATLAB.

Figura 24 - Resultado final da filtragem após preenchimento de buracos

Definido esse algoritmo de filtragem, o processo de detecção do cordão de solda torna-se mais eficiente, pois foi eliminado o ruído luminoso provocado pelo ambiente.

3.2 INTERFACE DE AQUISIÇÃO DE IMAGEM POR HARDWARE

O diagrama da Figura 25 apresenta as partes básicas do projeto de PDI. A FPGA contém a parte relacionada ao processamento da imagem.

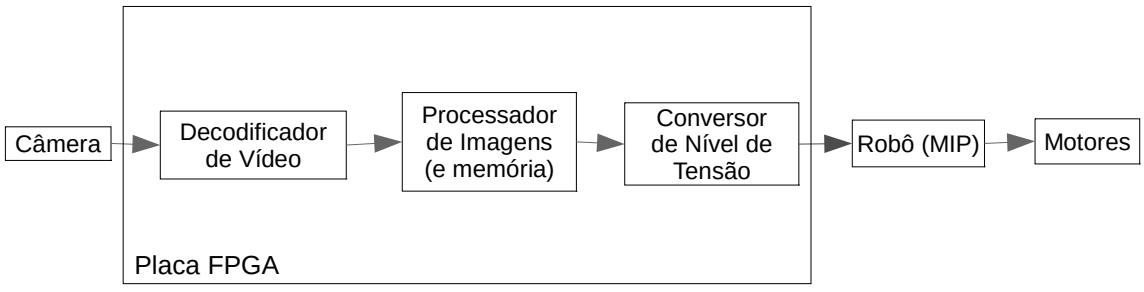

Figura 25 - Diagrama de blocos do sistema com ênfase na FPGA

A função do sistema de aquisição da imagem é capturar imagens dos feixes de laser que incidem sobre o cordão de solda. Este sistema é composto de uma

câmera analógica SONY (NS-C3901N), que é apresentada na Figura 11. A câmera possui um sensor do tipo CCD e uma lente. Este dispositivo gera um sinal de vídeo em cores no padrão analógico NTSC e fornece quadros com uma resolução de 640x480 pixels.

A lente original possui uma distancia focal de 3,6mm, o que corresponde a um ângulo de abertura de 68°, para este tipo de lente. Porém, esta lente é própria para câmeras de segurança, que têm o objetivo de filmar ambientes amplos. Para aproximar a imagem do cordão adquiriu-se outra lente, esta de 8,0mm, que fornece um ângulo de abertura de 35°. Com essa abertura é possível capturar, com nitidez, a imagem levando-se em conta a altura na qual a câmera foi posicionada em relação à chapa com o cordão de solda, que é de aproximadamente 30 cm.

Para compor a interface de captura de quadros com a imagem do cordão de solda, foi utilizado um código de aquisição da imagem. O projeto utiliza blocos de hardware, escritos em *Verilog* e disponíveis em bibliotecas da *Altera*, para captar o vídeo por meio de uma câmera analógica. O circuito converte o sinal de entrada para o padrão RGB e o envia para o monitor pela interface VGA.

A interface VGA não é um dos requisitos do projeto, porém foi utilizada, durante o desenvolvimento do projeto, para a visualização da imagem captada, facilitando assim o processo de *debug*.

O código implementado utiliza um decodificador de vídeo analógico, o ADV7181. Este chip, que está disponível e *roteado* na própria placa DE2, é um decodificador de vídeo que detecta e converte automaticamente um sinal analógico de televisão de banda base padrão (NTSC, PAL ou SCAM) em um sinal no formato de vídeo digital ITU-R BT 656 no padrão de cores YCbCr, o qual é posteriormente convertido para RGB pelo circuito em *Verilog*. A saída VGA tem os níveis de tensão ajustados por outro chip, o conversor de vídeo digital para analógico ADV7123, também disponível no kit de desenvolvimento utilizado no projeto.

O código de processamento de imagem desenvolvido em *VHDL* neste projeto foi incorporado como um componente extra ao projeto em *Verilog*.

A comunicação do sinal de erro para o robô foi feita com a utilização de protocolo de comunicação serial. Para tanto foi codificada também uma UART que utiliza a saída RS232, também roteada na placa DE2, e incluída no projeto como um componente extra.

No inicio do desenvolvimento dó código de aquisição, implementado em hardware, foram estudadas as possibilidades e ferramentas disponíveis pelo software utilizado (Quartus II) para desenvolver o programa. Uma das ferramentas consideradas foi o diagrama de blocos. O diagrama de blocos possibilita a construção de um esquemático de todo o sistema ou parte dele. No esquemático pode-se inserir os blocos de código e depois apenas conectá-los, gerando assim o sistema final de acordo com o algoritmo.

Num primeiro instante, o diagrama de blocos pareceu otimizar a codificação do algoritmo, pois, por exemplo, pode-se alterar o código rapidamente, apenas alterando as ligações entre os blocos. Porém à medida que a estrutura do código foi aumentando, começaram a surgir alguns problemas. O diagrama se tornou muito extenso, dificultando a análise do sistema para alterações e solução de problemas. Além disso, o código gerado pelo software, a partir do esquemático acaba sendo, geralmente, maior do que o necessário. Comparando com um código gerado posteriormente pela equipe, verificou-se isso de fato. Um código maior que o efetivamente necessário pode utilizar mais hardware, desperdiçando-o. Então esta ferramenta acaba não otimizando o uso do hardware. Portanto essa ferramenta foi descartada pela equipe, que a utilizou apenas para auxiliar no desenvolvimento do código.

3.3 ALGORITMO DE PROCESSAMENTO DE IMAGEM IMPLEMENTADO EM HARDWARE

Para o desenvolvimento do projeto foi adotada uma metodologia de trabalho composta por quatro etapas. Sendo a primeira a definição de requisitos, após esta há a etapa de análise e projeto, a próxima fase é a de implementação e finalmente a parte de testes. Sendo que, ao se identificar algum problema em alguma dessas fases, pode ser necessário voltar a etapas anteriores e rever o projeto ou os requisitos, por exemplo.

Na parte de implementação em *VHDL* do algoritmo de filtragem, desenvolvido inicialmente em MATLAB, foram identificados problemas que tornariam a codificação extremamente custosa do ponto de vista de *hardware*. O processo de fechamento morfológico torna necessário o armazenamento duplo da imagem, pois

nesta operação a imagem é percorrida por um *iterador* e alterada, assim deve-se duplicar a imagem para que a operação ocorra sobre a imagem original e não sobre imagem já alterada. Portanto foi necessário rever o algoritmo de processamento de imagem. Em um projeto como este, que é desenvolvido em ambiente de laboratório de pesquisa, há a disponibilidade de um kit de desenvolvimento com memória suficiente para guardar vários quadros, porém para um produto industrial são desenvolvidos chips dedicados sendo que o lucro oferecido ao produtor é tão maior quanto mais barato for o seu custo de produção, já que este último aumenta substancialmente com a área de silício utilizada para a confecção do chip.

Desta vez levando em consideração o padrão geométrico formado pelo laser sobre o cordão de solda, o feixe incidente sobre a chapa metálica forma uma linha reta que é interrompida por um calombo formado pelo cordão de solda, conforme ilustra a Figura 18. Para identificar e tratar tal deformação, foi desenvolvido então um algoritmo de processamento de imagem baseado em segmentação de imagem. Em visão computacional, segmentação se refere ao processo de dividir uma imagem digital em múltiplas regiões (conjunto de pixels) ou objetos, com o objetivo de simplificar e/ou mudar a representação de uma imagem para facilitar a sua análise. Segmentação de imagens é tipicamente usada para localizar objetos e formas (linhas, curvas, etc) em imagens.

O resultado da segmentação de imagens é um conjunto de regiões, objetos ou um conjunto de contornos extraídos da imagem. Como resultado, cada um dos pixels em uma mesma região é similar com referência a alguma característica ou propriedade computacional, tais como cor, intensidade, textura ou continuidade. Regiões adjacentes devem possuir diferenças significativas com respeito à mesma característica (GONZALES, 1992).

O algoritmo de segmentação só pôde ser desenvolvido com sucesso devido às características construtivas do sistema. Foi construída uma estrutura concisa que permite que a câmera focalize o cordão de solda em uma perspectiva de visão superior. Os lasers foram fixados mais próximos à base dessa estrutura e com ângulos de incidência de valores entre 30° e 50°. O *design* mostrado na Figura 26 é a segunda versão desta estrutura. A primeira versão trazia a câmera e os lasers fixos no mesmo eixo, sendo que este era preso ao corpo do robô no mesmo lugar onde está a base da estrutura atual. Esta primeira versão foi descartada devido ao

fato de a angulação relativa entre os lasers e a câmera não ser suficiente para que a deformação causada no feixe laser pelo cordão de solda fosse diferenciada do resto da linha luminosa. A protuberância era praticamente imperceptível na imagem adquirida. Outro problema encontrado foi que a reduzida distância entre a câmera e o cordão de solda diminuía o campo de visão, resultando em uma imagem pouco nítida, além de limitar o raio de giro do robô.

Figura 26 - Estrutura fixa da câmera e dos lasers

O algoritmo de segmentação só pôde ser desenvolvido com sucesso devido às características construtivas do sistema. Com a câmera e os *lasers* fixados ao robô a distância relativa entre os *lasers* e a câmera é também fixa, permitindo assim analisar apenas certas linhas da figura, ou seja, aquelas da incidência dos feixes sobre a chapa metálica, excluindo a parte relativa à deformação formada pela reflexão do laser sobre o cordão de solda. Define-se então estas linhas como a região ou segmento da imagem a ser analisada.

Com esta abordagem por segmentação foi possível reduzir a memória utilizada. A quantidade de bytes armazenada por quadro já tinha sido reduzida em 30 vezes devido à binarização das imagens por *thresholding* de cores e é possível

reduzir ainda mais o tamanho da memória necessária analisando apenas as regiões onde se encontram as reflexões dos feixes laser sobre a chapa do tanque.

A memória utilizada na implementação deste projeto utiliza os blocos dedicados a memória que existem dentro da própria FPGA. A Cyclone II, que está incluída na placa DE2, fornece recursos de memória dedicados chamados de blocos M4K. Cada bloco M4K contém 4096 bits de memória, que podem ser configurados para implementar memórias de vários tamanhos. Um termo comum usado para especificar o tamanho de uma memória é o seu comprimento em palavras e a largura em bits. Alguns proporções suportadas pelo bloco M4K são 4K x 1, 2K x 2, 1K x 4 e 512 x 8. Nesta implementação foi utilizado o modo 256 x 16 e foi também criado um controlador de acesso à memória que permite ao circuito de processamento um acesso transparente à memória, sendo que este pode acessar bits específicos sem ter que levar em conta a posição do dado desejado no interior de uma palavra dentro da memória.

A abordagem mais utilizada para o projeto de um sistema de *hardware* é a confecção de uma máquina de estados cobrindo toda a lógica do circuito (PEDRONI, 2010). Após um estudo mais aprofundado do problema foram elaboradas algumas máquinas de estado. O *design* final da máquina de estados implementada encontra-se abaixo.

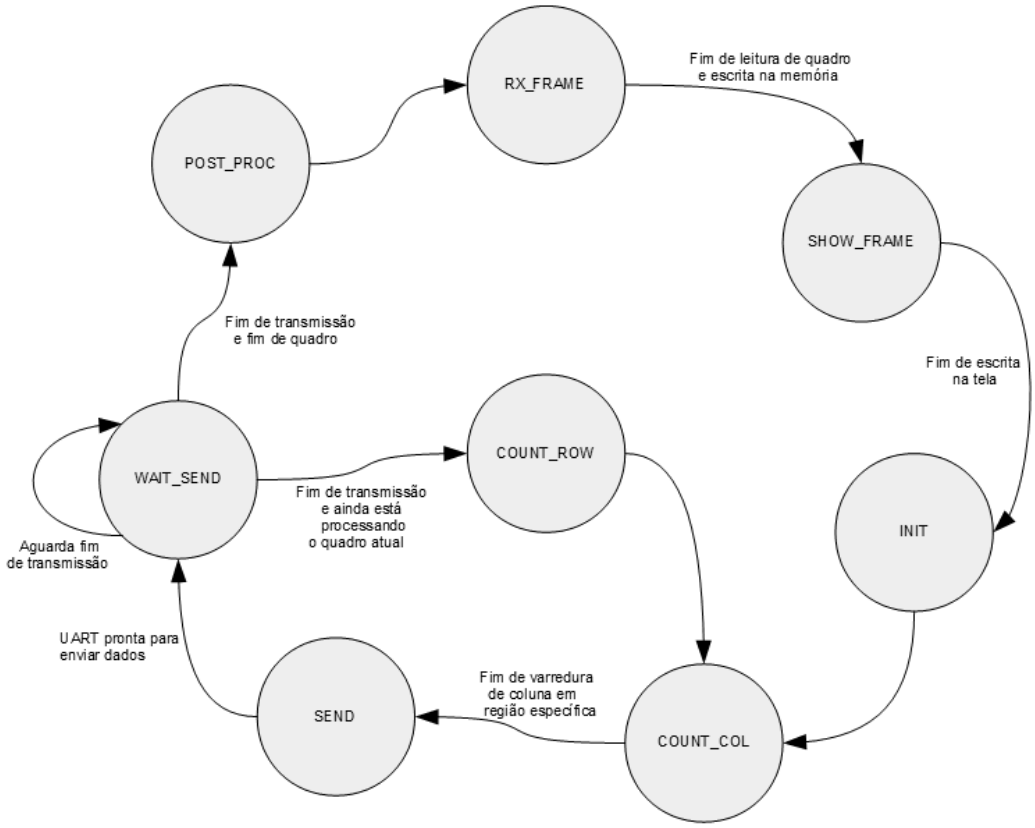

Figura 27 – Máquina de estados do circuito de PDI

Esta máquina de estados apresenta oito estados, conforme apresentado na Figura 27, e implementa o algoritmo de processamento de imagem por segmentação. Dentre os estados há um estado para auxiliar no processo de *debug*.

Estado	**Funcionamento**
RX_FRAME	Faz escrita do quadro na memória RAM.
SHOW_FRAME	Lê o quadro da memória e o imprime na tela de um monitor VGA.
INIT	Inicializa as variáveis para início do processo de varredura da imagem.
COUNT_COL	Atualiza a coluna da imagem que está sendo analisada.
COUNT_ROW	Atualiza a linha da imagem que está sendo analisada
SEND	Envia os dados para a UART.
WAIT_SEND	Espera até que a UART termine de transmitir os dados para devolver o controle do processo a outros estados.
POST	Só devolve controle do processo no início da recepção de um novo quadro vindo da câmera.

Quadro 4 - Descrição funcional de cada estado da máquina

O algoritmo de segmentação trabalha sobre a imagem sem a parte de preenchimento de buracos, ou seja, apenas com o contorno da imagem. São então analisadas as colunas da região especificada no processo de segmentação da imagem, percorre-se coluna por coluna em busca de algum *pixel* de *foreground*, caso seja encontrado esta coluna é considerada como vermelho. Neste processo reduz-se uma região de *pixels* a uma única linha, esta é enviada ao robô para que seja feito o controle dos motores para ajustar a sua trajetória em relação ao cordão de solda.

Como já citado anteriormente, uma das questões mais delicadas em projetos de hardware é o número de componentes utilizados, pois isto influencia diretamente no preço final do chip produzido. O projeto utilizou apenas 2.000 de um total 33.216 elementos lógicos da *Cyclone II*, o que representam apenas 6% de utilização da FPGA. Sendo que deste total o número de *flip-flops* é de 1.100. Estes *flip-flops* são utilizados como registradores para guardar valores necessários em futuros eventos de *clock*. Este total de elementos lógicos contempla todo o circuito, incluindo a interface de recepção e conversão do sinal de vídeo digital entregue pelo conversor ADV7181, o circuito de processamento de imagem e a UART, bem como o circuito que controla a saída VGA, que faz interface com o conversor digital-analógico ADV7123.

O circuito codificado em VHDL utiliza um sinal de *clock* com frequência de 27 MHz. Este valor foi escolhido, pois é um valor compatível com a taxa de amostragem da câmera utilizada e também devido ao fato de haver um cristal de 27 MHz disponível na placa da FPGA. O número resultante de *flip-flops* calculado pelo compilador *Quartus II* seria maior caso, na implementação, não fossem usados sinais de *clock* com frequências disponíveis na placa DE2. Isto geraria um número maior de registradores, utilizados para implementar contadores, o que aumentaria a área de silício utilizada para a fabricação de um chip dedicado e consequentemente aumentaria o preço de produção do projeto. Outro fator que diminuiu a quantidade de flip-flops utilizados no circuito foi a construção da memória, utilizada para guardar os *pixels* da imagem, com blocos dedicados que existem dentro da própria FPGA. Chamados de blocos M4K, cada um destes componentes contém 4096 bits de memória e, como especificado na parte de desenvolvimento, podem ser configurados em diversos modos de acesso. Caso não fosse especificado no código

que a memória deveria ser composta por blocos M4K, esta seria fabricada com a utilização de *flip-flops*. Como os blocos M4K são endereçados por palavra, e não bit a bit, foi também criado um controlador de acesso à memória que permite ao circuito de processamento um acesso transparente à memória.

Com a abordagem de processamento de imagem por segmentação foi possível reduzir a memória utilizada. Ao invés de 30 *bits* por *pixels*, 10 para cada canal RGB, é armazenado apenas 1 bit por pixel, devido à binarização das imagens por *thresholding* de cores. Assim uma imagem de 640x480 que ocuparia mais de 1,15Mbytes de memória em seu formato original, passa a ocupar apenas 38,4 kbytes. Analisando apenas as regiões onde se encontram as reflexões dos feixes laser sobre a chapa do tanque é possível reduzir a memória necessária para armazenar as informações relevantes de um quadro de imagem a 6,4 kbytes. Isto torna o projeto extremamente barato do ponto de vista de hardware. Logo nos primeiros testes foi constatado que é possível sintetizar esta quantidade de memória dentro do próprio chip da FPGA, sendo assim no caso da síntese de um chip dedicado com base no código de hardware desenvolvido haveria uma diferença muito grande entre o preço do chip otimizado e o preço do outro chip com abundância de memória. Sendo este chip feito com uma FPGA haveria ainda um ganho no tempo de acesso à memória, que por ser interna ao chip consegue ser mais rápido que o acesso a uma RAM externa.

3.4 CONTROLE DOS MOTORES

Para facilitar o estudo da situação de controle do robô em relação a uma posição determinada, como é o caso do controle da movimentação do robô ao longo do cordão de solda, foi adicionado à estrutura do robô o sensor de distâncias por ultra-som SRF02 da Devantec. A partir da leitura do sensor o robô deveria seguir uma trajetória paralela a uma parede e distante desta de 1m. Esta abordagem foi utilizada para facilitar o estudo da estratégia de controle do robô, principalmente na determinação de valores de ganho em controladores do tipo PID, enquanto a interface de aquisição de imagem ainda estava em desenvolvimento.

Configurada a conexão entre o sensor e a CPU, o desenvolvimento do *software* foi realizado na plataforma Eclipse, utilizando o mesmo programa de

acionamento e controle dos motores em C++. Assim, foi criada uma função para recepção dos dados vindos do sensor e para a transmissão de comandos de iniciar ou terminar as medidas de distância.

O objetivo do programa de teste é fazer com que o robô siga uma trajetória paralela a uma parede através do controle das velocidades v_1 e v_2. Estas são as velocidades definidas para os motores, cada uma para o motor de um dos lados do robô. Para isso, foi fixado um único sensor no centro da haste lateral do robô, conforme ilustrado na Figura 28.

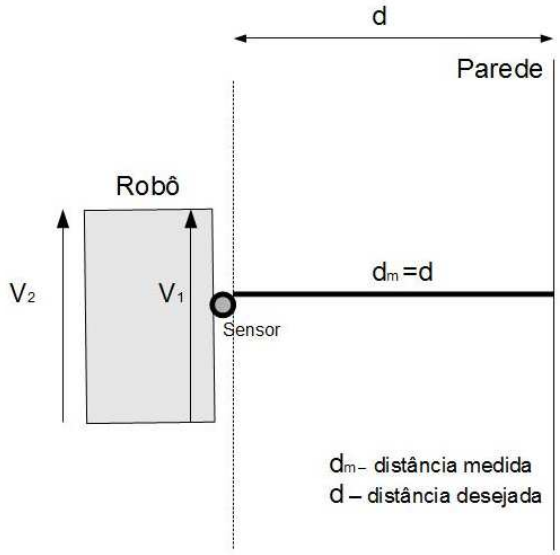

Figura 28 - Robô com o sensor a uma distância d da parede

Porém, ao realizar o controle simples do movimento do robô, o sensor capta a distância imediatamente à sua frente, ou seja, a uma angulação de 0°, considerando um eixo perpendicular ao motor. Sendo assim, ao girar o robô para atingir a distância d desejada, a distância d_r real é diferente da distância d_m medida, formando um ângulo θ entre essas duas medidas.

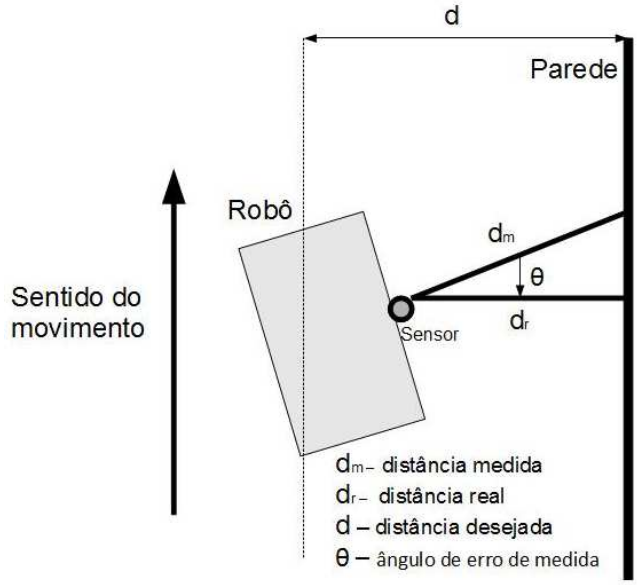

Figura 29 - Diferença entre a distância real e a distância medida

Como é disponibilizado apenas um sensor para receber informações de distância, não é possível realizar um controle eficiente, considerando apenas a angulação das retas dr e dm mostradas na Figura 29. Sendo assim, foi desenvolvido um controle proporcional (P), que utiliza o erro, definido como a diferença entre a distância d desejada e a distância d_m medida (equação erro), para realizar a correção entre as velocidades do dois motores, definidas pela equações (7) e (8).

$$e = d - d_m \quad (4)$$

$$e_{máx} = d_m - 20 \quad (5)$$

$$e_{mín} = d_m - 200 \quad (6)$$

$$V_1 = \frac{V_{máx} - V_{mín}}{e_{máx} - e_{mín}}\, e + \frac{V_{máx}}{2} \quad (7)$$

$$V_2 = -\left(\frac{V_{máx} - V_{mín}}{e_{máx} - e_{mín}}\right) e + \frac{V_{máx}}{2} \quad (8)$$

A partir da aplicação dessas equações foi possível realizar o controle da distância entre a parede e o robô. Porém, o sobre-sinal do sistema permaneceu alto, fazendo com que o tempo de estabilização também fosse alto. Perante as limitações desse sistema, como a disponibilidade de apenas um sensor para realizar as medidas, os resultados foram positivos. Isto porque, mesmo com a demora para a estabilização do robô, foi possível controlar o robô muito próximo da distância desejada.

3.5 INTEGRAÇÃO ENTRE O PDI E O MÓDULO DE CONTROLE DO ROBÔ

A partir do processamento de imagem desenvolvido no kit da Altera é possível especificar a região que contém o cordão de solda, possibilitando o controle da trajetória do robô. Para isso, é necessário comunicar o robô com a placa DE2, podendo ser realizado via Ethernet, serial, USB ou PS2, que são os periféricos disponíveis nos dois componentes. Como a equipe já havia implementado o recebimento de dados pela entrada serial no programa de controle dos motores do robô, optou-se por utilizar a porta serial para realizar esta comunicação.

Para a configuração da interface serial na FPGA foi utilizada a própria saída RS232 do kit da Altera, realizando a codificação em VHDL, que envia 10 bits (1 start bit, 8 bits de informação e 1 *stop* bit) à uma taxa de 115200 bps. No lado do robô, foi utilizada a entrada USB da CPU, fazendo-se necessário um conversor USB/serial. A recepção dos dados pelo programa de controle foi realizada utilizando a biblioteca *termios*, configurada para receber 1 *start* bit, 8 bits de informação e 1 *stop* bit à uma taxa de 115200 bps.

Para o programa que transmite os dados resultantes do PDI, o algoritmo foi desenvolvido basicamente para montar uma mensagem de 10 bits, com um contador específico para um ciclo de clock e para uma taxa de transmissão, fixados em 50 MHz e 115200 bps respectivamente (Figura 30). Assim, ao receber uma mensagem do programa de controle para iniciar a transmissão, esse programa envia os dados referentes às duas linhas de laser. Essa troca de mensagens ocorre durante todo o momento em que o robô encontra-se sobre o cordão de solda.

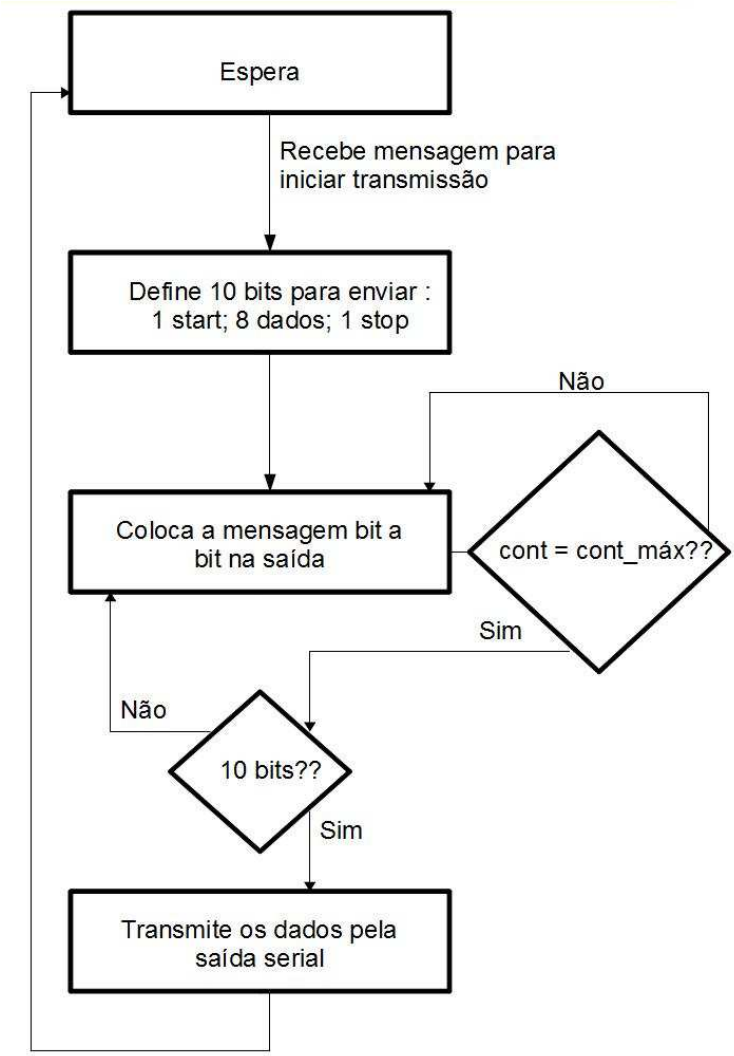

Figura 30 - Fluxograma da transmissão de dados pela FPGA

Os dados transmitidos foram padronizados como sendo a representação de uma das colunas de uma linha específica, da seguinte maneira: caso a coluna contenha um ou mais bits 1, são transmitido 8 bits 1 (representando a trajetória sem o cordão de solda). Caso contrário, se todos os bits forem 0, será transmitido 8 bits 0 (representando o cordão de solda). Com essas informações, o programa de controle monta a linha do laser, sendo o cordão de solda representado pela maior concentração de bits 0s, conforme mostra a Figura 31.

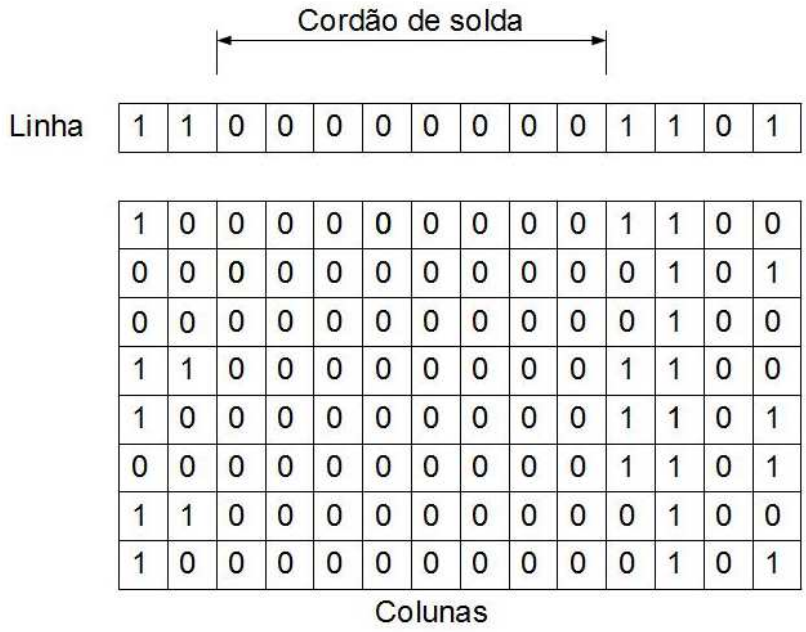

Figura 31 - Padrão para a transmissão de dados e reconstrução do feixe laser

Seguindo essa lógica, as duas linhas de laser, com o cordão de solda identificado, são remontadas no programa de controle, especificando quais são as duas coordenadas (uma em cada linha) que contém o cordão de solda. Inicialmente foi considerado que o ângulo formado por essas duas coordenadas representava o erro da trajetória do robô Figura 32, sendo utilizado para realizar o controle proporcional dos motores, ou seja, quanto maior o erro maior a velocidade de correção da trajetória.

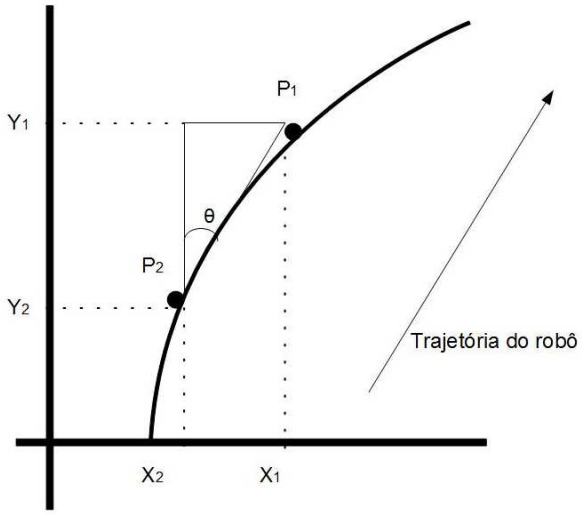

Figura 32 - Representação do cálculo do erro da trajetória do robô

As equações V1 e V2 foram encontradas a partir da definição de que a soma das duas velocidades deve ser Vmáx. Além disso, os erros máximo e mínimo foram considerados +90° e -90° respectivamente, como most ra a Figura 33.

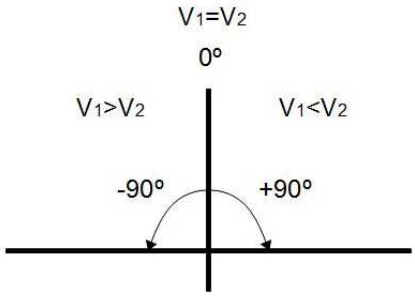

Figura 33 - Representação da definição do ângulo entre os pontos P1 e P2

A equação para o controle proporcional parte da relação entre o erro angular (θ), uma constante de proporcionalidade (ganho K) e as variáveis envolvidas (Vmáx, V1 e V2). Estas relações foram definidas a partir da Figura 34, que define a mesma velocidade para os dois motores em caso de erro zero. Assim, seguindo as Equações (9), (10) e (11), foram aplicadas nos motores velocidades proporcionais ao deslocamento do cordão de solda.

$$\theta = \text{atg}\left(\frac{X_1 - X_2}{Y_1 - Y_2}\right) \quad \textbf{(9)}$$

$$V_1 = -\left(\frac{K * Vmáx}{emáx - emín} * \theta\right) + 0.5 * Vmáx \quad \textbf{(10)}$$

$$V_2 = \left(\frac{K * Vmáx}{emáx - emín} * \theta\right) + 0.5 * Vmáx \quad \textbf{(11)}$$

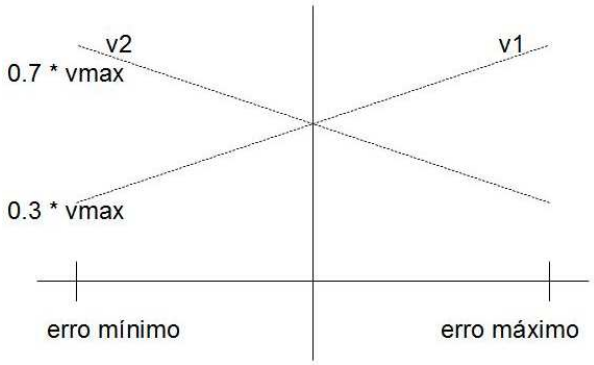

Figura 34 - Esquema utilizado para a definição das equações de controle

Porém, ao aplicar estas equações e realizar alguns testes, verificou-se que o robô não tinha o comportamento esperado, ou seja, o robô não foi capaz de seguir o cordão de solda, que foi representado por uma corda. Ao realizar uma nova análise e um estudo do comportamento durante os testes, observou-se que, assumindo o ângulo como o erro, o robô não mantinha sua trajetória alinhada ao cordão de solda, não atendendo aos requisitos desta proposta.

Para solucionar o problema apresentado acima foram levantadas algumas possibilidades para o controle da trajetória do robô. A primeira delas foi alterar o parâmetro de erro, que antes era o ângulo, para a diferença entre a média das duas coordenadas do cordão de solda e o ponto central da trajetória do robô (Figura 35). A segunda possível solução foi utilizar tanto o ângulo quanto a diferença descrita como parâmetros para definir o erro. Como a segunda solução resulta em uma equação mais complexa e difícil de implementar no programa que realiza o controle dos motores, optou-se por utilizar a primeira solução.

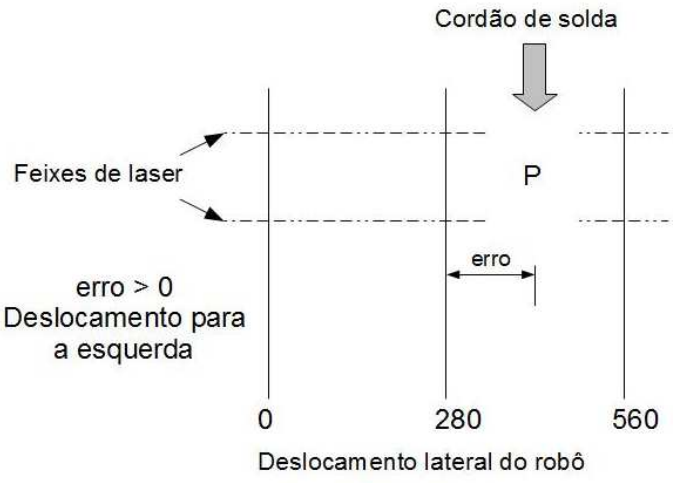

Figura 35 - Representação do novo erro

Para calcular a diferença das posições descritas acima, definiu-se como unidade de medida o quadro de pixels da câmera que realiza a aquisição da imagem. Assim, como mostra a Figura 35, com o erro positivo o movimento será para a esquerda, negativo será para a direita e com o erro zero o robô terá trajetória retilínea. Para realizar o controle proporcional o erro foi definido conforme a equação (12), e, para definir as equações de controle, foi realizada a mudança de coordenadas permitindo o cálculo dos erros máximo e mínimo (Figura 36).

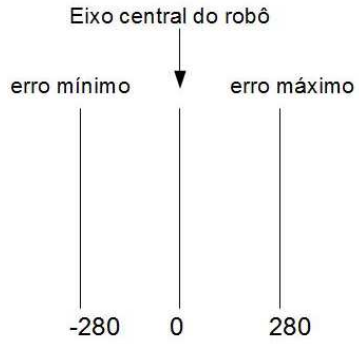

Figura 36 - Esquema após a mudança de coordenadas

$$erro = P - 280 \quad \textbf{(12)}$$

Ao serem realizados novos testes, o robô manteve-se sobre o cordão de solda em alguns momentos, mas em outros apresentou um desalinhamento e, não tendo mais a visão do cordão, o robô perdeu os parâmetros de entrada para realizar a trajetória da maneira desejada. A partir do levantamento de amostras dos valores de entrada e de saída do módulo de controle foram encontrados *glitches* que interferiam consideravelmente na definição da posição do cordão de solda.

A solução encontrada para reduzir este problema foi aplicar um filtro de médias nos vetores de dados recebidos, sendo que cada elemento da linha de laser foi remontada a partir de outros três elementos. A partir dessa última alteração, os vetores, que representam as linhas de laser, puderam ser definidos de tal maneira que as coordenadas da posição do cordão de solda ficaram muito próximas dos valores reais, tornando possível definir a trajetória corretamente.

Em seguida foram realizados novos testes a partir dos quais observou-se que o robô está apto a realizar tanto uma trajetória retilínea quanto curvilínea sobre

um cordão de solda. Além disso, o robô mantém o cordão sob o eixo central do robô, permitindo a inspeção da solda.

3.6 ARQUITETURA FINAL DO PROJETO

Ao final da etapa de desenvolvimento, após algumas alterações realizadas no projeto original, o robô chegou ao resultado esperado, sendo capaz de seguir um cordão de solda a partir de uma visão computacional. Apesar das modificações implementadas a metodologia de desenvolvimento definida em projeto foi seguida, visto que o PDI foi desenvolvido em *hardware* a partir de uma FPGA e seus periféricos, a visão computacional foi composta por esta interface em conjunto com uma câmera e dois lasers. O algoritmo de processamento de imagem utiliza o processo de segmentação com o objetivo de diminuir a memória utilizada, otimizando assim o *hardware* empregado. O controle do robô foi desenvolvido em um programa em linguagem C++, com o auxílio da IDE Eclipse, tendo como dados de entrada os resultados provenientes do processamento digital de imagem.

4 TESTES E ANÁLISE DE RESULTADOS

Ao término do desenvolvimento do projeto, foram realizados testes em laboratório para analisar o comportamento do robô a partir da definição de diferentes percursos, podendo ser retilíneos ou curvilíneos. Para realizar esta análise, um cordão de solda foi simulado a partir de uma corda disposta no chão, sendo possível verificar a resposta do robô sobre uma reta ou uma curva.

4.1 DETECÇÃO DO CORDÃO DE SOLDA

No decorrer dos testes, o módulo de detecção do cordão de solda apresentou o comportamento desejado, oferecendo ao módulo de controle de trajetória dados corretos que representam a localização do cordão de solda. Além disso, devido ao campo de visão oferecido pelo conjunto câmera-lente e à altura na qual este foi posicionado, foi possível a aquisição de imagens em uma região apropriada, que é a área localizada entre as linhas das rodas do robô, para que o mecanismo seja capaz até mesmo de realizar trajetórias curvilíneas.

A Figura 37 mostra o cordão de solda com a incidência dos feixes de laser. É possível verificar o padrão geométrico de protuberância do laser ao incidir em ângulo sobre o cordão causado pela diferença entre a altura deste e a da chapa. Este padrão permite o PDI a partir de segmentação de imagem.

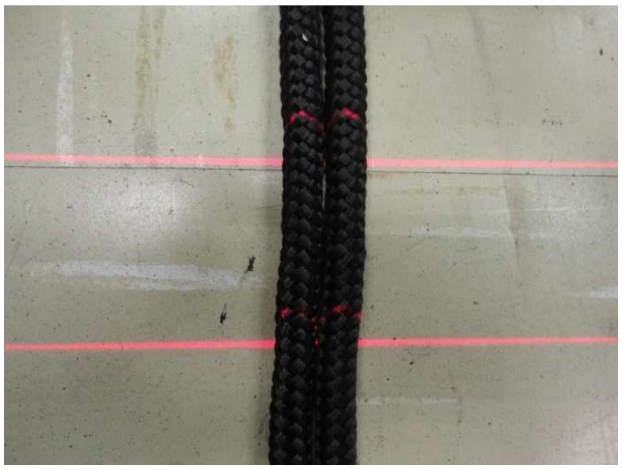

Figura 37 - Laser incidindo sobre o cordão de testes

4.2 TRAJETÓRIA RETILÍNEA

Partindo da aplicação deste robô de inspeção, os tanques de armazenamento de derivados de petróleo possuem cordões de solda dispostos em sua grande maioria de maneira retilínea, apresentando curvas nos nós de ligação entre dois cordões. Assim, um robô seguidor de solda irá realizar uma trajetória retilínea na maioria dos casos (SANTOS, CASAROTTO, BRESSAM, 2010).

Para verificar a eficácia do robô para este caso, foram realizados testes que simulam a trajetória retilínea sobre um cordão. Ao receber os dados da visão computacional, o módulo de controle do robô realiza o controle proporcional dos motores. Ao verificar que o cordão de solda está alinhado com o eixo central do robô, os motores são configurados com a mesma velocidade, resultando em uma trajetória retilínea sobre o cordão (Figura 38). Caso contrário, é realizado o controle proporcional dos motores, assumindo uma trajetória curvilínea.

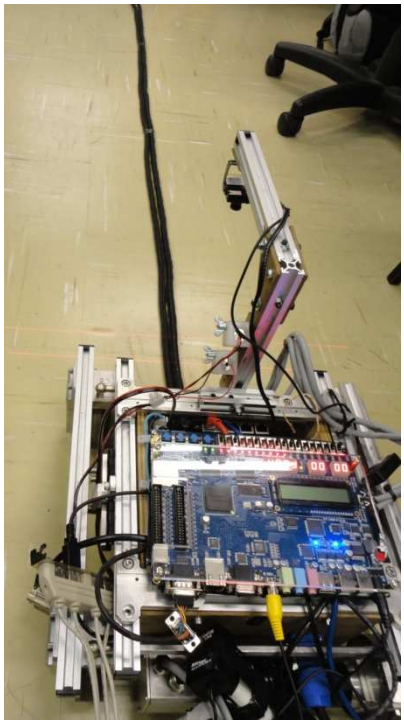

Figura 38 - Robô realizando uma trajetória retilínea

4.3 TRAJETÓRIA CURVILÍNEA

O controle do robô durante a trajetória curvilínea depende do raio desta curva. Se o raio de curvatura for menor que as dimensões do robô, não haverá ângulo de giro para o mecanismo, resultando na perda do cordão de solda pela visão computacional do robô e no mau funcionamento do sistema. Porém, ao respeitar este raio mínimo, os módulos de aquisição de imagem e de controle respondem satisfatoriamente, de maneira que os motores são configurados para que o robô siga a curva do cordão de solda suavemente, conforme ilustra a Figura 39.

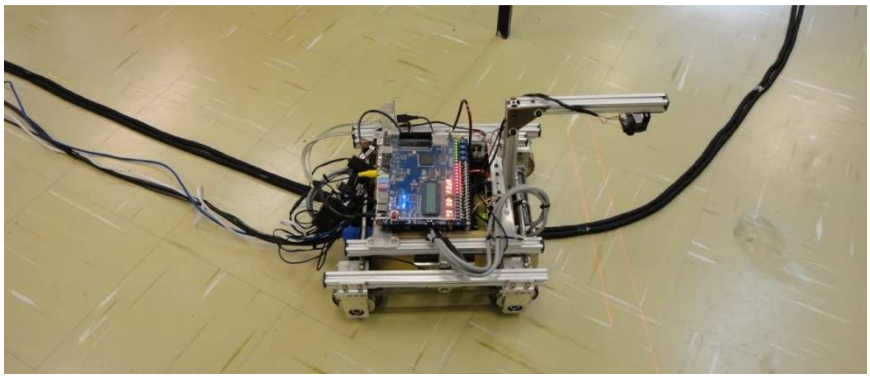

Figura 39 - Robô realizando uma trajetória curvilínea

Assim, visto que as curvas em cordões de solda em tanques de armazenamento não são comuns, esta limitação de trajetória curvilínea não compromete o bom funcionamento do robô e o controle proporcional definido pela Figura 34, que é representada pelas equações (9), (10) e (11), é suficiente para garantir a trajetória esperada do sistema, conforme registrado em http://youtu.be/Vp4-p07JSeA.

5 GESTÃO DO PROJETO

Para o sucesso de um projeto, além das especificações técnicas para o desenvolvimento, é necessário formular um cronograma, baseando-se nos prazos de entrega dos componentes e no ritmo de trabalho dos integrantes da equipe, organizando as atividades do projeto em função do tempo de execução de cada uma delas. Além disso, é preciso considerar a possibilidade de uma tarefa não ser executada, através da realização de uma análise dos riscos, oferecendo alternativas aos problemas.

5.1 CRONOGRAMA

Após a definição do projeto foi elaborado um cronograma para auxiliar seu desenvolvimento, facilitando a visualização das atividades a serem realizadas com seus respectivos prazos (Quadro 5).

	Cronograma Detalhado	2013											
	Atividade	fev		mar		abr		mai		jun		jul	
1.1	Testes com os lasers			■									
1.2	Testes de aquisição de dados pela câmera			■									
1.3	Testes com a Altera DE2	■	■										
2.1	Especificação da transferência de dados entre câmera e FPGA			■	■								
2.2	Definição dos algoritmos de Processamento de Imagem		■	■									
3.1	Simulação no MATLAB do processamento da imagem adquirida			■	■								
4.1	Implementação do sistema em VHDL					■	■	■	■	■	■	■	
4.2	Integração da FPGA com a câmera e os laseres							■	■				
5.1	Integração do módulo de PDI com o controle de movimento								■	■			
6.1	Testes da funcionalidade de seguimento do cordão de solda								■	■	■		
7.1	Análise de mercado/riscos										■	■	
7.2	Documentação do desenvolvimento do projeto	■	■	■	■	■	■	■	■	■	■	■	
8.1	Apresentação do trabalho												■

Quadro 5 - Cronograma detalhado por atividade previsto para o projeto

No decorrer do desenvolvimento do projeto, mesmo a equipe sendo atenta ao cronograma, ocorreram alguns atrasos. Dentre esses, cita-se o aumento do tempo dos testes com o kit da Altera DE2, pois, para realizar a especificação da

transferência de dados entre a câmera e o kit, foi necessário um estudo mais aprofundado das características tanto da câmera quanto do processo de aquisição da imagem pela *FPGA*. Consequentemente, a especificação da transferência de dados entre a câmera e a *FPGA* e as outras atividades definidas para começar em seguida foram afetadas por esse atraso em, pelo menos, 2 semanas. Os atrasos ocorreram, principalmente, devido às dificuldades encontradas ao realizar o interfaceamento entre a câmera, a *FPGA* e o *driver* do motor.

5.2 ANÁLISE DE CUSTO DO PROJETO

Com a definição do projeto foi feito um levantamento dos custos de hardware, listando na Tabela 2 apenas os componentes que foram especificados no início do desenvolvimento do robô, como motores e *drivers*, componentes para montar os cabos dos motores e conectá-los ao *driver*, a placa de desenvolvimento da Altera, a câmera e acessórios e os *lasers*.

Atividade	**Custo (R$) – Estimado**	**Custo (R$) – Efetivo**
Compra da câmera e acessórios	100,00	130,00
Compra da placa da Altera DE2	700,00	678,43
Compra dos 3 laseres	60,00	27,00
2 kits de motor e driver	8.568,00	8.568,00
Componentes para ligação motores	200,00	150,00
Total	9.628,00	9.553,43

Quadro 6 - Custo em reais dos componentes eletrônicos

Além disso, o Quadro 6 também mostra o custo efetivo dos componentes comprados, verificando uma pequena economia de aproximadamente R$25,00. Percebe-se que os componentes de maior valor foram estimados corretamente, visto que os valores foram baseados em orçamentos realizados diretamente com o fornecedor. Assim, a economia observada foi devido aos pequenos componentes, como os necessários para a ligação dos motores e os *lasers*.

Os custos envolvidos com a aquisição de software não são mensuráveis, pois, os que não são gratuitos, acompanham o *hardware* ou estavam disponíveis no laboratório LASCA da UTFPR. Por exemplo, o *software* utilizado para os testes com os motores e *drivers* (*EPOS-Studio*) foi disponibilizado com a compra desses

componentes. Além disso, o MATLAB estava disponível no laboratório e as versões utilizadas para o Eclipse e para o Quartus II são livres e estão disponíveis para *download*.

Além dos custos com *hardware* e *software* foram levantados os custos em horas de trabalho da equipe. O Quadro 7 mostra a divisão das atividades entre os integrantes do grupo e as horas de trabalho previstas e as realizadas. A partir disso é possível verificar que o aumento das horas trabalhadas ocorreu principalmente em função de problemas nos testes com a interface entre a câmera e a placa da Altera.

	Responsável	**Custo previsto**	**Custo efetivo**
1.1	Gabriel/Isabela	20 horas/pessoa	20(G)/10(I)
1.2	Gabriel	20 horas/pessoa	30 horas
1.3	Érika	30 horas	60 horas
2.1	Todos	40 horas/pessoa	40 horas/pessoa
2.2	Gabriel	40 horas	30 horas
3.1	Gabriel	40 horas	40 horas
4.1	Todos	40 horas/pessoa	50 horas/pessoa
4.2	Érika	20 horas	20 horas
5.1	Isabela	50 horas	50 horas
6.1	Isabela/Gabriel	20 horas/pessoa	20 horas/pessoa
7.1	Isabela	10 horas	15 horas
7.2	Todos	30 horas/pessoa	40 horas/pessoa
8.1	Todos	20 horas/pessoa	20 horas/pessoa

Quadro 7 - Cronograma com o custo e as atividades por integrante previsto

Analisando os dados apresentados no Quadro 8 verifica-se que as horas totais trabalhadas correspondem a mais de 12% além do previsto. Como citado acima, o motivo principal para este aumento foi a dificuldade para a familiarização com as interfaces envolvidas no projeto. Assim, a equipe poderia ter reservado mais horas no início do desenvolvimento para o estudo dos periféricos da placa da Altera e também para o estudo das possíveis soluções para a aquisição da imagem.

Membro	**Previsto**	**Efetivo**
Érika	180	230
Gabriel	270	290
Isabela	230	245
Total	700	785

Quadro 8 - Horas trabalhadas por pessoa (previstas/efetivo)

5.3 ANÁLISE DE RISCOS DO PROJETO

Ao iniciar o desenvolvimento do projeto foram levantados alguns riscos de tarefas não serem realizadas e, consequentemente, os resultados não serem alcançados. Os possíveis efeitos no projeto foram definidos, assim como uma ação a ser tomada caso o problema se concretize. Estes riscos, efeitos e ações estão organizados no Quadro 9 seguindo a legenda do Quadro 10, dispostos do maior para o menor grau, sendo este calculado através da equação (13).

$$Grau = Impacto * Probabilidade \quad \textbf{(13)}$$

Grau	Risco	Efeito	Probabilidade	Impacto	Ação
0,7	Erro ao definir os componentes de *hardware*	Falha nos módulos de PDI e de trajetória	0,7	1	Eliminar: Pesquisa e teste de componentes que atendam as especificações de projeto
0,48	O projeto pode não ser robusto o suficiente para ser aplicado em campo	Dificuldade para validar a solução de projeto desenvolvida	0,6	0,8	Reduzir: ao longo do desenvolvimento realizar testes em laboratório simulando ambientes reais
0,45	Componentes comprados podem demorar a chegar	Atraso no desenvolvimento do projeto	0,5	0,9	Eliminar: realizar o pedido de componentes com antecedência
0,25	Algoritmo de PDI não atende ao funcionamento online do robô	Falha na trajetória do robô (sem sincronia entre movimento e localização)	0,5	0,5	Reduzir: verificar o tempo de processamento de hardware e software
0,3	Erro na integração do módulo PDI com o módulo de controle de trajetória	Não será possível definir a movimentação do robô	0,3	1	Eliminar: desenvolver os módulos separadamente, verificando a compatibilidade
0,07	Petrobras cancelar o incentivo ao projeto	Dificuldade financeira para adquirir componentes	0,1	0,7	Reduzir: orçar componentes necessários

Quadro 9 - Análise de riscos do projeto

Legenda		
Grau Baixo 0 – 0,4	Grau Médio 0,4 – 0,7	Grau Alto 0,7 - 1

Quadro 10 - Legenda para a análise de riscos do projeto

No decorrer do projeto a equipe teve problemas com a definição da câmera, sendo enquadrado no risco de maior grau descrito como o erro ao definir componentes de *hardware*. A primeira câmera testada estava disponível no laboratório, mas apresentou problemas de resolução, que era incompatível com o padrão da placa da Altera. Assim, uma nova câmera foi adquirida resolvendo este problema, mas a equipe apresentou dificuldades ao realizar a codificação em *RGB* na *FPGA*. Após várias tentativas, iniciou-se a pesquisa de uma terceira câmera, que disponibilizava na saída os pixels em *RBG*. Porém, não foi encontrado um fornecedor adequado para a esta aplicação. Sendo assim, a câmera utilizada no projeto foi a com resolução correta e que necessitou ter a saída codificada para o padrão *RGB*.

Assim, o grupo continuou com dificuldades ao estudar, entender e codificar a interface entre a câmera e o kit de desenvolvimento da Altera. Este foi o fator responsável pelo maior atraso no projeto, afinal é o pré-requisito para iniciar a codificação em VHDL. Mas, mesmo com atrasos, estes problemas não impediram que o projeto fosse desenvolvido e finalizado conforme os objetivos especificados.

6 CONCLUSÃO

Para que o sistema funcionasse corretamente e também para guiar o desenvolvimento do projeto, foram definidos os seguintes objetivos:

- Adquirir feixes de *laser* paralelos incidindo sobre uma superfície plana ou curva, desde que esta última possua um grande raio, contendo um cordão de solda.
- Identificar e capturar, por meio de uma câmera de vídeo, a imagem do cordão de solda sobre o qual incidem os feixes luminosos.
- Codificar em *VHDL* um circuito para processar as imagens obtidas pela câmera e embarcar este código em um *FPGA*, visando um processamento em tempo real.
- Atuar no controle dos motores ajustando a direção de movimento do robô e fazendo com que este percorra o cordão de solda.

O desenvolvimento deste projeto foi então iniciado a partir da visão computacional. Composto por dois feixes de lasers paralelos incidindo sobre um cordão de solda e uma câmera, o sistema de visão artificial realiza a aquisição de imagem em tempo real. Após a captura da imagem, esta é organizada na memória interna da FPGA. Tendo isto armazenado, o algoritmo de PDI, desenvolvido em *hardware* e codificado em VHDL, utiliza a segmentação de imagem para filtrar os feixes de laser que incidem sobre o cordão de solda. Os dados resultantes deste módulo de processamento de imagem são enviados serialmente para o módulo de controle do robô. Ao receber os dados da DE2, o programa utiliza um filtro de médias para minimizar o erro referente à determinação do ponto do cordão de solda e, logo em seguida, realiza o controle proporcional dos motores.

Ao finalizar a implementação de todas as etapas foram realizados testes, nos quais o robô apresentou ótimos resultados. A partir de uma visão computacional o robô seguiu de maneira autônoma um cordão de solda, mantendo-o centralizado em relação ao eixo central do robô, possibilitando assim a inspeção da solda e atingindo todos os objetivos definidos no projeto.

Além do desenvolvimento técnico, a gestão do projeto estimou corretamente os custos com equipamentos, economizando menos de 1% do valor final. Porém, em relação às horas empregadas no desenvolvimento do projeto houve um aumento

de mais de 12% ao que foi previsto inicialmente. Isto aconteceu principalmente porque a equipe não definiu muitas horas para o estudo das interfaces entre os módulos de desenvolvimento, que foi um dos pontos mais críticos do desenvolvimento do trabalho que, devido a dificuldades de implementação, causou atraso na finalização do projeto.

6.1 PROJETOS FUTUROS

Após a análise dos resultados obtidos é possível verificar que existem melhorias a serem feitas para que este projeto resulte em um produto confiável do ponto de vista técnico e de segurança, mas também um produto econômica e financeiramente viável.

Ao verificar que o robô realiza sua trajetória sobre o cordão de solda de maneira autônoma, é possível acoplar um módulo de inspeção de solda ao robô. Esta fase de desenvolvimento do projeto robô de inspeção já foi iniciada no laboratório LASCA, apresentando algumas simulações de verificação de falhas em solda por meio de um *software*.

Além de novas funcionalidades, também podem ser feitas melhorias de *hardware* neste projeto, como a produção de um chip do circuito desenvolvido em *hardware* e que atualmente é sintetizado na FPGA da placa DE2, visto que o espaço de memória utilizado foi reduzido e o número de componentes foi otimizado, justamente para viabilizar esta etapa, que irá diminuir o número de componentes embarcados no robô, devido à substituição da placa DE2.

Outro problema a ser resolvido em futuras versões do sistema é o superaquecimento da CPU. Isto reduz a confiabilidade do sistema, pois ao atingir a temperatura máxima suportada pelo processador, a CPU é reiniciada. Com a solução encontrada nesta implementação há uma perda na velocidade de processamento, o que é extremamente custoso para sistemas em tempo real. Isto pode ser solucionada acoplando um sistema melhor de dissipação de calor, como um cooler ou dissipadores mais eficientes.

Além disso, para dar mais flexibilidade para as inspeções de tanques de armazenamento de derivados de petróleo, seria interessante alterar a alimentação de cordão umbilical para uma alimentação por baterias. Porém, ao realizar esta

substituição deverá ser levado em consideração o tamanho e o peso da bateria, além da potência dissipada pelos componentes, garantindo que o processo de inspeção não seja prejudicado pela falta de energia ou pelo excesso de peso do robô.

REFERÊNCIAS

ALECRIM, Emerson. **Tecnologia USB (Universal Serial Bus)**. Disponível em <http://www.infowester.com/usb.php>. Acesso em: 09 jun. 2013.

CAN IN AUTOMATION. **CAN Protocol**. Disponível em <http://www.can-cia.org/index.php?id=systemdesign-can-protocol>. Acesso em: 03 mai. 2013.

DE SOUSA, Rafael V.; INAMASU, Ricardo Y.; TORRE NETO, André. **CAN (Controller Area Network):** Um Padrão Internacional de Comunicação de Transdutores Inteligentes para Máquinas Agrícolas. Circular Técnica Embrapa, número 12, São Carlos, SP, Outubro, 2001.

DOS SANTOS JUNIOR, Eduardo F.; CASAROTTO, Matheus L.; BRESSAM, Wagner C. **Sistema de Análise e Deslocamento para Inspeção de Solda** (SADIS). 2010. 105 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná. Curso Superior de Engenharia Industrial Elétrica – ênfase Eletrônica/Telecom., Curitiba, 2010.

ESSER, Eduardo M.; LARA, Guilherme R.; ARANTES, Lucas G.; BRONDANI, William M. S**istema de Autolocalização Inteligente por Trilateração** (SALIT). 2009. 180 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná. Curso Superior de Engenharia Industrial Elétrica – ênfase Eletrônica/Telecom., Curitiba, 2009.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital image processing.** Reading, Mass.: Addison-Wesley, 1992.

GONZALEZ, Rafael C.; WOODS, Richard E.; EDDINS, Steven L. **Digital image processing using MATLAB®.** Upper Saddle River, NJ.: Prentice-Hall, 2004.

GOODWINE, Bill. **RS-232 Serial Protocol**. Disponível em <http://controls.ame.nd.edu/microcontroller/main/node24.html>. Acesso em: 06 mai. 2013.

LI Y., WANG Q., LI Y., XU D., TAN M.On-line Visual Measurement and Inspection of Weld Bead Using Structured Light**. IEEE International Instrumentation and Measurement Technology Conference**, Victoria, Canadá, 12-15 Maio, 2008.

LI, Y.; WANG, Q.; LI, Y. XU,D. e TAN M.. Measurement and Defect Detection of the Weld Bead Based on Online Vision Inspection. **IEEE Transactions on Instrumentation and Measurement**, Vo. 6, No. 7, p. 1841-1849, Julho, 2010.

MAXON MOTORS. **Online Catalog**. Disponível em <http://www.maxonmotor.com/maxon/view/catalog/>. Acesso em: 02 mai. 2013.

NATIONAL INSTRUMENTS**. Introdução à tecnologia FPGA.** Disponível em <http://www.ni.com/white-paper/6984/pt>. Acesso em 02 fev. 2013.

NET TUTORIAIS. **Comparativo: Windows x Linux**. Disponível em <http://www.nettutoriais.com.br/2013/03/comparativo-windows-x-linux.html>. Acesso em: 09 jul. 2012.

OGATA, Katsuhiko. **Engenharia de controle moderno.** 5. ed. São Paulo, SP: Pearson Education do Brasil, 2012.

PEDRONI, Ricardo Umbria. **Sistema autônomo em FPGA para captura e processamento em tempo real de imagens da pupila.** 2011. 85 f. Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná, Curitiba, 2011.

PEDRONI, Volnei A. **Circuit Design and Simulation with VHDL**. London, UK, 2nd edition, The MIT Press, 2010.

PEDRONI, Volnei A. **Eletrônica digital moderna e VHDL.** Rio de Janeiro, RJ: Elsevier, 2010.

ROVANI, Anderson. **Desenvolvimento do Protótipo de um Robô para Inspeção de Cordões de Solda em Superfícies Metálicas Verticais.** 2013. 114 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná. Curso Superior de Engenharia Mecânica, Curitiba, 2013.

SAIDONR, M.S.; DESA, H.; RUDZUAN, M.N. A differential steering control with proportional controller for an autonomous mobile robot. **Signal Processing and its Applications (CSPA) IEEE 7th International Colloquium**,p. 90-94, Março, 2011.

TANENBAUM, Andrew S. **Sistemas operacionais modernos.** 2. ed. São Paulo, SP: Prentice-Hall, 2003.

TERZIAN, Ricardo L. **Conceitos e metodologias de gestão de projeto e sua aplicação ao caso da integridade da malha dutoviária:** Gerenciamento de Projetos de Gás Natural. 2005. 0 f. Dissertação (Mestrado em Engenharia de Produção) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

VÍDEO TESTE, U.S.S. AIR-12 - Teste em Laboratório. Disponível em <http://youtu.be/Vp4-p07JSeA> Acesso em: 13 ago. 2013

WATANABE, Angélica L.; BORBA, Camila B.; OLEINIK, Daniel A. **Sistema de Detecção de Obstáculos e Obtenção de Trajetória** (SiDOOT)**.** 2010. 125 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná. Curso Superior de Engenharia Industrial Elétrica – ênfase Eletrônica/Telecom., Curitiba, 2010.

WEI, Boyu; YING, Fan; BAOQUAN, Gao. Mobile Robot Vision System Based on Linear Structured Light and DSP. **IEEE International Conference on Mechatronics and Automation**, Changchun, China, p. 1285-1290, 9-12 Agosto, 2009.