UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANA BEATRIZ PRUDÊNCIO DE ALMEIDA REBOUÇAS

APLICAÇÃO DE METODOLOGIA DE ENGENHARIA REVERSA PROPOSTA PARA DOCUMENTAÇÃO DE UM ROBÔ MÓVEL REAL

FORTALEZA

2016

ANA BEATRIZ PRUDÊNCIO DE ALMEIDA REBOUÇAS

APLICAÇÃO DE METODOLOGIA DE ENGENHARIA REVERSA PROPOSTA PARA DOCUMENTAÇÃO DE UM ROBÔ MÓVEL REAL

Monografia apresentada ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fabrício Gonzalez Nogueira.
Coorientador: Prof. Dr. Bismark Claure Torrico.

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R24a Rebouças, Ana Beatriz Prudêncio de Almeida.
Aplicação de metodologia de engenharia reversa proposta para documentação de um robô móvel real / Ana Beatriz Prudêncio de Almeida Rebouças. – 2016.
190 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2016.
Orientação: Prof. Dr. Fabrício Gonzalez Nogueira.
Coorientação: Prof. Dr. Bismark Claure Torrico.

1. Engenharia Reversa. 2. Robô Móvel. 3. Documentação. I. Título.

CDD 621.3

ANA BEATRIZ PRUDÊNCIO DE ALMEIDA REBOUÇAS

APLICAÇÃO DE METODOLOGIA DE ENGENHARIA REVERSA PROPOSTA PARA DOCUMENTAÇÃO DE UM ROBÔ MÓVEL REAL

Monografia apresentada ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Fabrício Gonzalez Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Bismark Claure Torrico (Coorientador)
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Wilkley Bezerra Correia
Universidade Federal do Ceará (UFC)

_____
Msc. Magno Prudêncio de Almeida Filho
Programa de Pós-graduação em Engenharia Elétrica (UFC)

Dedico este trabalho a minha avó Maria Lopes de Almeida (*in memorian*) por ter apoiado meus estudos desde que eu era uma criança e aos meus pais por serem minha base sempre.

AGRADECIMENTOS

A Deus, que em sua grandiosidade tem estado ao meu lado sempre, abençoando-me com grandes conquistas.

Aos meus pais, Júnior e Ida, que são meu maior exemplo de persistência e meu suporte diário, que compreenderam a minha ausência, mas se fizeram presentes na minha vida em todos os momentos.

Aos meus irmãos, Danielle e Davi, por compartilharem o entusiasmo com as minhas vitórias.

Ao tio André, pelo apoio, inspiração e incentivo diário.

Ao meu namorado, Filipe, que me encorajou e me ajudou por diversos momentos, que me serviu de exemplo e suporte. Obrigada por acreditar tão convictamente no meu êxito.

Ao meu orientador, Prof. Dr. Fabrício Nogueira, que soube reconhecer esse trabalho desde o início, empolgou-se com os frutos decorrentes e as possibilidades de prosseguimento e foi bastante compreensivo durante a execução dessa monografia. Pela orientação, apoio e confiança, meu muito obrigada.

Ao meu coorientador, Prof. Dr. Bismark Torrico, pelo incentivo, disponibilidade e apoio a mim sempre demonstrados.

Ao Thiago e ao Nadson, por terem aceitado o desafio de trabalhar com o Nanook quando estávamos nos Estados Unidos e por terem contribuído tão sabiamente para o sucesso resultante.

Ao Michael Comberiate, que proporcionou a oportunidade ímpar de trabalhar com o Nanook.

Ao Emiliano, por sempre ter sido bastante prestativo comigo e ter me ajudado junto às burocracias exigidas durante esse curso de graduação.

Ao Jeymyson por ter incentivado o meu intercâmbio acadêmico, que significou o início da jornada que derivou nesse trabalho.

À Celina, aqui representando meus colegas de curso ou universidade, que estiveram presentes diariamente nesses longos anos, que me ajudaram, me divertiram, me encorajaram nas muitas noites acordada, nas diversas vésperas de provas, nos almoços no R.U. e nos intervalos de aula. Eu jamais conseguiria chegar aqui sem a presença de vocês, amigos que fizeram parte da minha formação e que, por certo, continuarão presentes em minha vida.

À Universidade Federal do Ceará, à *Capitol Technology University* e à *New York Institute of Technology* pela estrutura que me foi proporcionada durante essa etapa da minha vida. Pelas oportunidades e atividades desenvolvidas na universidade, hoje encontro repleto em mim esse sentimento de realização e sucesso.

Aos meus professores, que souberam me repassar tão bem um conhecimento tão precioso; alguns que souberam inclusive transmitir a paixão por essa profissão escolhida por nós.

À Capes e ao CNPq por terem financiado o meu intercâmbio acadêmico nos Estados Unidos, proporcionando-me uma experiência única que fomentou meu desenvolvimento pessoal, cultural e profissional.

"*If I cannot do great things, I can do small things in a great way.*"

Martin Luther King Jr

RESUMO

Este trabalho propõe uma metodologia de engenharia reversa para realizar a documentação detalhada de um robô previamente desenvolvido. São apresentados conceitos iniciais e uma breve revisão sobre a legislação cabível a fim de embasar a proposição da metodologia empregada, que é uma adaptação das proposições de Ingle (1994) e Sharples (2010). O processo de engenharia reversa proposto foi aplicado em um caso prático, sendo analisado um robô móvel que emprega um sistema de laser (LIDAR) para obter imagens do ambiente ao seu redor. As quatro etapas básicas estabelecidas foram: reconhecimento do projeto; análise do robô; conclusões e documentação. Elas estão detalhadamente descritas ao longo deste trabalho de modo que possam ser replicadas. Finalmente, os resultados do processo de engenharia reversa no robô, o Nanook, são expostos na documentação gerada. As documentações desenvolvidas incluem tanto o sistema de hardware quanto o de software (embarcado e do programa de interface do usuário), o que torna acessível todo o conhecimento e entendimento adquirido durante a realização desse projeto. Entre os benefícios desse processo, tem-se a facilidade de compreensão, sem a necessidade de uma investigação mais detalhada que requereria a desmontagem parcial do robô e de despender um tempo considerável em algo que já foi realizado. Portanto, os investimentos e esforços podem ser focados em gerar novos resultados, contribuindo para o avanço de novos projetos. Além da documentação, foi desenvolvido um esquemático do *hardware* usando o AutoCad®. Tem-se, portanto, uma maneira mais segura para que, no futuro, as pessoas possam entender o sistema do robô e propor a implementação de mudanças no Nanook. Por fim, a contextualização da aplicação da engenharia reversa derivando resultados positivos busca contribuir para a difusão desta como um campo de estudo válido, que permite melhorias, modificações e documentação de um produto final, fomentando a inovação no ambiente acadêmico e profissional.

Palavras-chave: Engenharia Reversa. Robô Móvel. Documentação.

ABSTRACT

This work proposes a reverse engineering methodology to create a detailed documentation of an already developed mobile robot. Basic concepts and a brief legislation review are presented as the basis of the applicable methodology proposition, which is an adaptation of Ingle (1994) and Sharples' (2010) propositions. The practical case analyzed for the reverse engineering process is the analysis of a mobile robot that uses a laser system (LIDAR) to obtain environmental images. The four proposed steps were project reconnaissance, robot analysis, conclusions and documentation. Their detailed description through this work allows the methodology replication. Finally, the generated documentation exposes the results of the application of reverse engineering process in the robot, Nanook. It includes both hardware and software (embedded and user's interface program) systems. This guarantees that knowledge and understanding acquired during this project are accessible to everyone interested. The comprehension facility is one of the benefits of this process. As a physical investigation would imply in partially disassembling the robot and in spending time in something already done, the needless of these steps are an advantage. Therefore, investments and efforts can focus on generating new results, contributing to the advance of new projects. Aligned with the documentation, hardware schematics were developed using AutoCad®. Hence, there is a safe way (non-invasive) to permit people to understand the robot's system and to implement improvements in Nanook. Ultimately, the contextualization of the reverse engineering application that derives positive results seeks to contribute to its diffusion as a valid field of study, allowing a final product's improvement, modification, and documentation, which encourages innovation in the academic and professional environment.

Keywords: Reverse Engineering. Mobile Robot. Documentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATX	*Advanced Technology Extented*
CC	Corrente Contínua
CI	Circuito Integrado
CPR	Ciclos por Revolução
CPU	*Central Processing Unit* (Unidade de Processamento Central)
CREA	*Certified Reverse Engineering Analyst* (Certificação de Analista de Engenharia Reversa)
GPAR	Grupo de Pesquisa em Automação e Robótica
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
HD	*Hard Disk* (Disco Rígido)
IACRB	*Information Assurance Certification Board* (Conselho de Certificação de Segurança da Informação)
LADAR	*Laser Detection And Ranging* (Detecção e Telemetria de Laser)
LIDAR	*Light Detection And Ranging* (Detecção e Telemetria de Luz)
NASA	*National Aeronautics and Space Administration* (Administração Nacional da Aeronáutica e Espaço)
PC	*Personal Computer* (Computador Pessoal)
SSD	*Solid State Drive* (Unidade de Estado Sólido)
UART	*Universal Asynchrounous Receiver/Transmiter* (Receptor/Transmissor Universal Assíncrono)
UFC	Universidade Federal do Ceará
UML	*Unified Modeling Language* (Linguagem de Modelagem Unificada)
USB	*Universal Serial Bus* (Porta Universal)
VCC	Tensão (Volts) em corrente contínua

SUMÁRIO

1 INTRODUÇÃO

A história recente mostra um crescimento exponencial no último século no que se refere a novas tecnologias e produtos. A engenharia, ao mesmo tempo em que tem sido responsável, em parte, por esses avanços, também tem se desenvolvido para manter-se alinhada com esse progresso.

Por definição, engenharia é a aplicação do conhecimento científico, econômico, social e prático, com o intuito de inventar, desenhar, construir, manter e melhorar estruturas, máquinas, aparelhos, sistemas, materiais e processos (LANDTEC, n.d.). Com uma definição tão ampla, é notável a presença da engenharia no cotidiano de todas as pessoas, em suas ações diárias ou no ambiente que as cerca.

Por sua abrangência, é fato que a engenharia tem um papel fundamental na arte de inventar o progresso, mas é preciso reconhecer que nem todas as invenções e avanços do mundo moderno são resultantes de um processo de criação que surge de uma ideia completamente nova. Algumas revoluções ocorrem a partir do estudo do cenário atual, de produtos já existentes, mas que podem ser melhorados ou modificados de maneira até mesmo independente do seu propósito inicial. Apesar dessa visão reversa não ser tão comumente associada ao que as pessoas associam como engenharia, deve-se reconhecê-la como tal.

Em uma visão mais científica, o imaginário popular atribui a definição de engenharia ao que hoje é conhecido por engenharia progressiva. Portanto, a engenharia que é descrita como “a aplicação de princípios científicos e matemáticos para fins práticos, tais como a concepção, fabricação e operação de estruturas eficientes e econômicas, máquinas, processos e sistemas” (apud GHODKE e AWATI, n.d., p. 5) é apenas uma parte do conceito atual de Engenharia, já que se refere a definição de engenharia progressiva. Pode-se acrescentar a esse conceito, a utilização inversa da engenharia progressiva, que é a engenharia reversa.

Segundo Telea (2012, p. IX):

> A engenharia reversa abrange o conjunto de atividades destinadas a (re)descobrir a semântica funcional, estrutural e comportamental de um determinado artefato, com o objetivo de alavancar essas informações para um uso eficiente ou para adaptação desse artefato, ou mesmo a criação de artefatos relacionados.

A engenharia reversa é um resultado natural da curiosidade humana, ainda que possa também ter outras motivações, como econômicas e militares. O ser humano tem, inerente

a ele, uma necessidade de compreensão e de domínio sobre o mundo ao seu redor, que provoca avanços em todas as áreas do conhecimento humano. Quando essa curiosidade é empregada metodologicamente para entender um produto, seja sua operação, seu comportamento ou seu modelo, o que inclui códigos, hardware e montagens mecânicas, tem-se um projeto de engenharia.

De acordo com Várady et al (1996, p.1), enquanto a engenharia convencional (progressiva) transforma conceitos e modelos em partes reais, na engenharia reversa o caminho contrário ocorre (partes reais são transformadas em modelos e conceitos).

Apesar de muitas vezes ter sido associada ao simples processo de cópia, a engenharia reversa evoluiu e hoje pode ser empregada metodologicamente para entender um produto finalizado de modo a aprimorá-lo, modificá-lo e documentá-lo. Esse processo além de promover a difusão do conhecimento, pode inclusive suscitar inovações, o que é valorizado tanto no ambiente acadêmico quanto profissional. O tratamento da engenharia reversa como um processo sistemático válido como ciência e não como uma prática amadora é recente e depende da aplicação de uma metodologia ao longo do processo.

Ainda que seu uso remonte à Idade Antiga, já que a engenharia reversa é resultado da inerente necessidade humana de compreensão e de domínio sobre o mundo, sua recente popularização é consequência da rápida evolução científica, que cresceu não somente em volume, mas também em complexidade. Paralelamente a isso, os ciclos de desenvolvimento e produção tornaram-se mais curtos, uma vez que a concorrência de mercado impõe uma pressão neste, exigindo respostas rápidas e uma estrutura de custos cada vez mais competitiva (FURTADO e ASSAD, 2012 e TELEA, 2012). Segundo Mury (apud FURTADO e ASSAD, 2012, p. 2), a utilização de técnicas que possibilitem um rápido desenvolvimento ou adaptação de produtos é essencial para empresas e indústrias. Como consequência desse cenário, uma fração crescente do custo de criar novas aplicações e processos de fabricação mudaram da criação de novos produtos para a adaptação dos já existentes (TELEA, 2012, p. IX). É por essa razão que a engenharia reversa vem ganhando cada vez mais notoriedade.

Para se ter uma ideia do ganho de tempo obtido pela utilização da engenharia reversa, Sharples (2010) compara o tempo de construção de diferentes projetos ao se utilizar e ao não se utilizar a técnica. A redução de tempo varia entre 30% e 75%, dependendo da característica de projeto. A Tabela 1 exemplifica esse ganho considerável de tempo ao se utilizar a engenharia reversa em comparação a sua não-utilização.

Tabela 1 - Dados estimados do tempo necessário para construir uma embarcação naval chinesa, um navio regional chinês e um reator nuclear francês, com e sem a utilização de engenharia reversa

	Tempo de Construção	
Tipo de Veículo	Sem Engenharia Reversa	Com Engenharia Reversa
Embarcação Naval Chinesa	35 anos	15 anos
Frota Regional da Marinha Chinesa	35 anos	25 anos
Reator Nuclear Francês	31,5 meses	7,5 meses

Fonte: Adaptado de Sharples (2010)

De acordo com Stefanelli et al (apud FURTADO E ASSAD, 2012), a Engenharia Reversa não só acelera o ciclo de desenvolvimento de um produto, como também estimula a popularização de tecnologias. A redução de custos gerada pela utilização da engenharia reversa no processo contribui para o desenvolvimento mais abrangente de tecnologias e tem como consequência a redução do preço final dos produtos desenvolvidos, o que propicia sua difusão.

1.1 Justificativa

O uso da engenharia reversa é particularmente útil no ambiente acadêmico. Entender novas tecnologias e produtos e mesmo técnicas padronizadas amplamente utilizadas na indústria é uma ferramenta de consolidação de conhecimento. Alunos de níveis diversos, professores e pesquisadores podem se beneficiar do conhecimento teórico e prático que pode ser adquirido. Samuelson e Scotchmer (apud FURTADO e ASSAD, 2012) destacam a utilização da Engenharia Reversa para o processo de descoberta e aprendizagem, possibilitando que engenheiros não aprendam o estado da arte apenas por meio da leitura de artigos impressos e publicações, mas também na prática, o que leva muitas vezes a novos produtos e avanços de conhecimento.

O domínio de técnicas de engenharia reversa habilita profissionais a se adequarem a um mercado altamente competitivo, em que se exige desenvolvimento rápido a um custo econômico limitado.

Em universidades, a engenharia reversa pode até mesmo facilitar o processo de inovação, visto que parte de uma estrutura ou modelo já finalizado. A inovação por modificação de algo já existente é mais simples e rápida que a criação de algo inteiramente novo, mas pode

resultar em um progresso igualmente revolucionário.

Apesar da conveniência, quando essa técnica é aplicada na academia, normalmente carece de uma aplicação científica, sendo sua utilização associada a uma técnica natural e instintivo. Não há inadequação nisso, mas é importante ensinar aos alunos uma metodologia aplicável e mesmo o contexto em que a engenharia reversa deve ser utilizada (principalmente devido a controvérsias éticas largamente envolvidas na engenharia reversa).

Em alguns lugares, a engenharia reversa é devidamente vista como uma ciência e são ofertados cursos e disciplinas inteiras a seu respeito. A *Purdue University*, por exemplo, chegou a considerar sua inclusão no currículo básico do primeiro ano de Engenharia (DALRYMPLE, 2009). O IACRB (*Information Assurance Certification Board*) possui uma certificação para profissionais de engenharia reversa, o CREA, sigla em inglês para *Certified Reverse Engineering Analyst* (IACRB, n.d.). Esse tratamento da engenharia reversa como ciência e não como uma prática amadora comumente praticada é recente e depende da aplicação de uma metodologia ao longo do processo (SHARPLES, 2010). É esse reconhecimento como um campo de estudo válido que ainda precisa ser amplamente disseminado.

Este trabalho foi desenvolvido pela curiosidade em entender o conceito de engenharia reversa, pela necessidade de aplicá-la em um robô móvel real para adquirir conhecimentos e para produzir uma documentação até então inexistente. Sua proposta, contudo, vai além e se justifica pela necessidade de disseminar a engenharia reversa como uma ferramenta estruturada, importante e benéfica.

1.2 Objetivos

Este trabalho tem como principais objetivos:

- Contextualizar e esclarecer informações importantes sobre a engenharia reversa;
- Desenvolver uma metodologia de utilização da engenharia reversa, aplicável na robótica;
- Expor a aplicação dessa metodologia em um robô desenvolvido na NASA (*National Aeronautics and Space Administration*);
- Mostrar a documentação resultante do processo de engenharia reversa nesse robô.

1.3 Contribuições

Esse trabalho consolida uma metodologia de engenharia reversa aplicável na robótica. Além de propor as etapas necessárias dessa utilização, ainda ilustra seu emprego em um estudo de caso com um robô. Por conseguinte, o sistema de hardware e software do robô analisado não somente é explicado, como também uma documentação a seu respeito foi produzida e aqui apresentada.

Por fim, destaca-se como contribuição deste trabalho a propagação da engenharia reversa como uma ferramenta técnica que deve ser conhecida e empregada pelos alunos e professores.

1.4 Organização do texto

Este texto foi organizado em capítulos de modo a proporcionar uma leitura simples, com uma progressão natural dos assuntos abordados. Esta monografia é dividida em oito capítulos conforme a seguinte estrutura adotada:

- Capítulo 1: Este capítulo apresenta uma introdução deste trabalho, esclarecendo a motivação de sua elaboração, objetivos almejados e contribuições decorrentes.
- Capítulo 2: Este capítulo define o que é engenharia reversa e apresenta uma contextualização geral, abordando sua história, a legislação atual e o entendimento de sua necessidade.
- Capítulo 3: Este capítulo apresenta a proposição de uma metodologia de aplicação de engenharia reversa na robótica e detalha a metodologia utilizada.
- Capítulo 4: Este capítulo expõe a aplicação da engenharia reversa em um estudo de caso, um robô da NASA chamado Nanook.
- Capítulo 5: Este capítulo apresenta os resultados obtidos referentes à documentação do hardware do robô.
- Capítulo 6: Este capítulo apresenta os resultados obtidos referentes à documentação do software do robô.
- Capítulo 7: Este capítulo apresenta uma discussão geral dos resultados obtidos.
- Capítulo 8: Este capítulo contém conclusões e propostas para trabalhos futuros.

2 ENGENHARIA REVERSA

A Engenharia Reversa é bastante útil no âmbito acadêmico e industrial. Ainda que no primeiro caso seja fácil entender os benefícios da engenharia reversa, por proporcionar a difusão de conhecimentos e acelerar o processo de inovação, muito se questiona sobre o uso de engenharia reversa na indústria no geral devido a dicotomia do benefício de sua aplicação. O grande problema surge quando se confronta problemas relacionados a direitos autorais, violação de patente e pirataria, pois a engenharia reversa abre espaço para a descoberta de segredos industriais e comerciais. Apesar da tênue divisão entre o seu emprego legal e o seu emprego ilícito, influenciada majoritariamente pela necessidade do uso de uma postura ética para que se evite a infração da lei, a engenharia reversa vem ganhando espaço. Hoje, reconhece-se sua importância como uma metodologia empregável nos mais diversos contextos, permitindo não só o entendimento do objeto de análise como ações subsequentes a essa etapa inicial. A engenharia reversa ganha ainda mais destaque quando passa a ser utilizada na reengenharia, processo que exige uma etapa adicional além da simples cópia e, portanto, resulta em uma nova estrutura ainda que de mesma funcionalidade.

2.1 Conceitos

Ainda que geralmente associada a cópia de algo já existente, a engenharia reversa apresenta um conceito mais amplo. Não se trata de duplicar ou replicar tecnologias e processos, mas de estudar algo que já foi criado. Segundo Portella (2015), a “cópia é somente uma das consequências possíveis do estudo de um produto ou ideia”.

Entender claramente os conceitos associados à engenharia reversa ajuda não só a compreender sua dimensão de aplicabilidade, como também a evitar falsas associações preconceituosas a seu respeito.

De acordo com o dicionário Merriam-Webster (n.d.), engenharia reversa é o processo de desmontar e analisar um objeto para entender seu funcionamento, de modo a ser capaz de duplicá-lo ou aprimorá-lo. Em um conceito amplo, entende-se o processo de engenharia reversa como uma metodologia aplicada para entender algo que já existe. Para isso, pode-se estudar suas qualidades visuais, funcionais e até mesmo estruturais.

O próprio termo empregado já agrega a metodologia que é requerida na engenharia reversa. Enquanto um projeto comum de engenharia parte de um modelo lógico, para então definir as características do projeto e finalmente poder implementá-lo, um projeto de engenharia

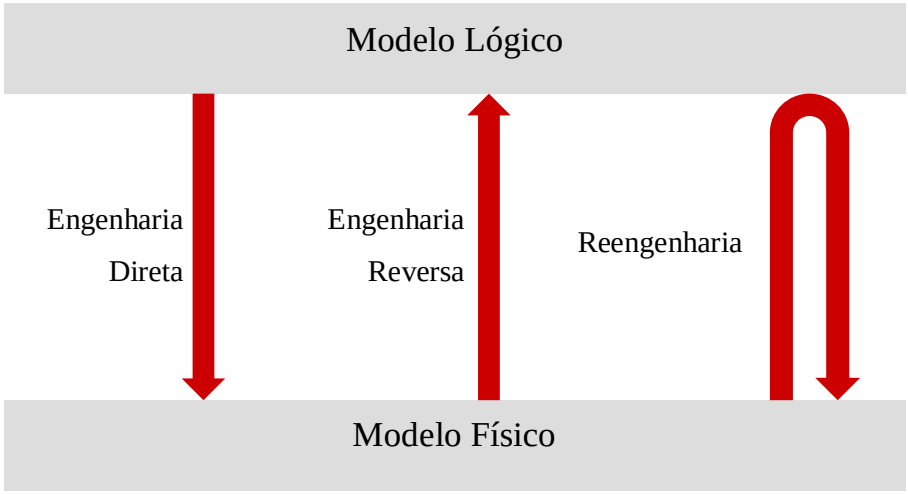
reversa começa com a observação de algo já implementado. Busca-se então entender as características desse modelo físico e, por fim, entender por completo o modelo lógico do objeto de estudo. A reengenharia complementa o conceito de engenharia reversa ao adicionar uma etapa subsequente de engenharia direta (BRAGA, n.d., p. 17 e CANHOTA JUNIOR et al, 2005, p. 6). Parte-se de um modelo físico já existente e após entender suas características de projeto, acrescenta-se a ele funcionalidades (estruturais ou conceituais) de modo a realizar um novo projeto de engenharia para implementá-las. O termo reengenharia está relacionado com a reconstrução de algo do mundo real, e independentemente de sua aplicação, o seu principal propósito é a busca por melhorias que permitam produzir algo de qualidade melhor ou, pelo menos, de qualidade comparável ao produto inicial (PIEKARSKI e QUINÁLIA, 2000, p. 40).

A distinção entre engenharia progressiva e reengenharia está relacionada ao ponto de partida de cada um dos processos. A engenharia progressiva inicia-se com uma especificação do que será construído, enquanto que a reengenharia inicia-se tomando como base um sistema já desenvolvido. Comparando a reengenharia com a engenharia reversa, percebe-se a diferenciação pelo ponto de chegada. O objetivo da reengenharia é produzir um sistema novo, enquanto a engenharia reversa á usada como parte do processo de reengenharia, visto que é responsável por fornecer o entendimento do sistema a ser reconstruído. (PIEKARSKI e QUINÁLIA, 2000, p. 40). A Figura 1 ilustra simplificadamente a diferença conceitual desses termos.

Talvez uma das formas mais simples de entender o contraste de engenharia reversa e engenharia direta (ou progressiva) é entender que enquanto em um projeto de engenharia um produto é construído (entendendo que esse produto pode ser conceitual, como um código, por exemplo), na engenharia reversa ocorre uma “desconstrução criativa” (TECHOPEDIA, n.d.), ou seja, a partir do produto finalizado se chega aos conceitos que o originaram. O termo desconstrução não implica em uma desmontagem necessária, apenas na subdivisão do produto final em partes que possibilitem o seu entendimento. Outra maneira de visualizar essa diferença é considerar o nível de abstração, entendendo isto como a capacidade de isolar um elemento para considerá-lo separadamente. Enquanto a engenharia progressiva parte de um alto nível de abstração para um baixo nível de abstração, a engenharia reversa parte de um baixo nível de abstração para um alto nível de abstração (BRAGA, n.d., p.4), já que esta segue o sentido oposto daquela. Desse modo, a engenharia tradicional (progressiva) parte de uma ideia ou necessidade, modela um produto, faz sua prototipagem e teste para então obter sua versão final; enquanto a engenharia reversa parte dessa versão final de um produto, desmonta-o ou desconstrói-lo,

realiza medições e testes para entender a sua modelagem, para, por fim, compreender suas funcionalidades e ideias a ele associadas.

Figura 1 - Diferenciação entre Engenharia Direta, Reversa e Reengenharia.



Fonte: Elaborado pelo autor.

Portanto, engenharia reversa abrange as metodologias empregadas para entender como um produto, processo ou sistema funciona, de modo a que, a partir do domínio das ideias e tecnologias empregadas no desenvolvimento do objeto em estudo, seja possível reunir informações suficientes para explicá-lo, reproduzi-lo e aperfeiçoá-lo.

2.2 Breve Contextualização Histórica

Não há muita certeza sobre o início da utilização de engenharia reversa. Seu desenvolvimento, no entanto, é associado a aplicações militares (FARIAS, 2009). A busca da superioridade militar evidenciou a utilização da engenharia reversa para que se pudesse copiar a tecnologia de adversários.

Um dos primeiros registros da aplicação de engenharia reversa remonta à Primeira Guerra Púnica, ocorrida entre 264 a.C. e 241 a.C. A marinha romana capturava navios na cidade de Cartago e então construía suas embarcações seguindo o mesmo modelo, sem dispender muito tempo no projeto (LAZENBY, 1996, p. 49).

Na I e na II Guerra Mundial, a captura de aeronaves, armamento e outras tecnologias garantia ao inimigo uma vantagem competitiva devido ao uso de engenharia reversa.

Essa estratégia foi largamente utilizada e garantiu não só algumas vitórias decisivas, como também o desenvolvimento de novas tecnologias pelo aperfeiçoamento das tecnologias já existentes. Os soviéticos, por exemplo, desenvolveram uma aeronave de tecnologia de ponta durante a II Guerra Mundial a partir da engenharia reversa do Boeing B-29 *Superfortress* de domínio americano. Eles haviam conseguido apreender três dessas aeronaves americanas e decidiram manter uma como modelo de referência, uma para treinamento de pilotos e uma foi desmontada para medições e análise das 105.000 partes listadas dessa aeronave (SRINIVAS, 2010). Ao final do processo, os soviéticos conseguiram desenvolver um modelo superior ao dos americanos (MAYO, n.d.). A Figura 2 apresenta a imagem da aeronave desenvolvida pelos soviéticos e o modelo americano utilizado como base para a engenharia reversa.

Figura 2 - O Tupolev Tu-4, à esquerda, foi desenvolvido pelos soviéticos a partir da engenharia reversa do Boeing B-29 *Superfortress* dos EUA, à direita.

Fonte: Srinivas (2010) e Wang (2010).

Todavia, não somente em guerras a engenharia reversa era utilizada. Mesmo após a II Guerra Mundial, essa metodologia foi utilizada para, por exemplo, alavancar a recuperação do Japão. Segundo Paulino (2009), “foram desenvolvidos processos de reconstrução de produtos como ‘cópias fiéis – piratas’, a partir de modelos previamente estudados, principalmente os eletroeletrônicos e na indústria automobilística”, o que resultou em um expressivo retorno econômico. A utilização da engenharia reversa para cópia permitia a diminuição de custos com investimento em pesquisas e em estudos de implementação.

Hoje, além da cópia de produtos originais e do roubo de tecnologias, a engenharia reversa é empregada nas mais diferentes áreas do conhecimento, podendo ser aplicada positivamente em diferentes contextos, sendo um exemplo o emprego neste trabalhado.

2.3 Aplicações Positivas da Engenharia Reversa

Apesar da origem histórica associada mais fortemente ao aspecto negativo da engenharia reversa, ela pode ser usada positivamente para fins acadêmicos e avanços em pesquisa. Segundo Genghini (2013, p. 14), a importância da engenharia reversa pode ser entendida quando se considera situações em que há necessidade de "reproduzir um determinado produto a partir de um protótipo feito à mão, de produtos descontinuados, projetos muito antigos e/ou que não existam informações disponíveis" ou de efetuar "alterações em ferramentas ou processos de fabricação".

É fácil perceber que, quando por algum motivo inexiste a documentação do objeto de interesse, a engenharia reversa é a melhor metodologia para tornar possível a especificação deste. Isso ocorre, por exemplo, com tecnologias antigas sem documentação. Em algumas situações, até existe uma documentação inicial, mas a falta de sua atualização ao longo dos tempos gera uma defasagem entre a documentação existente e a configuração real. No cotidiano, para produtos de menor escala, é também comum deparar-se com situações em que a documentação não era necessária por que havia pessoas que dominavam aquela informação, ferramenta, produto ou sistema, mas que depois de um tempo deixam aquele ambiente. A partir daí surge a necessidade de outras pessoas também entenderem o raciocínio ali existente. Isso pode ser dificultado ainda mais porque algumas implementações de sistema podem ser feitas sem método algum ou por diferentes pessoas com métodos diferentes, cada uma sendo responsável por uma parte da implementação. A engenharia reversa é, portanto, bastante útil nessas situações. Mesmo em casos de atualização de tecnologia, em que um produto defasado precisa ser remodelado para que possa continuar sendo útil dado o contexto tecnológico existente, a técnica se mostra um meio bastante eficaz.

Além da já destacada utilização em cópia, melhoria de deficiência de documentação, modernização de produto, espionagem militar ou comercial e fins acadêmicos, observando o estado da arte de aplicações de engenharia reversa, pode-se ressaltar ainda como potenciais aplicações (PATEL, 2014):

- Estudo de produtos que já são abertos no mercado;
- Aquisição de conhecimento pessoal no produto e sua tecnologia;
- Compatibilização do produto com tecnologias disponíveis no mercado para interoperabilidade;
- Verificação da conformidade de um produto com os padrões anunciados;
- Determinação de falhas no produto quando comparado a competidores;

- Reprojeto de sistemas obsoletos, incorporando novas tecnologias;
- Identificação de potencial violação de patente ou direitos autorais.

2.4 Legislação

Segundo Samuelson e Scotchmer (2002, p. 2) e Furtado e Assad (2012, p. 4), se for considerado que a engenharia reversa é a extração de know-how ou conhecimento de um produto feito pelo homem, tem-se uma longa história como uma prática amplamente exercida e aceita. Apesar disso, existem certas exigências de restrições e legislações para proteger contra o mau uso da engenharia reversa, que consiste basicamente em copiar um produto para obtenção de lucro sem autorização. Essas legislações são específicas em cada país e baseiam-se principalmente na ideia de patente.

Segundo Sharples (2010), as patentes apareceram na Itália no século XV justamente como uma maneira de proteger os inventores de pessoas que poderiam copiar seus produtos e vendê-los para obter lucro.

Ainda segundo Sharples (2010), as duas preocupações legais da engenharia reversa são violação de patente e violação de modelo. Duplicar um produto patenteado é uma violação de patente, enquanto roubar o design de um componente ou sistema constitui uma violação de modelo. Na maioria dos casos, entretanto, a violação legal só ocorre quando o produto é protegido inteiramente por patente. Quando nenhuma ou apenas uma parte do componente é patenteado, a duplicação pode ser considerada legal.

Essas considerações, todavia, dependem da legislação de cada país e é uma preocupação para detentores de propriedade intelectual países que não possuem leis específicas sobre o assunto, pois o controle regulatório do uso adequado da engenharia reversa é dificultado. Ao longo dessa seção, são abordados alguns aspectos legais definidos em certos países ou regiões que tratam de restrições quanto a legalidade do uso da engenharia reversa, conforme exposto em Canhota Junior et al. (2005, p.12). Um sumário dos aspectos legais atribuídos às regiões exploradas nesta seção é apresentado na Tabela 2.

Tabela 2 - Atribuição de aspectos legais relativos à engenharia reversa por região

Região	Possui proteção aos direitos autorais e propriedade intelectual?	Possui legislação relativa à engenharia reversa?
Brasil	✓	✗
Estados Unidos	✓	✓
União Europeia	✓	✓
Japão	✓	✗

Fonte: Elaborado pelo autor com base em Canhota Junior et al. (2005, p.12).

2.4.1 Brasil

No Brasil, apesar da inexistência de legislação específica sobre engenharia reversa, existem a Lei de Software (Lei nº 9.609/98) e a Lei de Direitos Autorais (Lei nº 12.853/2013), que são usadas para definir a legalidade da utilização de engenharia reversa dado o fim a que seu uso se destina. Enquanto a engenharia reversa não resultar em pirataria ou infração de algum direito autoral, ela é considerada lícita (BRASIL, 1998a; BRASIL, 1998b e BRASIL, 2013).

Os Anexos A e B apresentam uma visualização preliminar das leis nº 9.609/98 e 12.853/2013 que estabelecem algumas restrições à prática da engenharia reversa, por infringirem direitos autorais ou de propriedade intelectual. Em seguida, alguns trechos das referidas leis são destacados na Figura 3 e na Figura 4.

Figura 3 - Trechos da Lei de Software.

CAPÍTULO II

DA PROTEÇÃO AOS DIREITOS DE AUTOR E DO REGISTRO

Art. 2º O regime de proteção à propriedade intelectual de programa de computador é o conferido às obras literárias pela legislação de direitos autorais e conexos vigentes no País, observado o disposto nesta Lei.

Art. 6º Não constituem ofensa aos direitos do titular de programa de computador:

I - a reprodução, em um só exemplar, de cópia legitimamente adquirida, desde que se destine à cópia de salvaguarda ou armazenamento eletrônico, hipótese em que o exemplar original servirá de salvaguarda;

II - a citação parcial do programa, para fins didáticos, desde que identificados o programa e o titular dos direitos respectivos;

III - a ocorrência de semelhança de programa a outro, preexistente, quando se der por força das características funcionais de sua aplicação, da observância de preceitos normativos e técnicos, ou de limitação de forma alternativa para a sua expressão;

IV - a integração de um programa, mantendo-se suas características essenciais, a um sistema aplicativo ou operacional, tecnicamente indispensável às necessidades do usuário, desde que para o uso exclusivo de quem a promoveu.

CAPÍTULO V

DAS INFRAÇÕES E DAS PENALIDADES

Art. 12. Violar direitos de autor de programa de computador:

Pena - Detenção de seis meses a dois anos ou multa.

§ 1º Se a violação consistir na reprodução, por qualquer meio, de programa de computador, no todo ou em parte, para fins de comércio, sem autorização expressa do autor ou de quem o represente:

Pena - Reclusão de um a quatro anos e multa.

§ 2º Na mesma pena do parágrafo anterior incorre quem vende, expõe à venda, introduz no País, adquire, oculta ou tem em depósito, para fins de comércio, original ou cópia de programa de computador, produzido com violação de direito autoral.

Fonte: Adaptado de Brasil (1998a).

Figura 4 - Trechos da Lei de Direitos Autorais.

Título I

Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 14. É titular de direitos de autor quem adapta, traduz, arranja ou orquestra obra caída no domínio público, não podendo opor-se a outra adaptação, arranjo, orquestração ou tradução, salvo se for cópia da sua.

Art. 30. No exercício do direito de reprodução, o titular dos direitos autorais poderá colocar à disposição do público a obra, na forma, local e pelo tempo que desejar, a título oneroso ou gratuito.

§ 1º O direito de exclusividade de reprodução não será aplicável quando ela for temporária e apenas tiver o propósito de tornar a obra, fonograma ou interpretação perceptível em meio eletrônico ou quando for de natureza transitória e incidental, desde que ocorra no curso do uso devidamente autorizado da obra, pelo titular.

Art. 33. Ninguém pode reproduzir obra que não pertença ao domínio público, a pretexto de anotá-la, comentá-la ou melhorá-la, sem permissão do autor.

Capítulo IV

Das Limitações aos Direitos Autorais

Art. 46. Não constitui ofensa aos direitos autorais:

II - a reprodução, em um só exemplar de pequenos trechos, para uso privado do copista, desde que feita por este, sem intuito de lucro;

IV - o apanhado de lições em estabelecimentos de ensino por aqueles a quem elas se dirigem, vedada sua publicação, integral ou parcial, sem autorização prévia e expressa de quem as ministrou;

VIII - a reprodução, em quaisquer obras, de pequenos trechos de obras preexistentes, de qualquer natureza, ou de obra integral, quando de artes plásticas, sempre que a reprodução em si não seja o objetivo principal da obra nova e que não prejudique a exploração normal da obra reproduzida nem cause um prejuízo injustificado aos legítimos interesses dos autores.

Fonte: Adaptado de Brasil (2013).

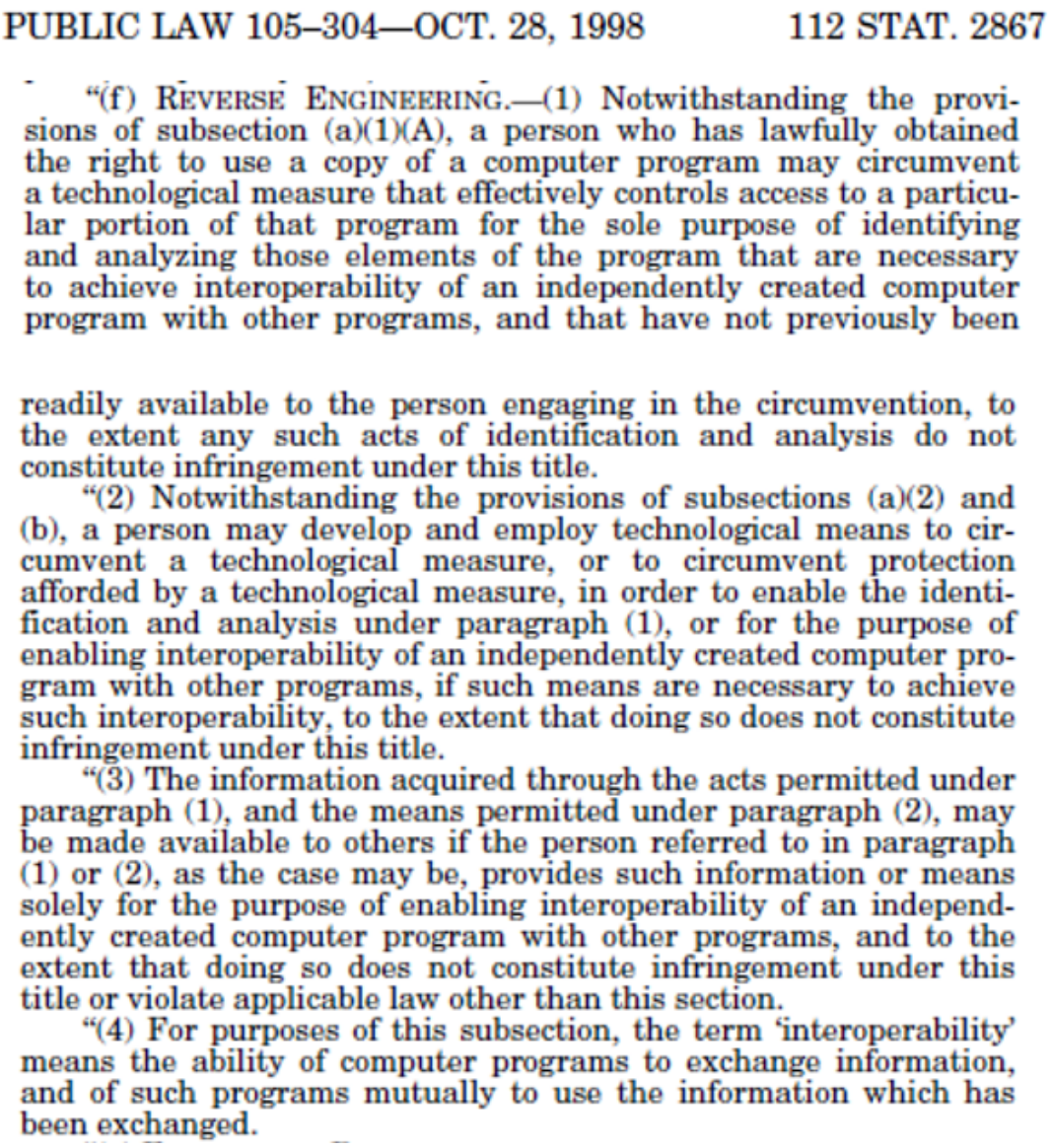
2.4.2 Estados Unidos

Nos Estados Unidos existe uma lei bem conhecida que faz restrições à utilização de engenharia reversa: o *Digital Millennium Copyright Act*, ilustrado parcialmente na Figura 5. Por essa lei, aplicada ao mundo digital e eletrônico, a engenharia reversa só pode ser utilizada para analisar a compatibilidade com outros software e hardware.

Já um produto protegido por segredos comerciais pode ser analisado por engenharia reversa se ele for obtido legalmente. Em relação a patentes, como é exigido uma divulgação

pública da invenção, muitas vezes não é nem mesmo necessário o processo de engenharia reversa.

Figura 5 - Trecho do *Digital Millenium Copyright Act*.



PUBLIC LAW 105–304—OCT. 28, 1998 112 STAT. 2867

"(f) REVERSE ENGINEERING.—(1) Notwithstanding the provisions of subsection (a)(1)(A), a person who has lawfully obtained the right to use a copy of a computer program may circumvent a technological measure that effectively controls access to a particular portion of that program for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs, and that have not previously been readily available to the person engaging in the circumvention, to the extent any such acts of identification and analysis do not constitute infringement under this title.

"(2) Notwithstanding the provisions of subsections (a)(2) and (b), a person may develop and employ technological means to circumvent a technological measure, or to circumvent protection afforded by a technological measure, in order to enable the identification and analysis under paragraph (1), or for the purpose of enabling interoperability of an independently created computer program with other programs, if such means are necessary to achieve such interoperability, to the extent that doing so does not constitute infringement under this title.

"(3) The information acquired through the acts permitted under paragraph (1), and the means permitted under paragraph (2), may be made available to others if the person referred to in paragraph (1) or (2), as the case may be, provides such information or means solely for the purpose of enabling interoperability of an independently created computer program with other programs, and to the extent that doing so does not constitute infringement under this title or violate applicable law other than this section.

"(4) For purposes of this subsection, the term 'interoperability' means the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged.

Fonte: Estados Unidos (1998).

2.4.3 União Europeia

Na União Europeia, existe a *EU Copyright Directive* (UNIÃO EUROPEIA, 2009), criado em 1991 e atualizado em 2009. Segundo sua versão mais recente, a reprodução não-autorizada, tradução, adaptação ou transformação da versão disponível de um código constitui uma violação de direitos autorais. Algumas circunstâncias excepcionais são autorizadas. Por

exemplo, quando a intenção é garantir a interoperabilidade entre um programa criado independentemente e outros programas, o processo de engenharia reversa utilizado é considerado legal, desde que não prejudique os interesses legítimos do autor.

2.4.4 Japão

No Japão, a lei aplicável é a Lei Japonesa de Concorrência Desleal (Japão, 2015), ilustrada parcialmente em uma versão traduzida para o inglês na Figura 6. Essa lei proíbe a imitação por um período de três anos, mesmo que se trate de produtos não patenteados ou não protegidos por direitos autorais. Dentre as restrições de aplicabilidade da lei, destaca-se que casos de progresso técnico ou de padronização e compatibilização industrial são permitidos. Além disso, ideias e conceitos não são considerados no âmbito dessa lei, requer-se patentes para tal fim.

Figura 6 - Ilustração parcial da Lei Japonesa de Concorrência Desleal, versão traduzida em inglês sem validade legal.

この不正競争防止法の翻訳は、平成二十七年法律第五十四号までの改正（平成２８年１月１日施行）について作成したものです。
この法令の翻訳は公定訳ではありません。法的効力を有するのは日本語の法令自体であり、翻訳はあくまでその理解を助けるための参考資料です。この翻訳の利用に伴って発生した問題について、一切の責任を負いかねますので、法律上の問題に関しては、官報に掲載された日本語の法令を参照してください。

This English translation of the Unfair Competition Prevention Act has been prepared (up to the revisions of Act No. 54 of 2015 (Effective January 1, 2016)).
This is an unofficial translation. Only the original Japanese texts of laws and regulations have legal effect, and the translations are to be used solely as reference material to aid in the understanding of Japanese laws and regulations.
The Government of Japan shall not be responsible for the accuracy, reliability or currency of the legislative material provided in this Website, or for any consequence resulting from use of the information in this Website. For all purposes of interpreting and applying law to any legal issue or dispute, users should consult the original Japanese texts published in the Official Gazette.

Unfair Competition Prevention Act (Act No.47 of 1993)

Table of Contents

Fonte: Japão (2015).

2.4.5 Clean Room Design

Apesar da legislação anteriormente apresentada proteger os direitos autorais contra possíveis cópias não-autorizadas, existe uma metodologia de engenharia reversa empregada que pode ser aplicada para contornar esses impedimentos legais: o *clean room design* (STEFANELLI et al, 2010).

Segundo Drizin e Buonanni (2016), o *clean room design* é:

> "Um método onde se pode utilizar a engenharia reversa para entender o funcionamento de um software e cloná-lo, desde que o time que faça a codificação do clone tenha contato apenas com a especificação do sistema, feita pelo time que fez a engenharia reversa. Deste modo alguém que teve contato com o código original não pode participar do desenvolvimento direto do clone, apenas da especificação detalhada".

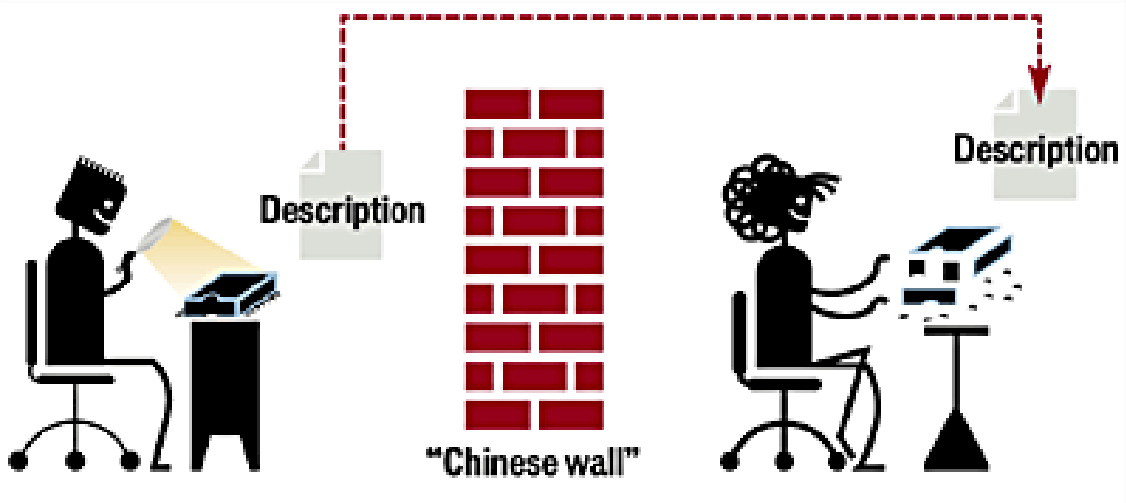
O *clean room design* utiliza a engenharia reversa para cópia agregando uma invenção independente, por isso é capaz de contornar infrações legais ao direito autoral. Contudo, essa adição independente não é suficiente para trespassar as restrições de patente, o que torna esse método inválido neste último caso.

Existem alguns exemplos em que o *clean room design* foi utilizado para evitar que medidas legais fossem aplicadas ainda que houvesse cópia de produto concorrente. Um dos casos mais notórios ocorreu em 1982, quando a Compaq copiou o IBM-PC, recém lançado pela IBM. Apesar de o código da BIOS do IBM-PC ser de propriedade da IBM e, portanto, ser protegido, a Compaq produziu um computador pessoal compatível. Quando acionada, a justiça se manifestou observando que a Compaq não poderia copiar diretamente a BIOS da IBM, mas poderia fazer a engenharia reversa desta e criar sua própria BIOS usando *clean room design*. (DRIZIN E BUONANNI, 2016). "Foi a partir do c*lean room design*, e da iniciativa da Compaq, que surgiu a indústria dos clones de PC (computador pessoal) e a popularização destes sistemas que alicerçaram a chamada 'cultura da informação'" (STEFANELLI et al, 2010).

Normalmente, para aplicar a metodologia de *clean room design* é necessário que haja um ambiente "limpo", ou seja, comprovadamente sem conhecimento de quaisquer técnicas utilizadas pelo proprietário. Pode-se, para isso, separar equipes diferentes para realizar o processo. Primeiramente, um grupo de pessoas examinaria o sistema que será reimplementado e escreveria suas especificações detalhadamente. Se essas especificações não

incluírem nenhum material que infrinja algum direito autoral, o que pode ser assegurado por um advogado, então um outro grupo, sem conexões com o anterior, poderia utilizar essas especificações para implementar o produto. A Figura 7 ilustra esse processo. Mesmo que o resultado inclua partes idênticas do produto copiado, não há infração de direito autoral devido à utilização da metodologia de engenharia reversa adequada (SCHWARTZ, 2001).

Figura 7 – Utilização da metodologia de *clean room* para aplicar engenharia reversa.



Fonte: Schwartz (2001).

3 METODOLOGIA

O processo de engenharia reversa apresenta complicação adicional comparativamente à engenharia progressiva porque algumas relações lógicas não são evidentes no modelo físico. Essas informações talvez precisem de alguma forma serem recuperadas por análise de dados ou por inferência de relações ausentes (STACK OVERFLOW, 2015). Para garantir que o objetivo final do processo de engenharia reversa seja alcançado, é necessário estruturar uma metodologia básica de ação. Como a engenharia reversa tem ganhado bastante espaço a ponto de ser reconhecida como um campo de estudo válido, algumas metodologias aplicáveis já estão estruturadas.

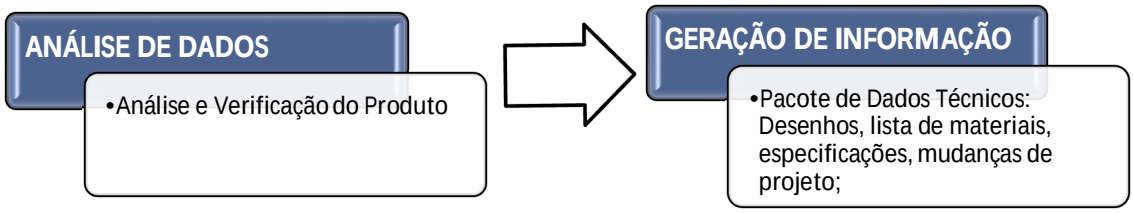
A consolidação de um processo sistemático é de fundamental importância porque diferencia uma mera atividade cotidiana para satisfazer a curiosidade pessoal de uma ciência aplicada.

Apesar de ser mais comum falar sobre metodologias de engenharia reversa aplicáveis a campos específicos, como aplicação automobilística ou em software, existem duas etapas básicas a serem seguidas, enumeradas abaixo e ilustradas na Figura 8:

I. Análise de dados;

II. Geração de informação.

Figura 8 - Etapas básicas de Engenharia Reversa.



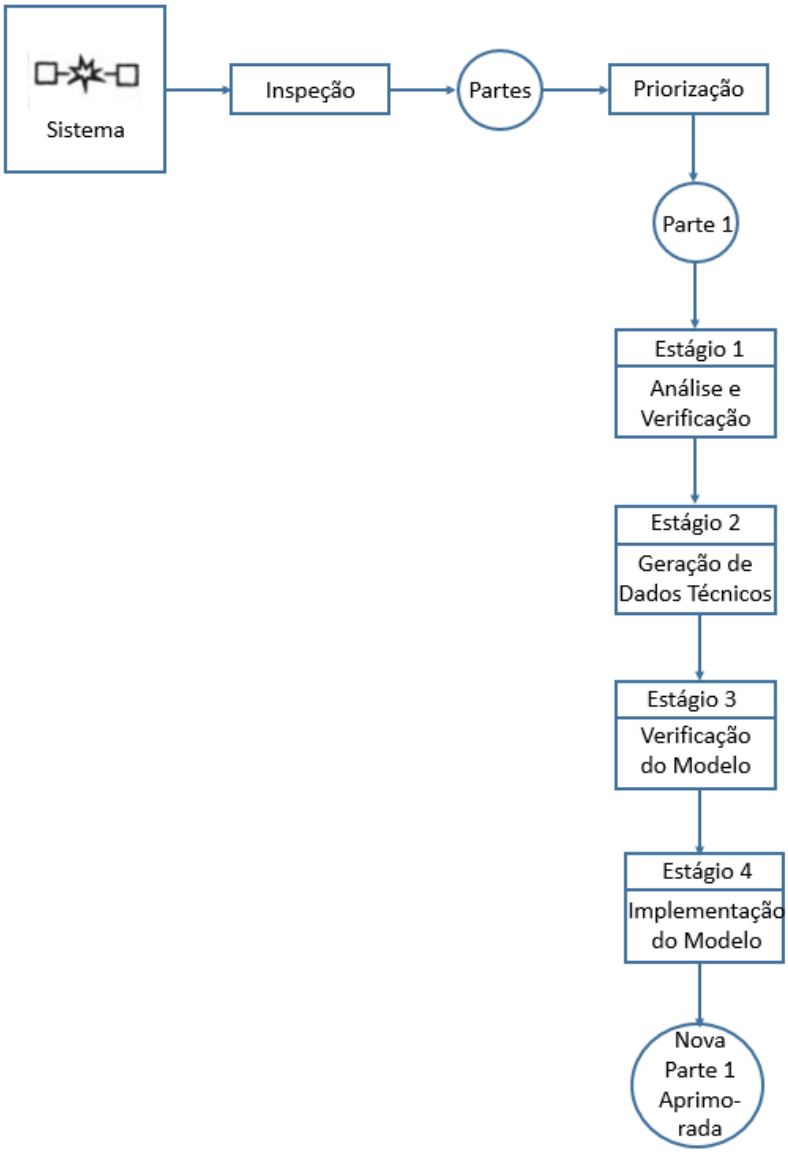
Fonte: Adaptado de Sharples (2010).

Na etapa de análise de dados, pode-se incluir tarefas como inspeções visuais, montagem e desmontagem, análise de material, testes operacionais e análise de falhas. O processo de montagem e desmontagem é interessante porque é nessa fase que se pode responder a várias dúvidas de “porquês” e “como” (SHARPLES, 2010), podendo-se determinar se a resposta é por razões de design, disponibilidade de material, funcionalidade ou alguma outra razão.

A etapa de geração de informação requer que um pacote de dados técnicos seja produzido para documentar as informações obtidas, que são os dados corroborados no processo de análise.

Ingle (1994) apresenta uma metodologia de quatro estágios, ilustrada na Figura 9, que além da análise de dados e geração de informação, ainda inclui duas etapas voltadas para o processo de reengenharia: verificação do modelo e implementação do modelo. Essas novas etapas averiguam o funcionamento atual do produto e propõem aperfeiçoamentos subsequentemente implementados.

Figura 9 - Visão geral do processo de engenharia reversa proposto por Ingle (1994).



Fonte: Ingle (1994).

A grande importância da metodologia apresentada por Ingle (1994), além do seu destaque na literatura correlata, consiste na identificação de etapas preliminares à sua metodologia propriamente dita. Esses passos são importantes para reconhecer o projeto e assegurar a sua viabilidade técnica e financeira. Assim, a escolha do projeto a ser estudado, deve ser seguida por uma inspeção preliminar que identifique suas partes e forneça um grau de prioridade do projeto em termos de sua exequibilidade. Somente após a aprovação do procedimento no sistema é que se deve prosseguir com os quatro estágios do processo, que resultará em um novo produto aperfeiçoado.

Manter a consistência no procedimento e ter em mente o objetivo almejado é o mais importante, independentemente do algoritmo aplicado. É relevante entender quais questões se deseja responder para que se possa estabelecer uma metodologia eficaz para obtê-las. A consistência metodológica evita o acréscimo de erros que afetem o resultado.

3.1 Metodologia Aplicada à Robótica

Pensando em aplicações no campo da robótica, as etapas básicas descritas anteriormente podem ser desdobradas em etapas adicionais. Algumas sugestões (SHARPLES, 2010) são listadas abaixo.

Na análise de dados, pode-se:

- Determinar o tamanho, a forma, o peso do objeto e as tolerâncias envolvidas;
- Desmontar o robô ou uma parte dele, se isso for possível, o que pode ajudar a entender melhor o design do produto e seu processo de montagem. Uma boa prática nesse processo é documentar a listagem das peças e mesmo a ordem em que foram retiradas;
- Verificar o desempenho e as características do robô e comparar os resultados com as informações esperadas, conforme obtidas em desenhos técnicos e especificações do projeto;
- Realizar testes operacionais, o que pode ajudar a entender a escolha de alguns parâmetros do robô.

Na geração de informação, o pacote de dados técnicos deve ser completo o suficiente, de modo a garantir a reprodutibilidade e o fácil entendimento do projeto. Portanto, pode-se:

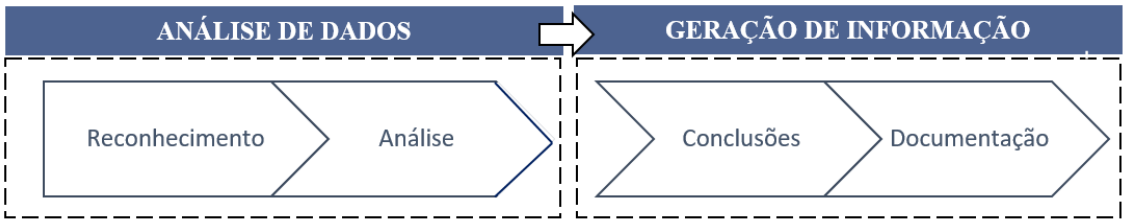
- Incluir desenhos cotados e modelos sólidos, bem como as tolerâncias nos dados;
- Indicar quaisquer alterações feita no projeto, sejam elas adequações de partes e materiais obsoletos ou aprimoramentos;
- Determinar as razões anteriores para determinação das decisões de projeto.

3.2 Metodologia Proposta

A metodologia empregada neste trabalho consistiu de quatro etapas básicas, sendo uma versão adaptada do processo de engenharia reversa exposto acima.

Na etapa de análise de dados, definiu-se como subetapas o reconhecimento do projeto e a análise em si. Na etapa de geração de informação, as subetapas foram as conclusões advindas da análise e a documentação gerada, conforme exposto na Figura 10.

Figura 10 - Etapas básicas da metodologia adotada.



Fonte: Elaborado pelo autor.

Esse processo de engenharia reversa permitiu o reconhecimento de um sistema completo, a partir de sua caracterização e documentação para posteriores consultas e aprimoramentos.

Em um passo-a-passo mais específico da metodologia adotada, as principais ações foram as especificadas na Tabela 3.

Tabela 3 - Passo-a-passo das etapas da metodologia adotada

Análise de dados	Reconhecimento	1) Reconhecimento do projeto;
		2) Estabelecimento de objetivos;
		3) Busca e compilação das informações disponíveis;
		4) Inspeção visual;
	Análise	5) Conclusões preliminares;
		6) Investigações pontuais;
		7) Análise completa de hardware e software;
	Conclusões	8) Conclusões finais;
Geração de Informação	Documentação	9) Documentação;
		10) Aprimoramentos (executados e propostos).

Fonte: Elaborado pelo Autor.

A diferenciação das metodologias propostas para as anteriormente apresentadas deve-se à sua aplicabilidade na robótica. As metodologias desenvolvidas são, em geral, específicas para a atuação de destino, restringindo sua utilização em outros projetos. Buscou-se propor uma metodologia que fosse facilmente replicável no caso de documentação inexistente ou incompleta. Muitas metodologias, ainda que aplicáveis em algum campo da robótica, especificam a utilização da técnica para a cópia de apenas uma parte do robô (muitas vezes, somente o modelo mecânico ou software). Devido a necessitar de documentar um robô móvel real completo, foi necessário estabelecer uma metodologia mais abrangente capaz de suprir os objetivos desse trabalho.

A metodologia proposta por Ingle (1994) é voltada para o processo de reengenharia de processos industriais, focando na necessidade de substituir peças obsoletas sem documentação. Portanto, não há uma menção sobre a engenharia reversa de software, necessária no caso do Nanook. A metodologia proposta por Sharples (2010) foca na reengenharia de um veículo subaquático autônomo, carecendo de detalhes que permitissem a replicação de seu uso por outras pessoas.

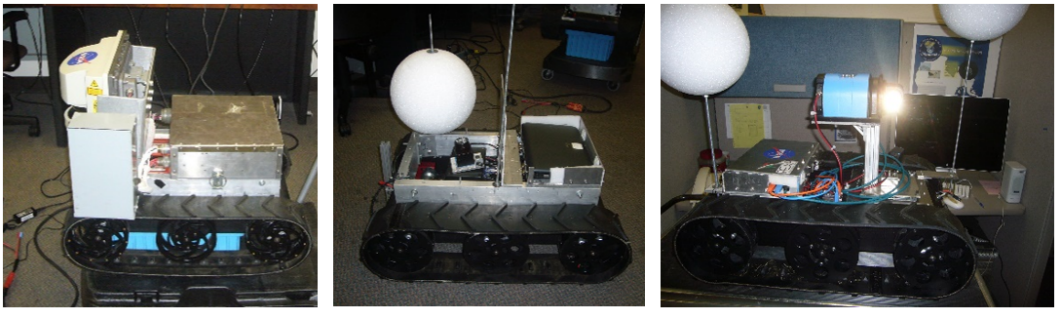
A metodologia proposta nesse trabalho possui embasamento nas identificadas por Ingle (1994) e Sharples (2010), mas as complementa para prover as necessidades de aplicação não expostas naquelas.

4 ESTUDO DE CASO

A metodologia exposta neste trabalho foi aplicada na engenharia reversa de um robô, o Nanook. Como será detalhado mais a frente, o Nanook é um robô desenvolvido por estagiários da NASA liderados por Michael Comberiate, o NASA Mike. Ele foi desenvolvido juntamente com outros dois robôs (Pinguim 1 e Pinguim 2) para explorar terrenos remotos e hostis, realizando ao mesmo tempo testes de campo de protocolos de comunicação com transferência de grande quantidade de dados. Esses robôs são ilustrados na Figura 11. O desenvolvimento desses três robôs objetivava ser capaz de reconhecê-los para que fosse produzida uma imagem 3D do ambiente. A adição de funcionalidades a um dos robôs auxiliares, como um braço robótico, permitiria a manipulação do ambiente, como recolhimento de amostras para análises posteriores.

O projeto de engenharia reversa surgiu da necessidade de documentar o robô Nanook, uma vez que essa documentação nunca havia sido desenvolvida. As maiores dificuldades encontradas advinham da necessidade de ser um projeto não destrutivo e dos sistemas do robô serem não-padronizados (diferentes pessoas trabalharam com diferentes métodos e estilos ao longo dos anos).

Figura 11 - Nanook, Pinguim 1 e Pinguim 2



Fonte: Acervo Pessoal de Patrick Stakem

4.1 Contextualização

Foi durante um programa conhecido em inglês por NASA/*CapTechU Engineering Bootcamp* que, quase cinco anos depois dos primeiros passos com o Nanook, Michael Comberiate lançou um desafio aceito por três estudantes da Universidade Federal do Ceará: entender e descrever o complexo funcionamento de um robô da NASA. Os três estudantes

faziam parte de um grupo de brasileiros em intercâmbio acadêmico nos Estados Unidos pelo programa Ciência sem Fronteiras e participaram de um projeto colaborativo entre a *Capitol Technology University* e engenheiros e cientistas da NASA.

O grupo que desenvolveu um projeto de engenharia reversa foi não só capaz de descrever o robô, como também de propor e implementar algumas melhorias. Com o meticuloso trabalho descrevendo o hardware, o software e a metodologia de engenharia reversa utilizada, a equipe conseguiu conquistar a admiração do engenheiro sênior da NASA. Um projeto de engenharia reversa é particularmente difícil porque requer a compreensão de um sistema complexo sem que seja possível simplesmente desmontá-lo. Ao desenvolver a completa descrição do robô, o trabalho dos três estudantes permite que ele seja duplicado ou que melhorias sejam implementadas muito mais eficientemente.

Foi pensando nessas possibilidades que Michael Comberiate resolveu propor uma colaboração com a Universidade Federal do Ceará (UFC) em um Projeto Especial de Robótica. O acordo foi firmado com o GPAR (Grupo de Pesquisa em Automação e Robótica), resultando na vinda do robô para o Departamento de Engenharia Elétrica e na realização de uma palestra em Fortaleza do experiente engenheiro.

4.2 Nanook

O Nanook foi desenvolvido com o objetivo de testar novas tecnologias. Sua principal aplicabilidade é a exploração de áreas remotas, utilizando protocolos de comunicação que hoje são usados nos *rovers* (veículos autônomos de exploração) em Marte. O robô semiautônomo, ilustrado na Figura 12, requer mínimos comandos do centro de controle e ainda assim é capaz de obter imagens do ambiente explorado por meio de uma tecnologia de sensoriamento remoto que utiliza laser, o LIDAR (*Light Detection And Ranging*, sigla em inglês para Detecção e Telemetria de Luz). O Nanook já realizou testes operacionais no Alasca, Polo Norte, Polo Sul e México, sendo capaz de obter com sucesso imagens dessas regiões. Apesar do trabalho pioneiro desenvolvido na NASA, nenhuma documentação foi concomitantemente elaborada para o robô.

De acordo com Stakem (2015), o objetivo do projeto que culminou no Nanook era:

> “[...] montar um time de robôs autônomos interconectados que pudessem ser usados para mapeamento tridimensional de um terreno, gerar imagens de alta resolução e coletar amostras de territórios inexplorados”.

Figura 12 - Nanook.

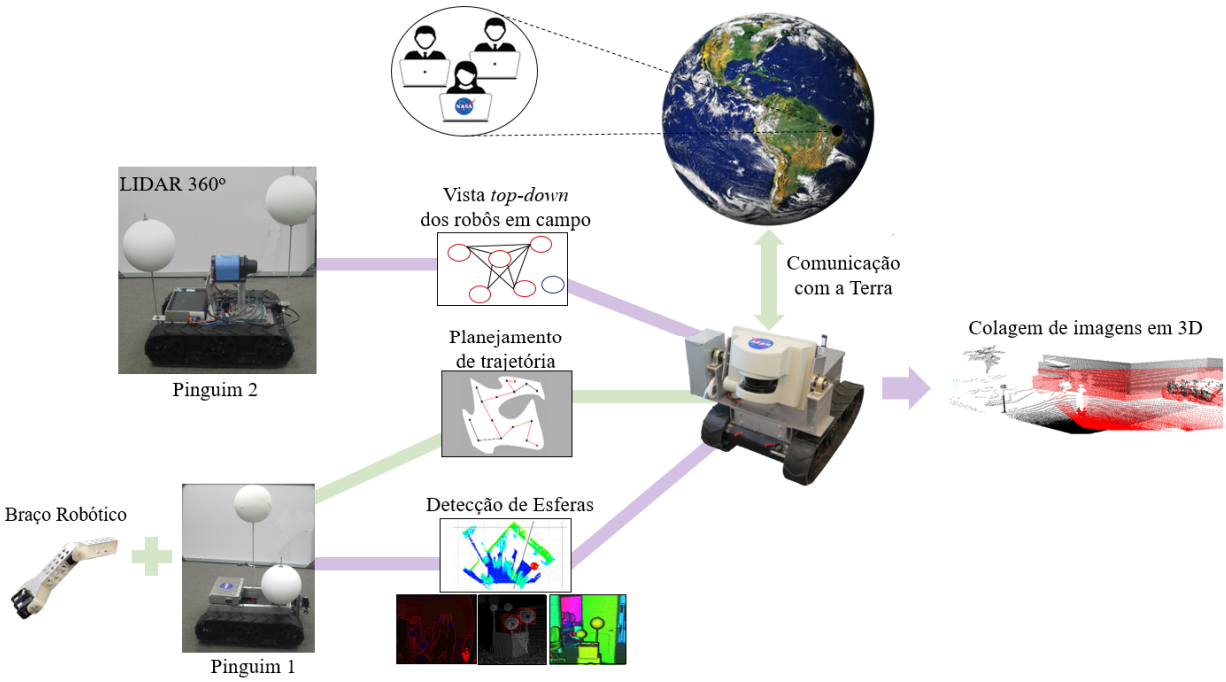
Fonte: Fornecido por Michael Comberiate.

Assim como na mitologia que deu origem ao nome do robô (na mitologia Inuit, Nanook é o mestre dos ursos polares), o Nanook é considerado o robô-mestre, sendo responsável por localizar outros dois robôs auxiliares (Pinguim 1 e 2), o que era feito baseado na detecção de esferas (cada robô auxiliar possuía uma esfera em seu topo unicamente disposta).

O rastreamento de determinada área era feito pelo envio de sucessivas imagens coladas para o centro de controle. No centro de controle, a estratégia de alto nível era decidida e enviada para o Nanook na forma de comandos. Como somente essa estratégia de alto nível era decidida no centro de controle, o que implica que o robô depende de mínimos comandos, ele é considerado semiautônomo.

A Figura 13 apresenta uma visão macro do sistema acima descrito: a interação do Nanook com os robôs auxiliares para reconhecê-los por meio da detecção de esferas e o planejamento da trajetória decidido pelo centro de controle na Terra. Conhecendo a localização dos robôs auxiliares, é possível colar as imagens obtidas, criando uma imagem refinada em 3D. O braço robótico, mostrado no diagrama, acrescentaria a funcionalidade de exploração física do ambiente, mas não foi implementado.

Figura 13 – Diagrama da interação do Nanook com os robôs auxiliares.



Fonte: Elaborado pelo autor a partir de imagens fornecidas por Michael Comberiate.

4.3 Procedimentos Gerais

A metodologia empregada no processo de engenharia reversa seguiu a proposição da seção anterior:

I. Reconhecimento do projeto;
II. Análise do Nanook;
III. Conclusões;
IV. Documentação do Nanook.

Algumas subetapas foram definidas para acompanhar o progresso das etapas do procedimento:

I. Reconhecimento do projeto
    i. Estabelecimento de objetivos;
    ii. Estabelecimento da metodologia de gestão do projeto;
    iii. Busca e compilação das informações disponíveis;

iv. Inspeção visual.

II. Análise do Nanook

i. Análise do hardware do Nanook;

ii. Análise do software do Nanook.

III. Conclusões

IV. Documentação do Nanook

i. Documentação do hardware do Nanook;

ii. Documentação do software do Nanook.

Adicionalmente a esse procedimento, foi adotado uma etapa zero: definição do espaço de trabalho. Uma mesa, alguns monitores, mouses e teclados foram separados em um laboratório da *Capitol Technology University*, onde as demais etapas desse projeto de engenharia reversa foram desenvolvidas.

4.3.1 Reconhecimento do Projeto

Essa etapa inicial é muito importante, uma vez que irá definir as ações posteriores. Uma boa organização das atividades e correto entendimento de todo o trabalho a ser desenvolvido e dificuldades que devem ser superadas fazem uma diferença sensível no desenrolar das atividades.

As primeiras informações do projeto foram fornecidas por Michael Comberiate, quando ele trouxe o Nanook para o projeto de Engenharia Reversa. Suas alegações abrangiam os seguintes problemas relacionados ao robô:

I. Falta de documentação;

II. *Glitch* de inicialização;

III. Problemas na exibição da imagem do LIDAR (não aparecia a imagem de perfil do escaneamento do ambiente);

IV. Falta de algoritmo para evitar obstáculos.

O *glitch* de inicialização, não destacado no escopo desse trabalho, consistia em uma falha apresentada na inicialização do Nanook, que o fazia apresentar um comportamento diferente do esperado. O robô deveria apresentar uma sequência de movimentos iniciais, antes mesmo que qualquer comando fosse enviado, para indicar que todas as suas funcionalidades motoras estavam operacionais. Como, por vezes, esse movimento inicial apresentou um comportamento aleatório, a falha na inicialização foi corrigida para retificar o problema.

As principais preocupações durante a execução do projeto de engenharia reversa eram quanto a uma possível sobrecarga das baterias ou colisão do robô em algum obstáculo.

O carregador da bateria estava superdimensionado, o que propiciava uma sobrecarga. Controlar o robô deveria ser fácil, mas instruções detalhadas sobre como operá-lo não estavam disponíveis. Aprender a parar o robô usando a interface gráfica foi um requerimento para a permissão de fazer testes com o robô no chão.

A partir dessas informações iniciais, traçou-se um plano de projeto, com o estabelecimento de objetivos, gestão do projeto, busca de informações mais detalhadas e inspeção visual, para que houvesse um reconhecimento completo do projeto em execução.

4.3.1.1 Estabelecimento de objetivos

Devido aos esforços da equipe, os seguintes objetivos foram estabelecidos:

I. Engenharia reversa do Nanook para entender seu funcionamento;
II. Documentação do hardware;
III. Documentação do software;
IV. Identificação de problemas e melhorias.

Pelas restrições de tempo, a falta de algoritmo para evitar obstáculos não foi abordada nos objetivos instituídos.

4.3.1.2 Estabelecimento da metodologia de gestão do projeto

Segundo Reis (2014), a gestão de projetos é "[...] a aplicação de técnicas, conhecimento e habilidades para garantir que um projeto tenha sucesso".

Devido a sua importância, algumas tarefas de gestão do projeto foram implementadas, conforme descrito abaixo:

i) Foi criado um grupo conectando as pessoas que estavam trabalhando no projeto;

ii) Todos os dados disponíveis foram organizados no mesmo lugar: uma pasta na nuvem (Google Drive);

iii) Todos os membros tiveram acesso as informações coletadas através do link: <https://drive.google.com/folderview?id=0Bycg1GnzcvjSfndPMEUwRGZrSVFNYk1HWXNUQVFvdVYyT2lGSXB5cTA3Y284RkMxWDN3dUE&usp=sharing>;

iv) Foi definida uma divisão de tarefas e responsabilidades. Apesar de todos

trabalharem no projeto inteiro, a atribuição de lideranças para as tarefas requeridas facilita a gestão do projeto. As atribuições individuais foram: um responsável por liderar a equipe, um responsável por liderar os trabalhos com o software e um responsável por liderar os trabalhos com o hardware.

4.3.1.3 Busca e compilação das informações disponíveis

O Anexo C resume as informações técnicas obtidas. Informações sobre os objetivos do Nanook e notícias antigas sobre ele também foram reunidas, uma vez que poderiam ajudar no entendimento do robô.

Destaca-se um trecho disponível em Stakem (2015), que mostra as razões para o uso da matriz de pontos na formação da imagem do LIDAR. O processo de comunicação entre o robô e o computador de interface com o usuário foi um grande avanço para aquela época:

> *"A comunicação entre o escritório deles [time do Nanook] em Maryland e o robô no Pólo Sul é similar à comunicação com um robô na superfície de Marte. Os engenheiros experimentam os mesmos problemas de sincronização de satélite com grandes volumes de dados a serem enviados e recebidos à medida que o robô captura imagens digitais no formato da matriz de pontos dos objetos que ele encontra, de modo similar ao que seria realizado em transmissões com o equipamento em outro planeta. As imagens são enviadas aos engenheiros, que então decidem quais objetos o robô deve se aproximar e analisar melhor. A matriz de pontos é usada porque é capaz de ser transmitida mais rapidamente que uma imagem de câmera digital. O robô usa um sistema de orientação baseado em laser, conhecido como LADAR (sigla em inglês para 'Laser Detection and Ranging') para encontrar e capturar imagens dos objetos. Ele é semiautônomo e possui capacidade para escaneamento 3D por colagem de imagens".*

Nas informações encontradas discorrendo sobre o Nanook, foram obtidas referências a ambos os termos: LIDAR e LADAR. Embora exista uma sútil diferença no significado da nomenclatura, haja vista que LADAR é a sigla para Detecção e Telemetria de Laser (*Laser Detection And Ranging*) e LIDAR é a sigla para Detecção e Telemetria de Luz

(*Light Detection And Ranging*), as duas expressões podem ser aplicadas referindo-se ao mesmo dispositivo do Nanook.

Também foi feito contato com algumas das pessoas que haviam trabalhado com o Nanook anteriormente. Duas dessas pessoas compartilharam seus conhecimentos sobre o robô com a equipe: Gabriel Trisca e Jaime Ivan Cervante. Suas observações foram que o código interno poderia ser obtido no computador embarcado ao conectar um monitor e um teclado ao robô; contudo, o código era poluído (não conciso, apresentando várias linhas inutilizadas) e desorganizado (escrito em diferentes estilos, sem padronização). Eles também informaram que o Nanook foi um projeto em que times diferentes trabalharam em diferentes anos, o que tornaria seu entendimento mais complexo. Em diferentes épocas, novas funcionalidades foram adicionadas e outras desabilitadas sem que nenhuma das mudanças realizadas fossem documentadas.

Quanto ao escaneamento 3D, certificou-se que essa funcionalidade havia sido realmente implementada por completo e, portanto, deveria funcionar corretamente. Foi informado que o código havia sido modificado para que a imagem não aparecesse automaticamente, então dever-se-ia descobrir um botão que habilitasse a visualização da imagem.

4.3.1.4 Inspeção Visual

Depois da pesquisa sobre o Nanook, foi realizada uma inspeção visual para coleta de informações adicionais. A Figura 14 apresenta a versão atual do Nanook, na qual o presente trabalho foi baseado.

As informações físicas do Nanook foram obtidas e são mostradas na Tabela 4 a seguir:

Tabela 4 - Informações físicas do Nanook

Informação Física	Medida
Peso	66,8 kg
Dimensões	Aproximadamente 51x70x60 cm

Fonte: Elaborado pelo autor

Figura 14 - Nanook atualmente.

Fonte: Acervo do Autor.

Além disso, foram obtidas informações sobre os componentes do robô, bem como possíveis problemas foram identificados e examinados.

4.3.1.4.1 Componentes Identificados

A Tabela 5 a seguir lista todos os componentes identificados no circuito do robô. Os *datasheets* podem ser obtidos no link de referência indicado na tabela. A sigla CC foi utilizada referindo-se à corrente contínua, UART para Receptor/Transmissor Universal Assíncrono (*Universal Asynchrounous Receiver/Transmiter*) e USB para porta universal (*Universal Serial Bus*).

Tabela 5 - Informação sobre os componentes do robô identificados na inspeção visual.

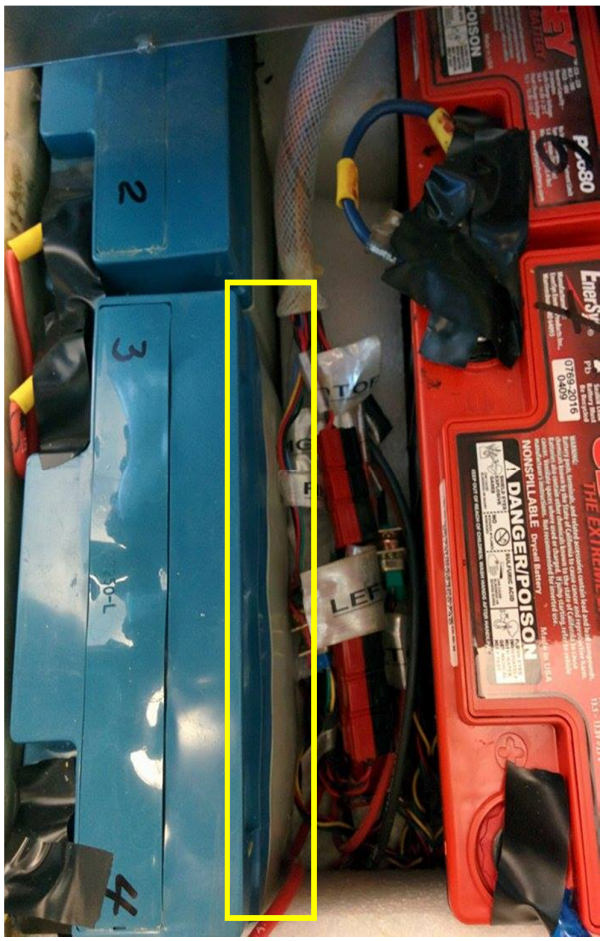
Componente	Breve Descrição	Informação Adicional	Link de Referência
Conversor CC/CC Isolado 20W 24V para 5V 4.0A	Conversor para 5V	Sistema de Conversão de Potência	http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf
Conversor CC/CC Isolado 20W 24V para 12V 1.66A	Conversor para 12V	Sistema de Conversão de Potência	http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf
CP2102	Ponte USB para UART	Sistema dos Motores CC	https://cdn.sparkfun.com/datasheets/BreakoutBoards/CP2102_v1.2.pdf
Driver CP210x	Driver VCP (interface de comunicação virtual) de ponte USB para UART	Sistema dos Motores CC	http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
SABERTOOTH 2X25	Driver do Motor	Sistema dos Motores CC	https://www.dimensionengineering.com/datasheets/Sabertooth2x25.pdf
AD4-B-S	Adaptador de Quadratura para RS232	Sistema dos Motores CC	http://www.digchip.com/datasheets/parts/datasheet/502/AD4-B-pdf.php
Controlador R256	Driver do Motor de Passo	Sistema do Motor de Passo do LIDAR	http://www.linengineering.com/drivers-Controllers/PDF/R256_Manual1.08.pdf
Modelo Intel D945GCLF2	Placa Mãe	Circuito Geral	http://downloadmirror.intel.com/16960/eng/D945GCLF2_TechProdSpec02.pdf
Kingston V125-S2	Dispositivo de armazenamento flash	Circuito Geral	http://www.kingston.com/datasheets/SNV125-S2_us.pdf
SICK LMS 221-30206	LIDAR	Circuito do LIDAR	http://sicktoolbox.sourceforge.net/docs/sick-lms-technical-description.pdf

Fonte: Elaborado pelo autor.

4.3.1.4.2 Informação adicional sobre as baterias

Durante a inspeção visual, foi observado que uma das baterias (a bateria azul, localizada no lado esquerdo do robô, identificada pelos números 3 e 4) estava danificada, conforme pode ser percebido na Figura 15. O inchamento dessa bateria pode ter sido ocasionado por uma sobrecarga prévia. Recomendou-se a substituição da bateria o mais brevemente possível, ou pelo menos um monitoramento ativo do seu estado.

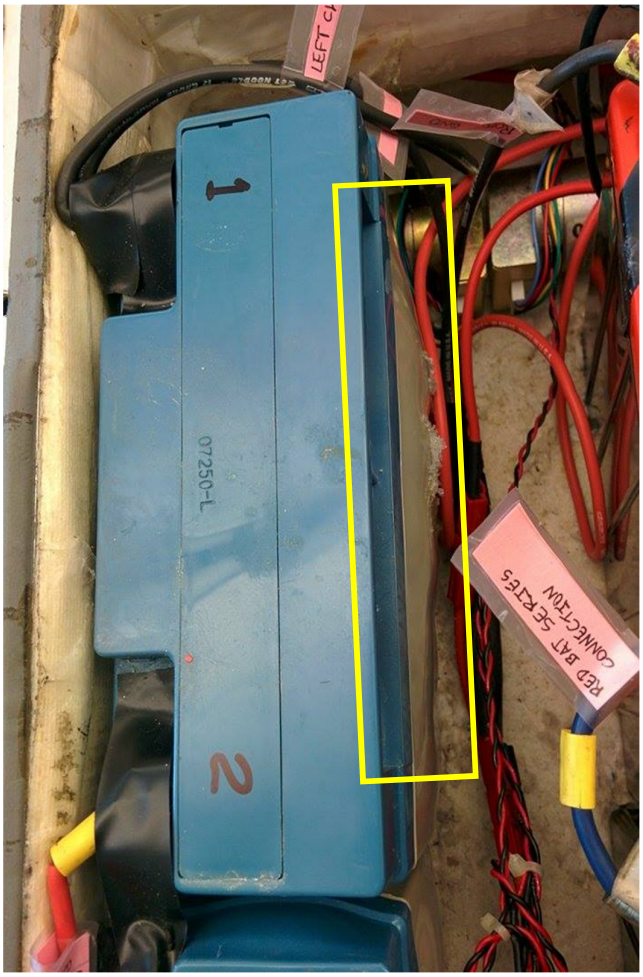
Figura 15 - Bateria azul inchada.



Fonte: Acervo do autor.

Em uma inspeção posterior mais detalhada do hardware, foi também observado que a outra bateria azul (localizada na parte da frente do robô, identificada pelos números 1 e 2) também estava inchada, conforme evidenciado na Figura 16 a seguir.

Figura 16 - Outra bateria azul inchada.



Fonte: Acervo do autor.

4.3.2 Análise do Nanook

Após a visão geral do projeto ter sido estabelecida, o robô foi examinado mais detalhadamente para que tanto seu hardware quanto seu software fossem bem compreendidos. Apesar de as análises terem sido feitas separadamente (hardware versus software), uma servia para complementar o entendimento da outra e por vezes essa estratégia foi utilizada. Por exemplo, quando não se sabia como ou em que momento determinado componente era utilizado no robô, fazia-se uma rápida análise do software para compreensão do sistema. O inverso também ocorreu, de modo que por vezes o robô foi analisado sem a divisão entre hardware e software, mas como um todo. Apesar dessas situações, didaticamente, optou-se por explorar as análises separadamente aqui: análise do hardware e análise do software do Nanook.

4.3.2.1 Análise do Hardware do Nanook

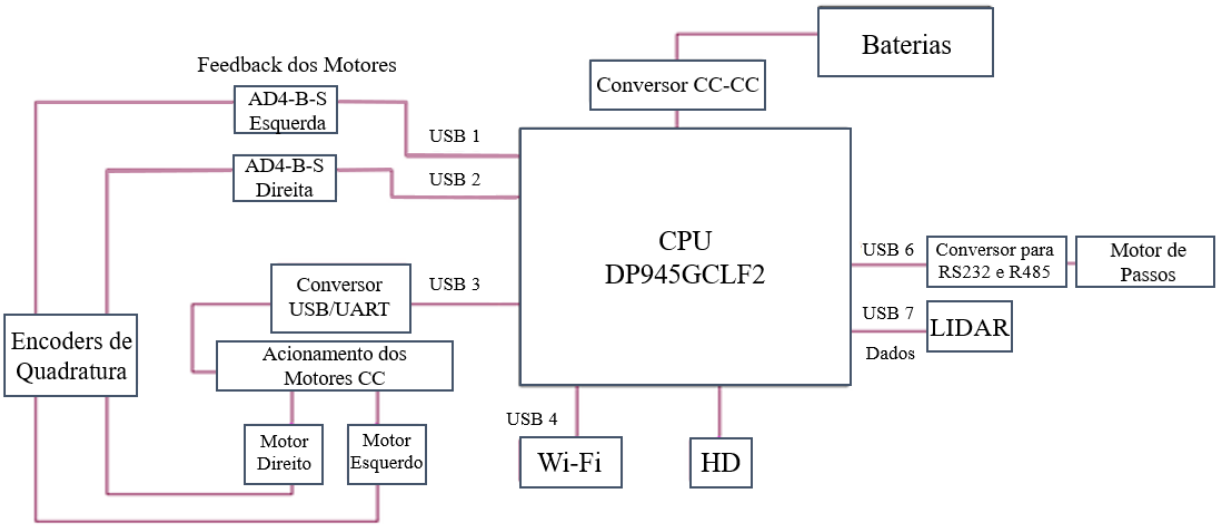
Para compreensão do hardware do Nanook, foram realizadas duas inspeções no hardware. Uma preliminar, que possibilitou o entendimento dos circuitos existentes e a lógica por trás deles, e uma mais detalhada para esclarecer algumas informações específicas que não ficaram evidentes na inspeção preliminar. Após a compreensão do sistema de hardware do robô, algumas medidas de organização e documentação parcial foram adotadas para viabilizar as etapas subsequentes da engenharia reversa do Nanook.

4.3.2.1.1 Inspeção do Hardware

Conforme exposto, uma primeira inspeção foi feita para entender os circuitos eletrônicos do robô. As observações dessa primeira análise são descritas nas próximas páginas, seguindo a ordem em que elas foram sendo obtidas.

A análise iniciou-se pela placa mãe, com a observação das conexões USB a ela interligadas. Foram identificadas um total de seis conexões USB sendo utilizadas na placa mãe, evidenciadas na Figura 17.

Figura 17 - Sistema em blocos do Nanook.



Fonte: Elaborado pelo autor.

As conexões identificadas como 1 e 2 são o sinal de feedback dos motores (parte do

sistema de controle dos motores), que passam anteriormente pelo conversor de quadratura para RS232, o AD4-B-S. A presença desse conversor trouxe uma suspeita da presença de *encoders* na caixa até então inacessível, na qual os motores estão. Uma terceira conexão USB (identificada como USB 3) passava por um conversor USB/UART que envia sinais para o driver do motor (Sabertooth). O driver do motor (Sabertooth) envia o sinal de acionamento para os motores do lado direito e esquerdo do robô. A conexão USB identificada pelo número 6 envia o sinal de controle da placa mãe para o motor de passo. Esses sinais passam pelo driver de motor R256 antes de realmente acionarem o motor que controla a extensão (vertical) da inspeção do LIDAR. Foi descoberto que o controlador *Parallax* não está sendo utilizado e por isso está desconectado, uma vez que o motor de passo recebe sinais diretamente da placa mãe através da conexão de número 6. Por fim, a conexão de número 7 recebe dados do LIDAR. A placa mãe também está conectada a um disco rígido externo e a uma fonte de alimentação. Adicionalmente, entendeu-se que o nível de tensão da bateria é ajustado pelos conversores CC-CC (para 12 V e 5 V) antes de alimentar os circuitos principais. A placa mãe é alimentada por um conector que possui os níveis de tensão ajustados por outro conversor, que tem 12 V como entrada.

A placa mãe tem oito conexões USB no total, sendo que duas delas não estão sendo usadas permanentemente (somente estão sendo usadas as seis já descritas). Poder-se-ia, portanto, conectar um pen drive ao sistema, caso necessário, usando uma dessas entradas USB. As suas identificações, relacionando-as a números de um a oito, estavam um pouco confusas, visto que a conexão USB poderia ter dois números diferentes (por exemplo, 1 e 3, conforme mostrado na Figura 18), sendo que nenhuma explicação sobre o porquê de eles serem diferentes e qual o significado desses números foi encontrada.

Cada um dos USBs da placa mãe foi identificado, a partir das observações das conexões a eles associadas, conforme segue:

i) USB de cabos na própria placa mãe:

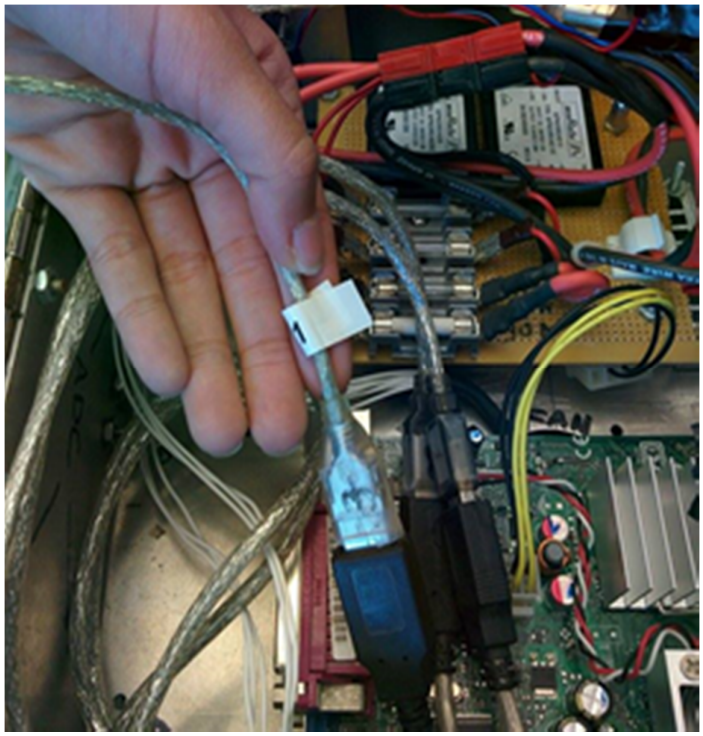
- USB 1 do cabo “*Right*” (ligado aos conectores VGA sem identificação): Conecta-se ao AD4-B-S da esquerda;
- USB 2 do cabo sem identificação ligado aos conectores VGA identificados como 1E: Conecta-se ao AD4-B-S da direita;
- USB 3 do cabo “1”: Conecta-se ao CP2102 (conversor USB/UART), que se conecta ao SABERTOOTH 2X25 (*driver* do motor);

- USB 4 (sem identificação atual) do cabo sem identificação: Conecta-se ao adaptador de Wi-Fi.

ii) USB de conexões na placa mãe:

- USB 6: Conexão para o motor de passo do LIDAR (conector branco): Essa conexão da placa mãe liga-se sucessivamente a: conversor RS232, conversor RS232/R485, R256 e motor de passos.
- USB 7: Conexão para trocar de dados com o LIDAR (conector verde): Essa conexão da placa mãe liga-se diretamente ao LIDAR e é responsável por receber os dados da imagem obtida pelo LIDAR e enviar tanto as configurações da transmissão desses dados, como o requerimento do envio da imagem.

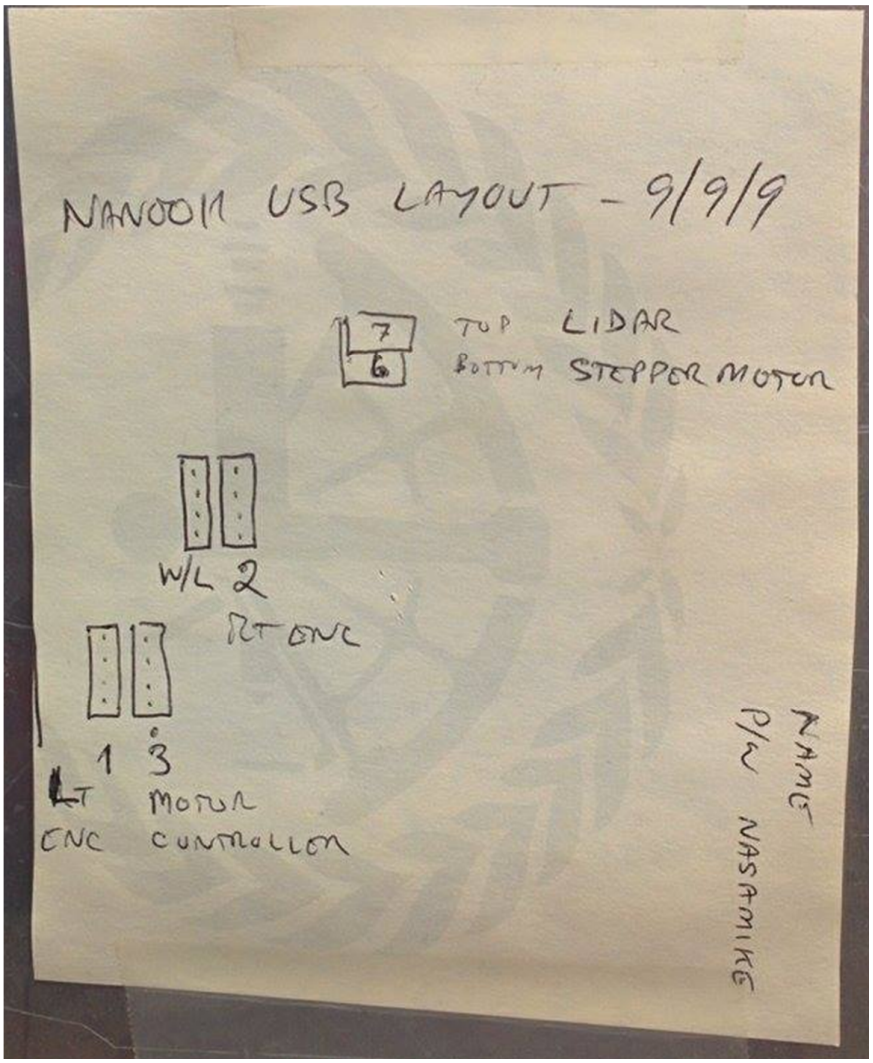
Figura 18 - Exemplo de conexão USB identificada com números diferentes.



Fonte: Acervo do autor.

Essas conclusões foram corroboradas a partir da ilustração do *layout* dos USBs, presente na tampa da caixa cinza do robô, já que ambos eram compatíveis. Essa ilustração está exposta na Figura 19.

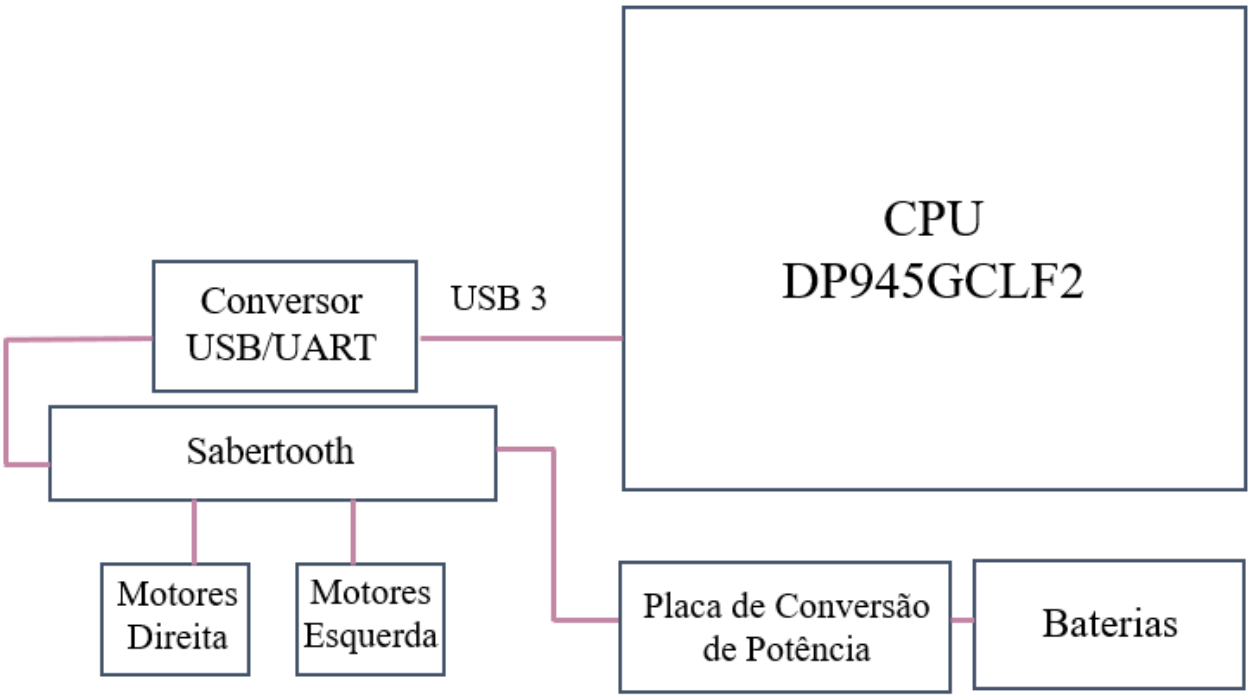
Figura 19 - Anotação previamente disponível das conexões dos USBs.



Fonte: Acervo do autor.

O Sabertooth está conectado (fios verde e azul) ao conversor CP2102 (um conversor USB/UART) e é alimentado pela placa de conversão de potência, conforme explicitado na Figura 20. A saída do Sabertooth é composta por dois pares de fios vermelho e preto que estão soldados em um conector na borda da caixa cinza principal (no canto direito) e então são ligados aos motores. Cada par conecta um motor de determinado lado (esquerda ou direita).

Figura 20 – Ligações dos componentes de acionamento dos motores CC.



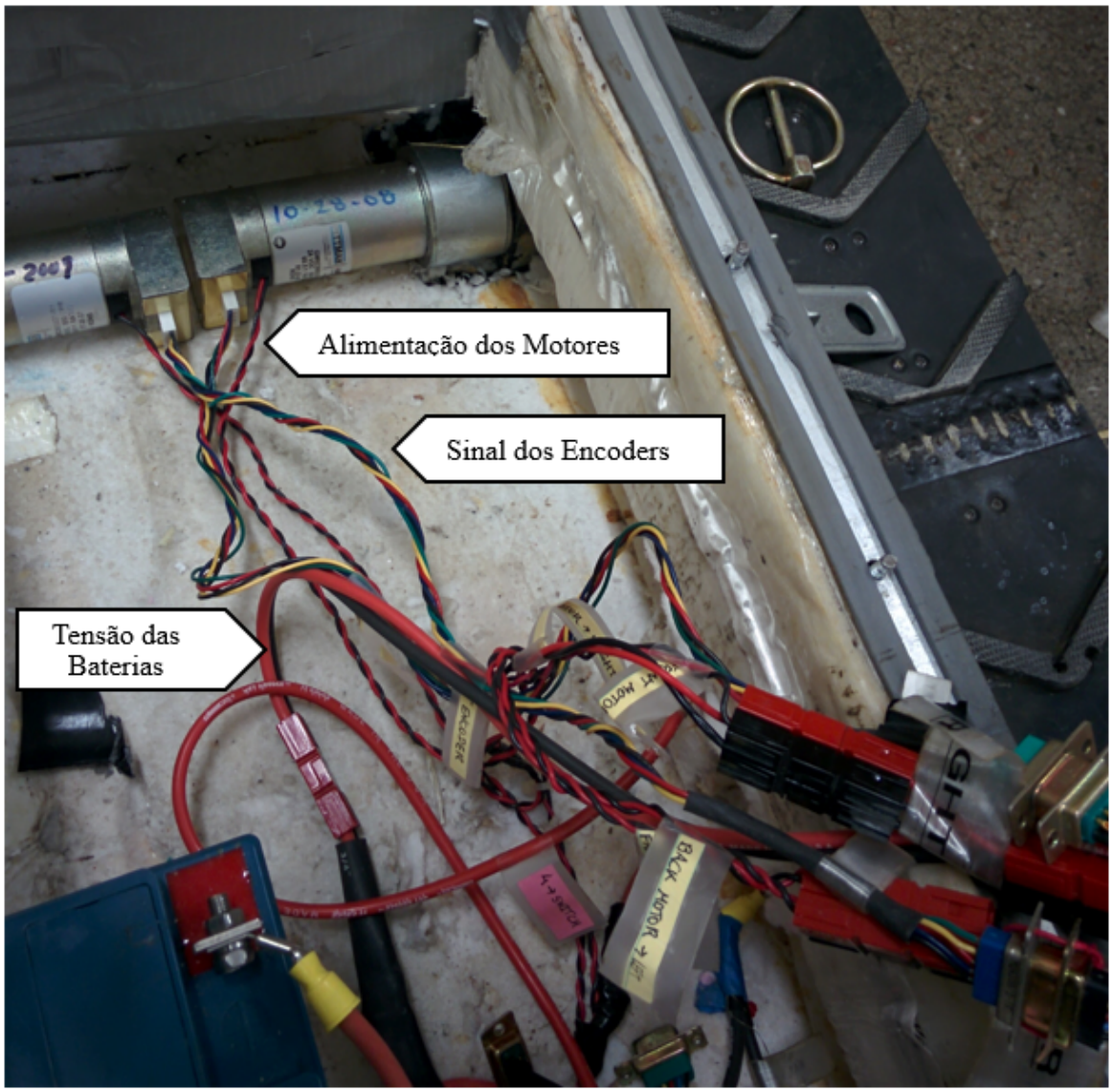
Fonte: Elaborado pelo autor.

Embaixo da caixa do circuito principal, identificou-se a alimentação (neutro - cabo preto e VCC (tensão em corrente contínua) - cabo vermelho) para os motores do lado direito e esquerdo (para cada motor, um par de cabos) e um conjunto de cinco cabos menores (amarelo, azul, verde, vermelho e preto) para cada motor.

Os fios de alimentação (cabos grossos vermelho e azul) conectam-se ao sistema de conversão de potência.

Os cabos acima descritos estão evidenciados na Figura 21 abaixo.

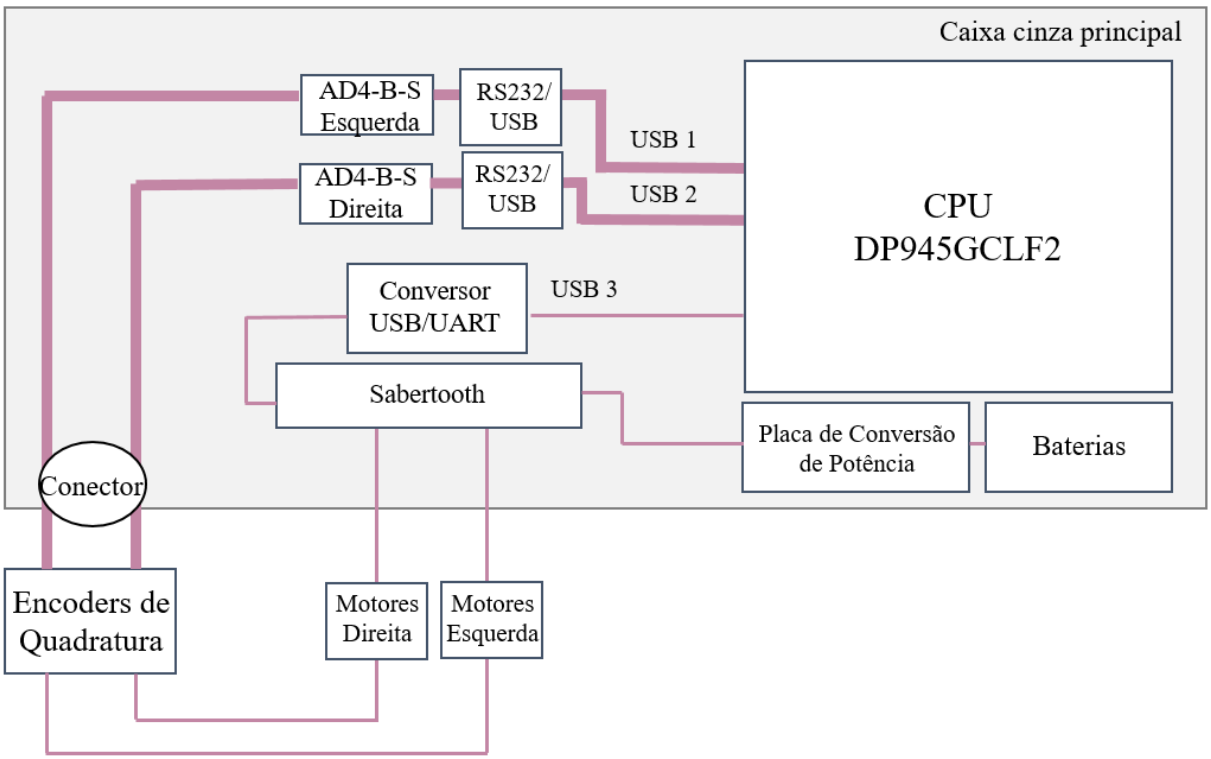
Figura 21 – Fios identificados embaixo da caixa do circuito principal.



Fonte: Elaborado pelo autor.

Esses dois conjuntos de cabos finos coloridos fazem a conexão dos *encoders* de quadratura ligados aos motores CC e os respectivos AD4-B-S dos motores, passando por um conector MS3106A28-21S localizado na borda da caixa cinza, no lado direito. Essa ligação está destacada na Figura 22 a seguir. O *encoder* de quadratura captura a posição de fase dos motores CC nos canais A e B e então essa informação é processada no AD4-B-S, no qual é convertido em comunicação serial (RS232).

Figura 22 – Conexão dos *encoders* ao AD4-B-S.

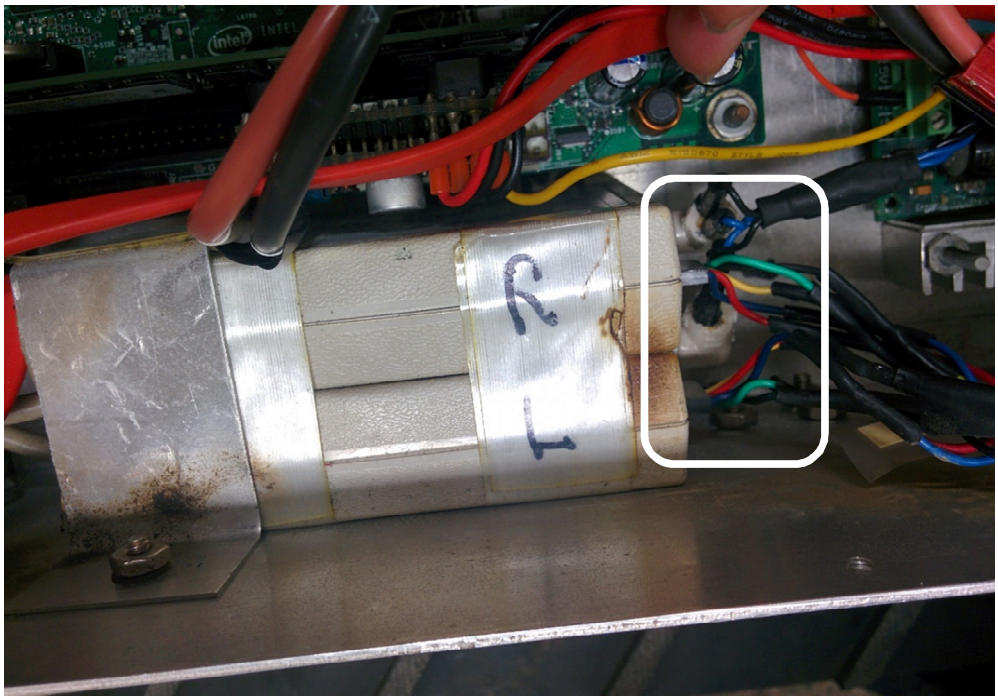


Fonte: Elaborado pelo autor.

A existência de *encoders* nos motores de corrente contínua foi uma conclusão advinda da observação das conexões vindas da caixa preta do motor (deveriam ser de um *encoder*). A caixa, até então inacessível, do motor é conectada ao AD4-B-S, um conversor de quadratura para RS232, antes de ser conectada à placa mãe. Analisando o *datasheet* do AD4-B-S, foi descoberto que seu sinal de entrada deveria ser de um *encoder* de quadratura.

Cada um dos conjuntos de fios finos conecta-se a um AD4-B-S diferente, conforme Figura 23. Um conjunto está ligado ao AD4-B-S correspondente ao motor esquerdo (AD4-B-S-E) e o outro ao AD4-B-S correspondente ao motor da direita (AD4-B-S-D), etiquetados, respectivamente, como L (do inglês *left*, para esquerda) e R (do inglês *right*, para direita).

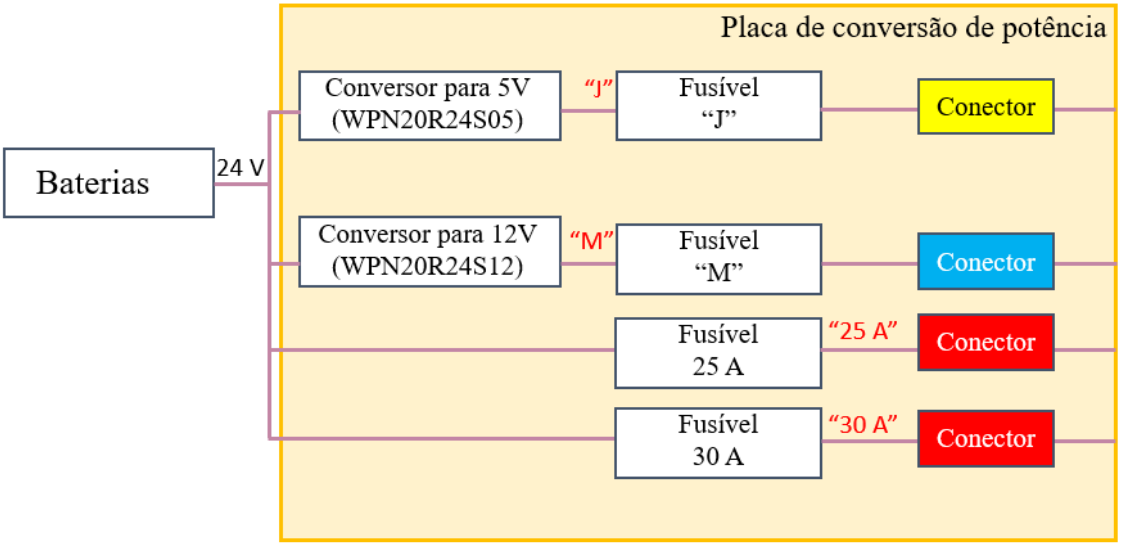
Figura 23 – Cada AD4-B-S recebe um conjunto de fios finos coloridos com as informações de feedback dos encoders.



Fonte: Acervo do autor.

Na placa de conversão de potência, mostrado no diagrama de blocos da Figura 24, o nível de tensão da bateria é convertido para 12 V e 5 V no WPN20R24S12 e WPN20R24S05, respectivamente. Esses componentes estão atualmente obsoletos.

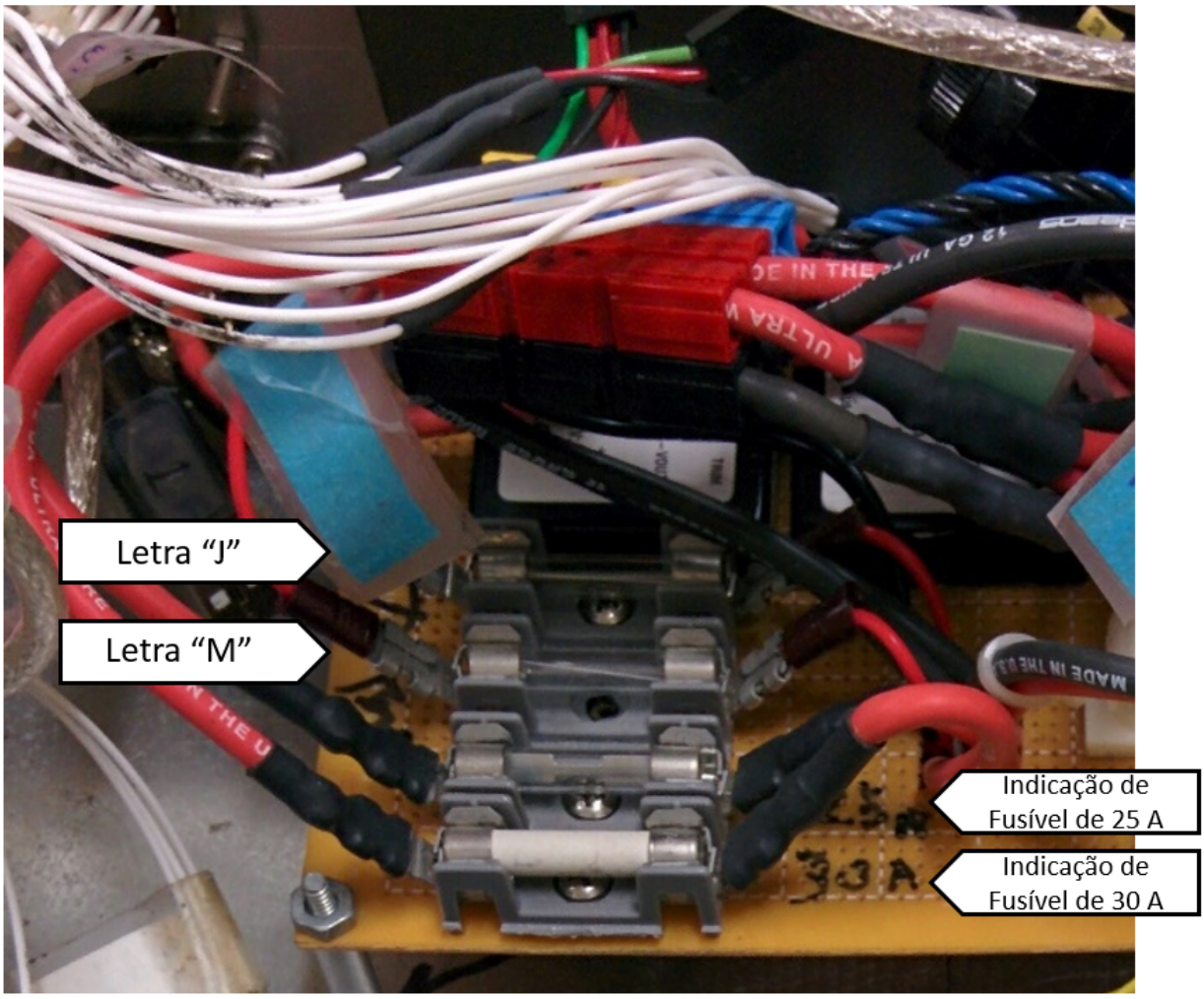
Figura 24 - Diagrama em blocos da placa de conversão de potência.



Fonte: Elaborado pelo autor.

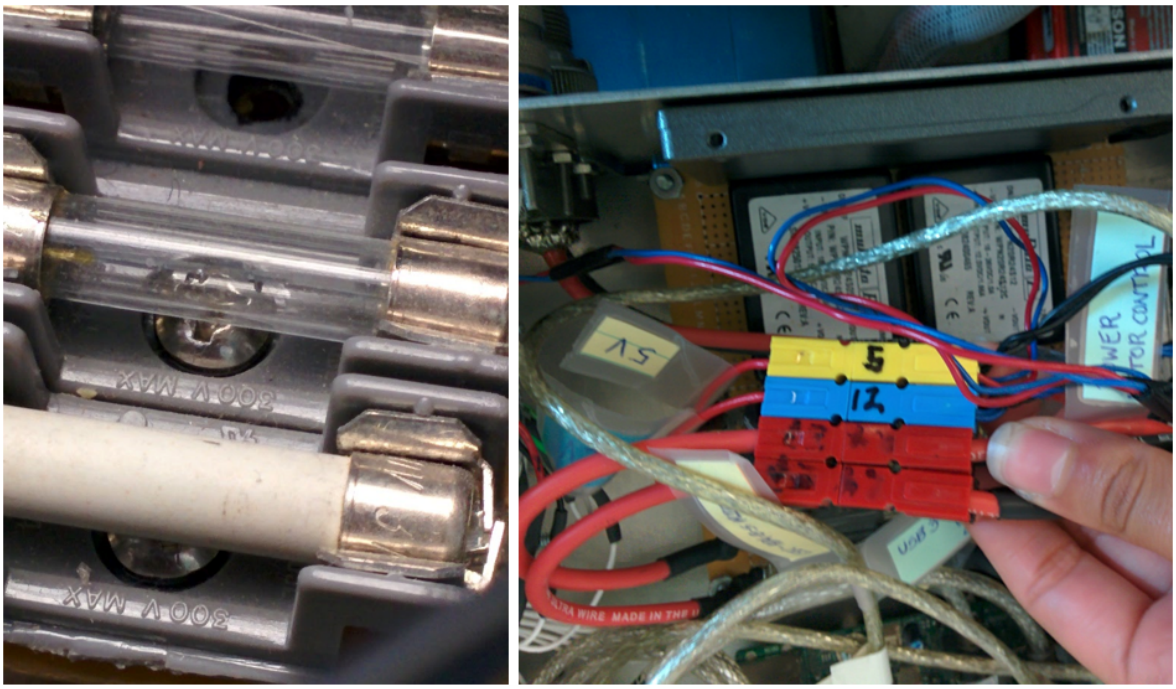
Após o conversor, o fio da fase passa por um fusível. Existe um para a tensão de 5 V e outro para a de 12 V. A tensão de 5 V está identificada na placa com a letra “J” e passa por um conector amarelo marcado com “5”. A tensão de 12 V é identificada na placa com a letra “M” e conecta-se ao conector azul marcado com “12”. Existem ainda dois fusíveis (um de 25 A e um de 30 A) que recebem alimentação. A saída deles passa por conectores vermelhos. A Figura 25 destaca a identificação dos fusíveis acima exposta. A Figura 26 mostra os fusíveis em detalhe e destaca os conectores descritos.

Figura 25 - Identificação dos fusíveis.



Fonte: Acervo do autor.

Figura 26 – Detalhe dos fusíveis, utilizados para proteção contra sobretensão e dos conectores dispostos nas saídas destes.



Fonte: Acervo do autor.

A partir do conector amarelo, cuja saída é 5 V, existem dois fios, um preto e um vermelho (ambos de 5 V, mesmo o fio preto). Os fios que são a saída dos conectores da placa de conversão de potência estão destacados na Figura 27. Juntamente com o fio preto, um outro fio preto (dessa vez, neutro) conecta-se a um conector intitulado "*for hub*".

A escolha das cores dos fios não foi a mais adequada, devido a confusão gerada por não estar em conformidade com o padrão de cores usualmente empregado por engenheiros eletricistas.

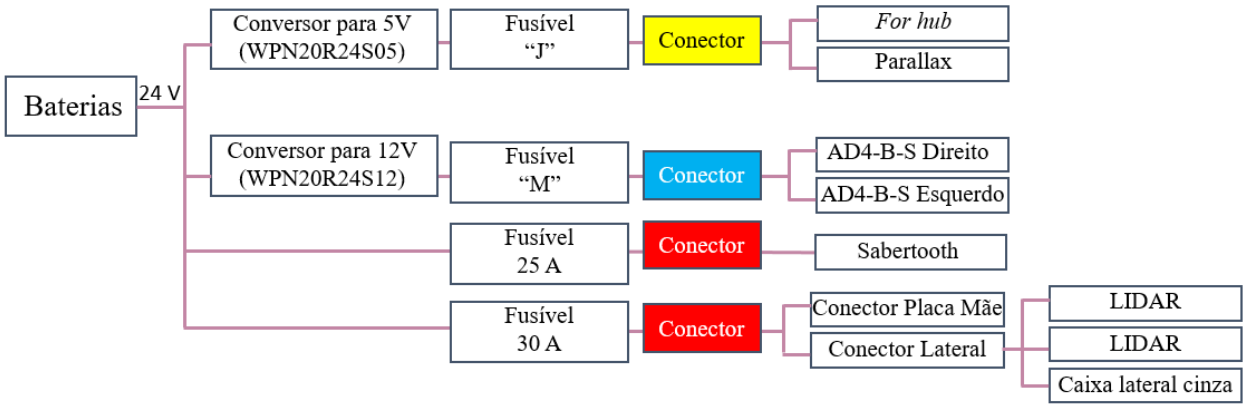
O fio vermelho (5 V) juntamente com um outro fio preto (neutro) liga-se ao *Parallax*. Do *Parallax*, seis fios de saída não estão sendo usados. Os sinais de controle para o motor de passo do LIDAR têm sua origem na placa mãe (USB 6), o que corrobora a não utilização do *Parallax.*

Figura 27 – Destaque para os fios de saída dos conectores da placa de conversão de potência.

Fonte: Acervo do autor.

A partir do conector azul (portanto, alimentação de 12 V), dois fios azuis conectam-se ao AD4-B-S. Um fio azul e um fio preto (vindo do conector preto - neutro) conectam-se ao AD4-B-S-E (AD4-B-S correspondente ao motor esquerdo), e outro par desses fios conectam-se ao AD4-B-S-D (AD4-B-S correspondente ao motor direito). A Figura 28 ilustra o destino das conexões dos fios que passam pelos conectores da placa de conversão de potência.

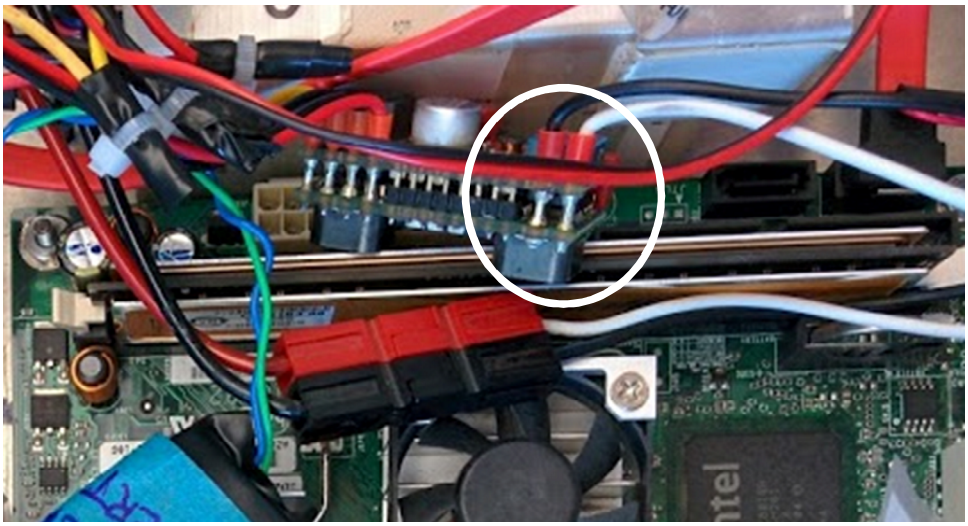
Figura 28 - Diagrama em blocos com as ligações após os conectores da placa de conversão de potência.



Fonte: Elaborado pelo autor.

A fase que passa pelo fusível de 25 A, passa também por um conector vermelho e então é conectada ao Sabertooth. A fase que passa pelo fusível de 30 A, também passa por um conector vermelho, a partir do qual passam a existir dois fios vermelhos (um grosso e um fino). O fio grosso é conectado em um outro conector na borda da caixa cinza. Desse conector, o fio vermelho se divide em três outros fios vermelhos. Dois deles são conectados ao LIDAR, enquanto o terceiro é conectado à caixa lateral cinza. O fio fino, saída do conector depois do fusível de 30 A, está ligado a um outro conector, onde se torna um fio branco conectado à alimentação ATX (acrônimo para *Advanced Technology Extended)* CC-CC, conforme mostrado na Figura 29. Essa alimentação é conectada a placa mãe por um conector ATX de 20 pinos. Quatro fios saem do conversor: um fio amarelo (+12 V), um fio vermelho (+5 V) e dois fios pretos (neutro). Esses fios e alguns outros vindo do HD (*hard disk*) externo são conectados à placa mãe por uma conexão macho-fêmea (4 pinos Molex).

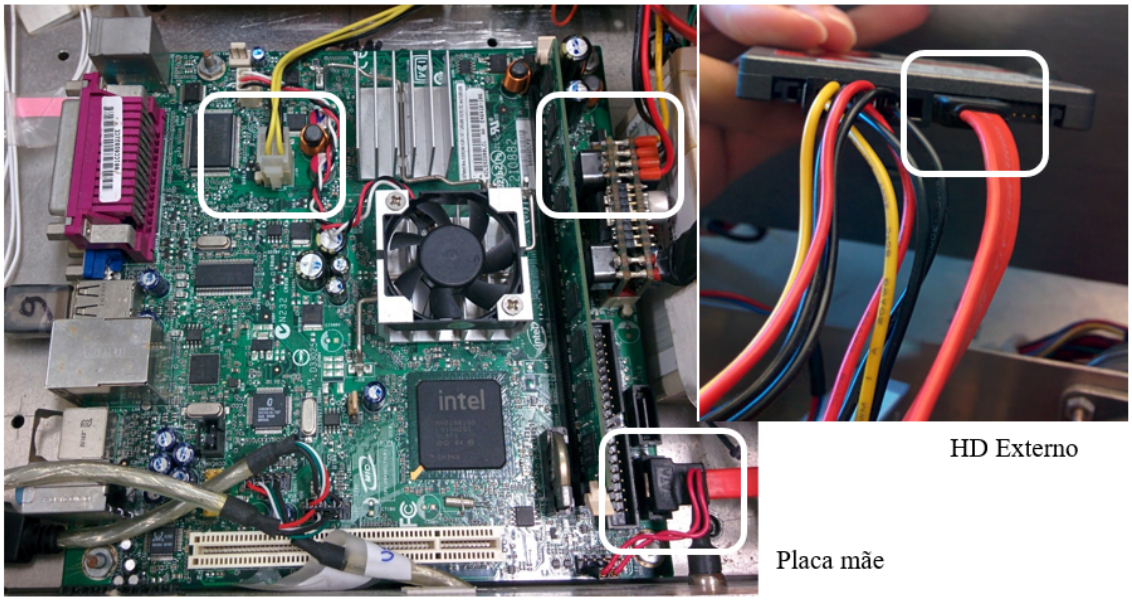
Figura 29 - Ligação da alimentação na placa mãe (conector ATX CC-CC).



Fonte: Acervo do autor.

Existe uma conexão de dados do tipo SATA entre o HD externo e a placa mãe (SATA0), conforme Figura 30. Além disso, há uma conexão de alimentação SATA entre o conector da alimentação principal e a placa mãe. Esse plugue também está conectado à alimentação 2x2 da placa mãe – somente dois GND e +12VCC (fio amarelo). O modelo da unidade de armazenamento flash SSD (*solid state drive*) é o SNV125-S2/64GB.

Figura 30 – Destaque para conexão SATA entre a placa mãe e o HD externo, alimentação 2x2 na placa mãe e saída da alimentação ATX CC-CC.



Fonte: Acervo do autor.

O botão externo que aciona os circuitos (botão branco) também está conectado à placa mãe.

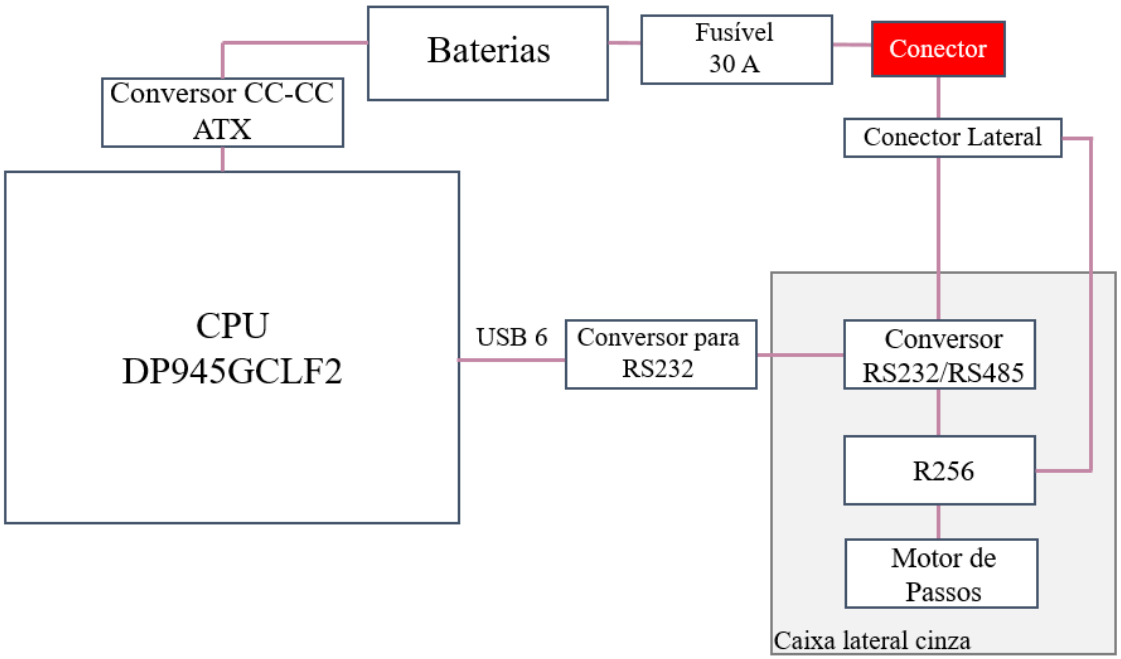
Figura 31 – Botão de inicialização da placa mãe.

Fonte: Acervo do autor.

Na caixa cinza lateral, há o circuito de controle do LIDAR, com um controlador de micro passos R256, um kit de comunicação RS232 e o próprio motor de passos controlado por eles, evidenciados na Figura 32. Primeiramente, a CPU envia os sinais a partir do USB 6, que é convertido para comunicação RS232. Esses sinais passam pelo conversor de RS232 para RS485 na caixa lateral. O conversor RS232-RS485 está conectado a CPU usando um cabo

padrão macho-fêmea DB-9. As saídas do conversor RS232-RS485 são uma fase A e uma fase B. Uma alimentação de +12 VCC é suprida para ambos, o conversor e o R256.

Figura 32 – Diagrama em blocos do circuito de controle vertical do LIDAR.



Fonte: Elaborado pelo autor.

Os sinais de saída estão conforme o padrão de comunicação RS485 e destinam-se ao controlador de passos R256. A saída do R256 consiste em quatro sinais que acionam o motor de passo. O fio vermelho é A, o azul é $\bar{A}$, o verde é B e o preto é $\bar{B}$.

4.3.2.1.2 Inspeção Detalhada Do Hardware

Como algumas informações não estavam bem esclarecidas quando a primeira inspeção foi concluída, inspeções adicionais foram realizadas para complementar o conhecimento necessário.

Detalhes adicionais sobre o sistema de potência e os *encoders* dos motores foram descobertos. Como essa parte não era acessível anteriormente, foi necessária a remoção de algumas partes do robô (o sistema do LIDAR, a caixa da CPU e também a isolação das baterias), de modo a possibilitar o acesso a caixa que continha as baterias e os motores.

Figura 33 - Caixa interna que continha baterias, motores e *encoders*.

Fonte: Acervo do autor.

Com esse acesso, visualizou-se quatro motores, quatro *encoders* e quatro baterias. Além disso, os fios conectados à bateria e aos motores foram rastreados. As baterias são conectadas a uma chave, de modo que o sistema possa alternar entre estar conectado ao robô ou ao carregador. Dois conjuntos de fios coloridos (preto, azul, vermelho, amarelo e verde) estão conectados nos *encoders* (cada lado, com um *encoder* e seu respectivo conjunto de fios). Existem dois *encoders* na parte frontal do robô, mas que não estão sendo usados no momento. Todos os motores recebem alimentação (VCC e neutro). Cada conjunto de baterias (duas de cada lado) é conectado em paralelo, enquanto cada conjunto é internamente conectado em série provendo 24 V cada. A alimentação (VCC e neutro) dos circuitos da caixa cinza principal é constituída por um fio vermelho e um fio preto. Esses fios alimentam a placa de conversão de potência, onde existem dois conversores: um conversor de 24 V para 12 V e um de 24 V para 5 V. Dois fios vermelhos são conectados à placa de modo que também estão conectados à tensão de entrada. A saída da placa são dois fios com tensão de 24 V, um fio com tensão de 12 V e um fio com tensão de 5 V. O fio com nível de tensão de 12 V é a entrada de um conector azul, o fio de tensão de 5 V é a entrada de um conector amarelo e os fios de 24 V são as entradas dos conectores vermelhos.

Um dos conectores na borda da caixa cinza principal foi identificado como sendo do modelo Bendix LJTPQ00RT-15-35S.

4.3.2.1.3 Inspeção das Baterias

No Nanook, há quatro baterias, duas do modelo PSH-12180FR do fabricante PowerSonic (referidas como baterias azuis) e duas do modelo PC680 do fabricante Odyssey (referidas como baterias vermelhas). As baterias azuis têm tensão nominal de 12 V, capacidade nominal de 21 $A \cdot h$, pesam aproximadamente 6 kg cada e são de chumbo-ácido, desenvolvidas especificamente para aplicações de rápida taxa de descarga (corrente máxima de 63 A por 7 min ou 210 A por 10 s) (POWER SONIC, n.d.). As baterias vermelhas têm tensão nominal de 12 V, capacidade nominal de 16 $A \cdot h$ e pesam aproximadamente 7 kg cada (WEST COAST BATTERIES, n.d.).

Como foi observado que as baterias azuis estavam inchadas, foi realizada uma investigação mais detalhada sobre esse problema.

Inicialmente, ambos os lados apresentaram nível de tensão similar quando medidos. Contudo, enquanto o robô era utilizado, percebeu-se que as baterias vermelhas descarregavam mais rapidamente que as baterias azuis, o que foi comprovado pelos níveis de tensão medidos após a utilização do robô, conforme exposto na Tabela 6.

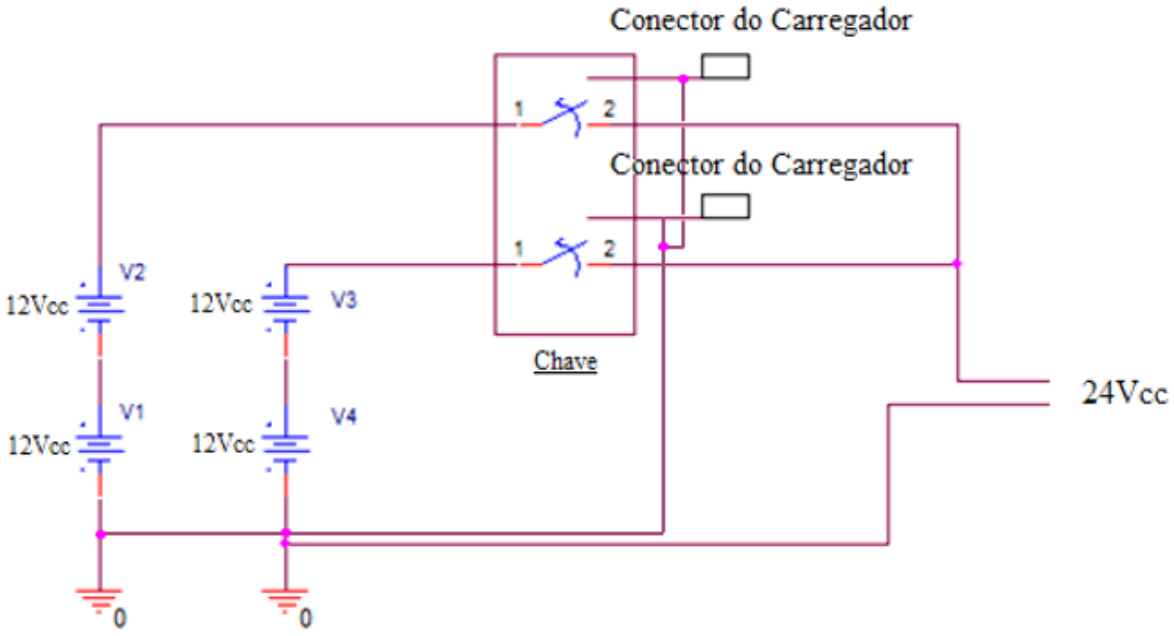
Tabela 6 - Nível de tensão por bateria após utilização do robô

Número de Identificação da Bateria	Números das Conexões da Bateria	Cor da Bateria	Tensão (V)
1	1-2	Azul	13.01
2	3-4	Azul	12.93
3	5-6	Vermelha	7.75
4	7-8	Vermelha	8.68

Fonte: Elaborado pelo autor.

As baterias azuis, do modelo PSH-12180FR do fabricante PowerSonic, são de uma marca superior (considerando as características gerais de capacidade nominal) à das vermelhas, do modelo PC680 do fabricante Odyssey, o que pode explicar o decaimento acelerado da tensão nas baterias vermelhas. Como exposto, a capacidade nominal das baterias azuis para 20h é de 21 Ah enquanto a das baterias vermelhas é de 16 Ah (POWER SONIC, n.d. e WEST COAST BATTERIES, n.d.). Por isso, enquanto as baterias azuis podiam manter a tensão adequada por mais tempo, as baterias vermelhas descarregavam mais rapidamente. Como ambos conjuntos de bateria (duas azuis e duas vermelhas) estavam conectados em paralelo, conforme esquematizado na Figura 34, o robô não conseguia funcionar corretamente quando as baterias vermelhas estavam descarregadas. O procedimento normal nesse caso seria carregar as baterias, o que sobrecarregava as baterias azuis. A sobrecarga dessas baterias pode explicar o porquê elas estavam inchadas.

Figura 34 - Esquemático do circuito das baterias.



Fonte: Elaborado pelo autor.

Recomendou-se substituir todas as baterias por novas baterias (todas do mesmo modelo). Fazendo isso, otimiza-se a chance de as baterias manterem a mesma tensão, evitando sua sobrecarga. Os modelos presentes no Nanook, PSH-12180FR do fabricante PowerSonic e PC680 do fabricante Odyssey, anteriormente citados, estão ilustrados na Figura 35.

Durante a execução do projeto, todavia, não havia quatro baterias azuis disponíveis (as da melhor marca), então estas foram substituídas por baterias vermelhas. Assim, o robô ficou com quatro baterias vermelhas idênticas. Entretanto, as duas baterias colocadas no robô não eram modelos novos e foi constatado que elas não estavam funcionando adequadamente.

Figura 35 - Ilustração do modelo das baterias: azul e vermelha.

Fonte: Power Sonic (n.d.) e Ballade Sports (n.d.).

Para atestar o mau funcionamento das baterias vermelhas adicionais, houve uma tentativa de trabalhar com o robô após carregar as baterias. Primeiro, uma das baterias não carregava mais que 11V. Quando o robô era ligado, a rotina de inicialização não era adequada. Somente após 30 minutos de tentativas, o robô foi capaz de funcionar corretamente. Como as baterias são mais eficientes quando estão aquecidas e a princípio elas não estavam funcionando bem, conclui-se que a temperatura as afetou a ponto de o robô nem mesmo funcionar por certo tempo.

A decisão final foi remover as baterias que não funcionavam corretamente (as vermelhas adicionais) e substituí-las por baterias cinzas, de modelo não identificado, que estavam disponíveis em estoque. Para não repetir o mesmo problema de antes (baterias diferentes em paralelo), uma bateria vermelha e uma bateria cinza foram colocadas em série em cada um dos lados.

4.3.2.2 Análise do Software do Nanook

Haviam dois códigos a serem analisados: o código embarcado e o de um programa de interface com o usuário. Esse programa estava no notebook disponibilizado, que era o utilizado no controle do Nanook. Como o Nanook é um robô semiautônomo, as decisões de mais alto nível eram enviadas por esse notebook a partir das interações do usuário com o programa citado.

4.3.2.2.1 Software de Interface com o Usuário

No notebook fornecido, havia uma pasta com informações do Nanook e versões antigas do código. Foi então identificado que o arquivo executável disponível na área de trabalho desse notebook não era a versão mais recente do código, de modo que um executável da versão adequada foi criado para corrigir essa falha. Isso possibilitou que a versão mais atual do código fosse a versão unicamente analisada pela equipe.

O software de interface do usuário foi escrito em C# usando o Visual Studio 2008. A principal dificuldade, a princípio, foi aprender a linguagem de programação usada, de modo a se estar apto a entender o código. Uma fonte de ajuda sobre C# foi criada concomitantemente a leitura do código, de modo a ajudar todos os integrantes da equipe a entender os arquivos. Todos os arquivos foram lidos, comentados para gerar documentação e explicados em um relatório próprio, que será mostrado na seção seguinte.

Como C# é uma linguagem orientada a objetos, o entendimento do código seguiu a estrutura principal deste: reconhecimento das principais classes, objetos e métodos.

4.3.2.2.2 Software Embarcado

Para obter o software embarcado, foi preciso, primeiramente, alimentar o sistema. Uma fonte de alimentação externa foi então conectada à placa mãe. A conexão da fonte diretamente à placa mãe foi importante, pois garantiu que somente a parte que se tinha interesse fosse ligada. Não seria conveniente e seguro que a parte mecânica recebesse energia quando apenas se pretendia obter o código embarcado.

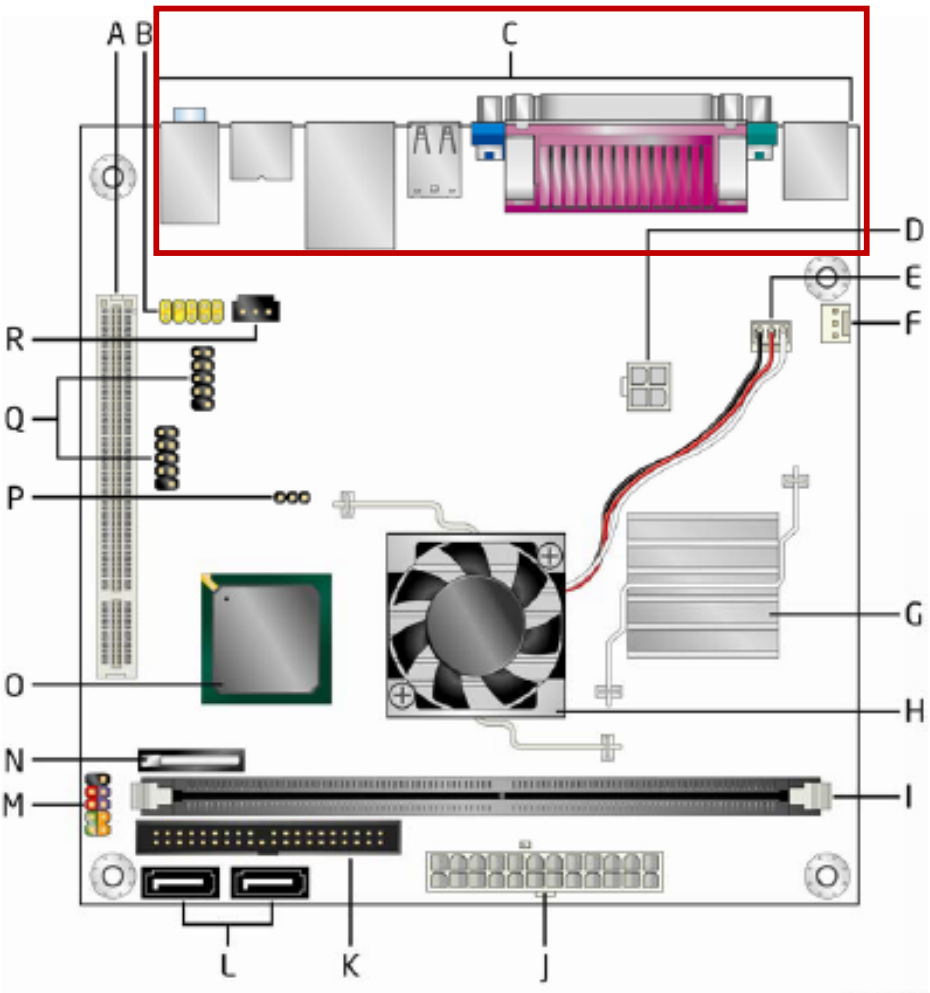
Algumas dificuldades contornadas foram:

i) Encontrar uma fonte de alimentação com as conexões corretas:

Era preciso uma conexão macho-macho para conectar a fonte de alimentação ao sistema. Como o padrão da fonte era uma conexão macho-fêmea, foi necessário adaptá-la para tornar possível a ligação.

Além da fonte de alimentação, também foram conectados à placa mãe, um teclado, um mouse e um monitor. Não foi necessário usar conexão sem fio, pois havia duas portas USB disponíveis e portas do tipo PS/2 para mouse e teclado no robô, todos localizados no painel de conectores da placa mãe, conforme destacado na Figura 36. Um teclado e um *pen drive* foram conectados nas portas USB e um mouse na porta PS/2. Uma informação que merece ser ressaltada é que, como o mouse foi conectado na porta PS/2, foi necessário reinicializar o computador após sua conexão, antes de poder utilizá-lo.

Figura 36 – Destaque do painel de conectores da placa mãe do Nanook.



Fonte: Intel (2008).

Depois de inicializar a CPU (utilizando a fonte de alimentação de um computador padrão, conforme exposto anteriormente), o código embarcado foi obtido e copiado para um *pen drive*.

Antes da análise propriamente dita, investigou-se o ambiente de desenvolvimento do código. Para desenvolver o software do projeto embarcado no robô foi utilizado o Visual Studio 2008. Para chegar a essa conclusão, um dos arquivos de extensão VCPROJ foi aberto usando o Bloco de Notas e foi obtida a seguinte descrição no cabeçalho do arquivo: Version="9.00". Pesquisando essa informação na *internet*, foi encontrado que a versão 9.00 corresponde ao Visual Studio 2008.

Foi feita uma cópia do código para usá-lo no Visual Studio 2013 Ultimate, de modo a ser possível gerar diagramas UML (*Unified Modeling Language*). UML é uma linguagem de modelagem padronizada "[...] designada para especificar, visualizar, construir e documentar um sistema" (MACORATTI, n.d.). Conforme Barros (1998, apud HAFEMANN, 2000), a utilização de diagramas UML é "[...] uma tentativa de padronizar graficamente a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível". A versão atualizada do código, contudo, foi utilizada apenas para a geração desses diagramas, mantendo-se a versão original do código segura em uma outra pasta.

O software embarcado foi então analisado seguindo o mesmo modelo do código de interface com o usuário. Contudo, nesse caso, a linguagem de programação usada foi C++. O código foi entendido e explicado conforme os arquivos *headers* (.hpp e .h), arquivos fonte (.cpp), arquivos de projeto (.vcproj) e arquivos de solução. Só depois de compreender a estruturação do código é que suas funcionalidades foram analisadas.

4.3.3 Conclusões da Análise

Algumas conclusões já foram expostas durante a descrição da análise, uma vez que serviram até mesmo para guiar o desenvolvimento desta. São citadas aqui as conclusões que levaram a implementação de melhorias no robô.

4.3.3.1 Hardware

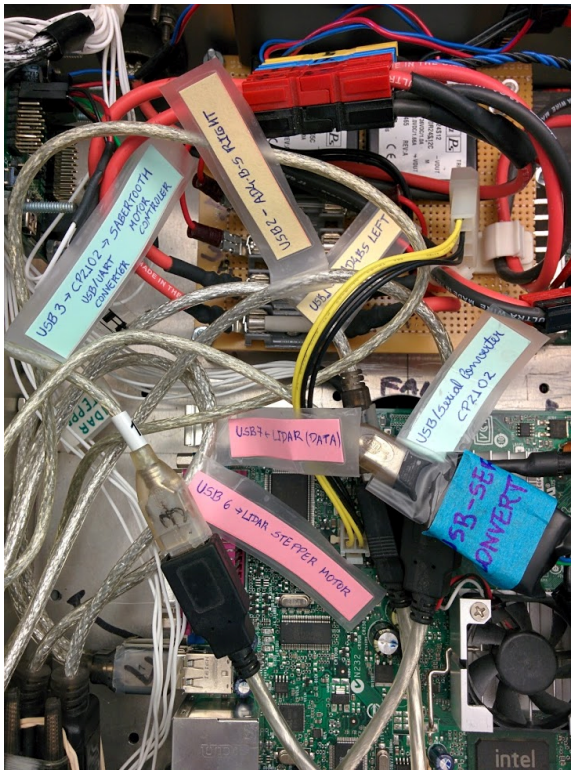
A partir das conclusões obtidas na análise do sistema eletrônico do Nanook, foi elaborado um esquemático deste usando o AutoCad®. Esse esquemático pode ser visualizado no apêndice A.

4.3.3.1.1 Organização do Hardware

Como era confuso ter um cabo identificado com a etiqueta “Direita” conectado ao motor esquerdo pelo AD4-B-S e um cabo identificado como “1” conectado ao USB “3” enquanto existe o USB “1”, optou-se por acrescentar novas identificações para facilitar o entendimento das conexões do sistema.

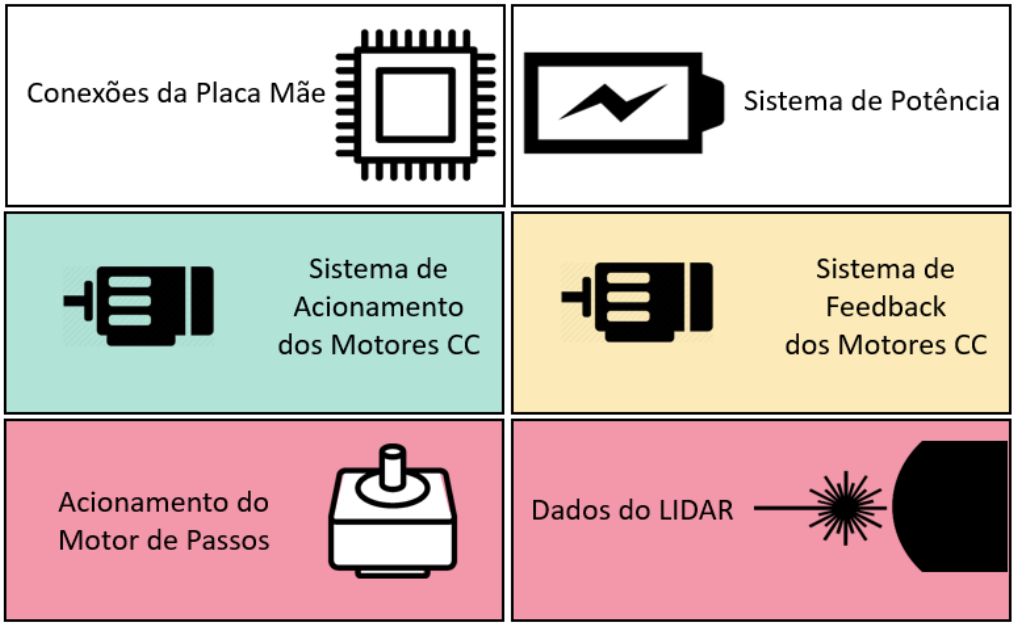
Etiquetas de identificação também foram colocadas na caixa cinza principal para facilitar a compreensão dos circuitos, conforme ilustrado na Figura 37. Os circuitos associados a cada cor estão evidenciados na Figura 38. Essa diferenciação torna mais simples a análise dos circuitos apenas olhando as etiquetas e entendendo o que são e a qual circuito pertencem. Diferentes cores evidenciam os diferentes circuitos principais (por exemplo, o sistema de motores CC possui etiquetas cinza, enquanto o sistema do LIDAR possui etiquetas rosa).

Figura 37 - Etiquetagem dos circuitos do robô.



Fonte: Acervo do autor.

Figura 38 – Destaque da divisão dos circuitos e a cor das etiquetas a eles associada.



Fonte: Elaborado pelo autor.

Para ajudar na identificação futura do hardware, etiquetas foram colocadas nos fios dos circuitos presentes embaixo da caixa cinza, conforme ilustrado na Figura 39. Desse modo, tornou-se possível rastrear todos os fios apenas retirando a caixa cinza (não sendo mais necessário retirar a isolação e o sistema do LIDAR).

Figura 39 - Identificação dos fios.

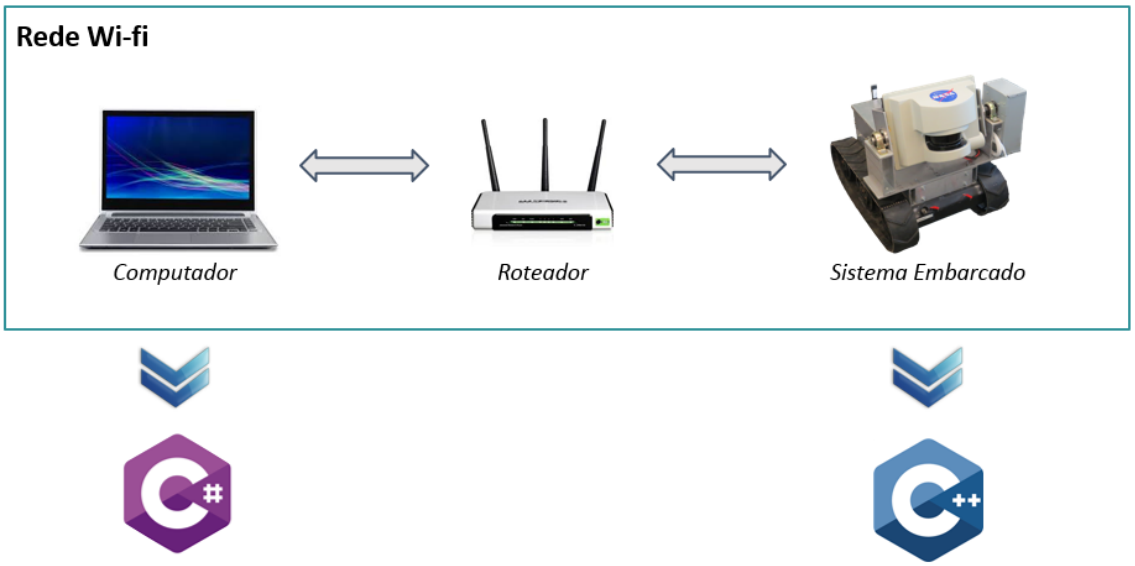
Fonte: Acervo do autor.

4.3.3.2 Software

Após a análise do código, que consistiu em duas etapas: análise do código embarcado (que roda na CPU do robô) e análise do programa de interface com o usuário, concluiu-se que o software do computador foi escrito em C#, enquanto o software embarcado foi escrito em C++. Ambos foram feitos utilizando o paradigma de Programação Orientada ao Objeto (POO). O ambiente de desenvolvimento utilizado foi o Visual Studio 2008.

O estudo, tanto do programa de interface com o usuário quanto do sistema embarcado, foi extremamente importante para garantir o entendimento de como a comunicação entre o computador do usuário e o robô era feita, como a transferência de dados ocorria e como o controle do robô acontecia, conforme evidenciado na Figura 40.

Figura 40 – Interação entre o programa de interface do usuário e o software embarcado por meio da rede wi-fi.



Fonte: Elaborado pelo autor.

Algumas melhorias foram feitas no código, uma vez que sua análise foi dificultada por muitas partes deste não serem usadas ou não serem funcionais. As modificações são descritas a seguir:

i) Eliminação de linhas não usadas:

O código agora está mais simples de ser entendido. Apesar da eliminação de linhas não utilizadas serem uma vantagem, algumas referiam-se a trabalhos iniciados, mas não finalizados. Para não perder essa informação, a versão original do código (apenas com a inclusão de comentários para a documentação HTML, que será referida mais adiante) foi mantida. As modificações foram feitas em uma nova versão do software, que foi colocada no Github. O Github é um serviço web gratuito que permite a hospedagem de projetos pessoais e funcionalidades extras aplicadas ao git. O git, simplificadamente, é um sistema de controle de versão de arquivos, permitindo que diversas pessoas possam contribuir simultaneamente em um mesmo projeto, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas. Assim, a partir de agora, será mais fácil rastrear as exatas mudanças em cada uma das versões do código. Além disso, por ser aberto, outras pessoas que utilizem o GitHub podem acompanhar as novas versões do código, contribuir informando bugs e até mesmo enviando código e correções. (POZZEBON, 2015)

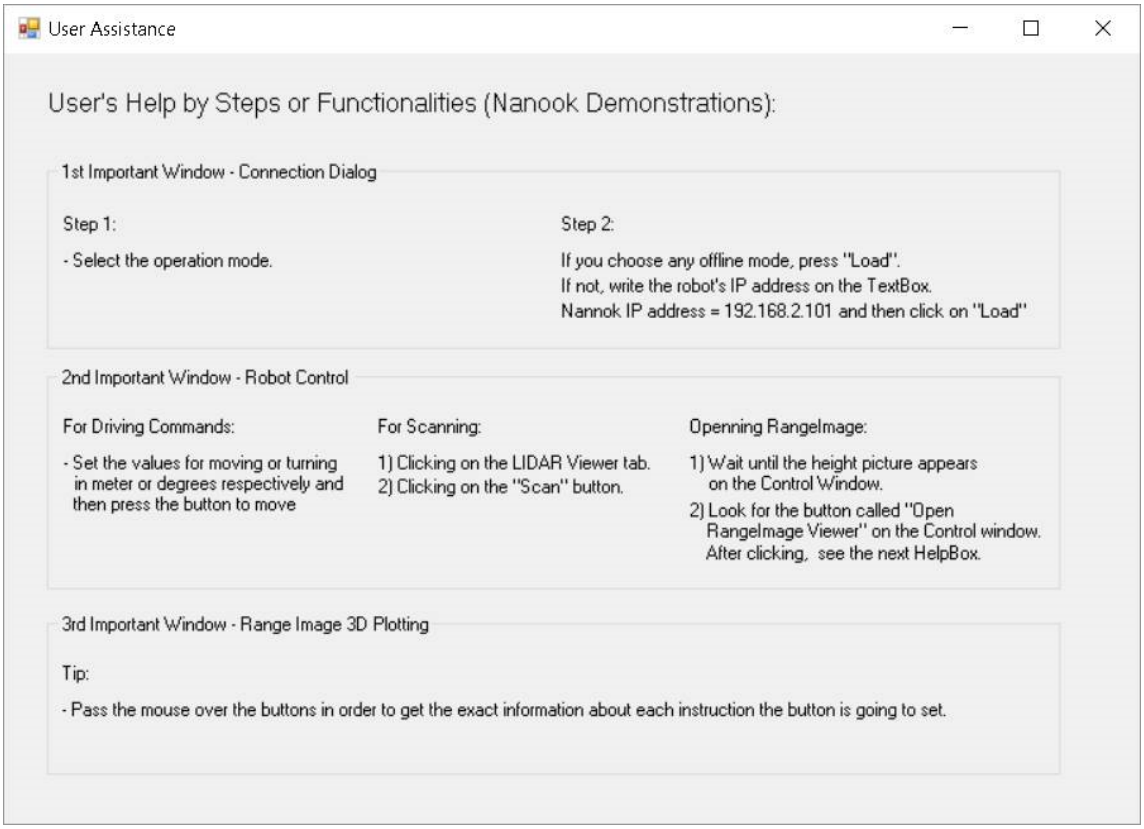
ii) Correção do Modo off-line:

Tornar o modo off-line funcional habilitou que todos as opções de operação do robô (os modos) funcionassem adequadamente. O modo off-line (planejamento de navegação, reconhecimento de esferas e visualização de imagens) não estava funcionando corretamente, então o código foi modificado para corrigir suas funcionalidades.

iii) Adição de um botão de ajuda:

Um botão de ajuda foi adicionado nas janelas, de modo que, caso uma pessoa tenha alguma dúvida sobre como usar o software, possa ter essas questões rapidamente esclarecidas. Isso irá ajudar a evitar que as funcionalidades do robô não funcionem adequadamente por culpa do usuário. O ícone do botão de ajuda, o símbolo de uma interrogação, conforme evidenciado na Figura 42, ao ser clicado, abre uma nova janela com informações de ajuda pertinentes. Essa nova janela está ilustrada na Figura 41.

Figura 41 – Janela de ajuda.



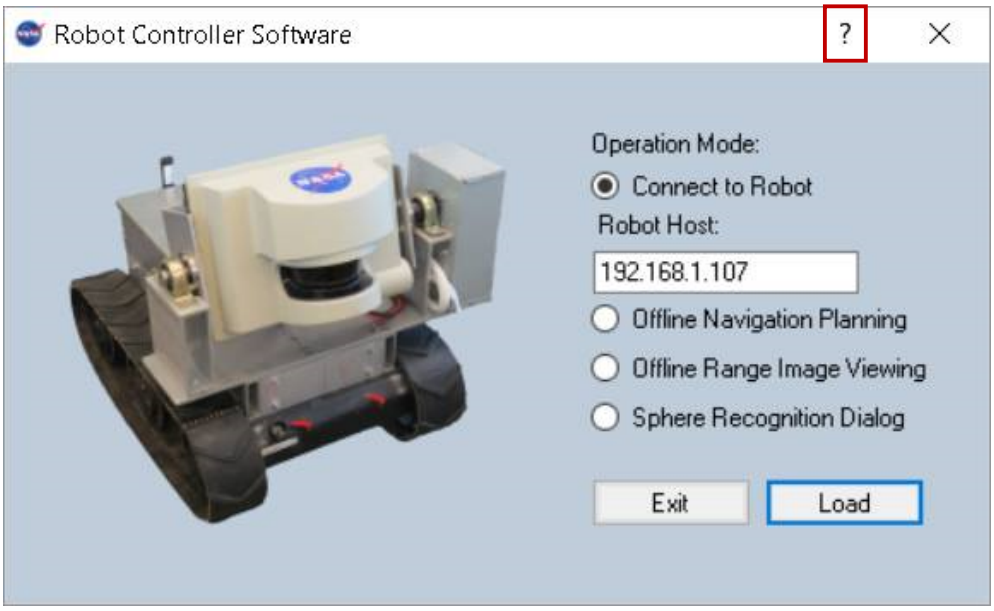
Fonte: Acervo do autor.

iv) Mudança da interface do diálogo de conexão:

Essas modificações foram mais estéticas do que a resolução de um problema em si. A imagem do Nanook foi adicionada no diálogo de conexão para personalizá-lo, conforme Figura 42.

Os dois códigos (embarcado e do programa de interface com o usuário) foram ainda comentados de modo a gerar uma documentação HTML. Para isso, foi usado o software Doxygen, porque se tratava de uma versão livre e uma ferramenta padrão para geração de documentação. Pavim (2006, p. 5) sintetiza o Doxygen como “um sistema flexível de documentação de código-fonte, multi-linguagem, multi-plataforma e com múltiplas saídas”. Sua principal função é reconhecer os comentários em linguagem específica no código fonte, de modo a criar e organizar uma documentação, que pode apresentar-se em formatos populares e publicações eletrônicas, como pdf (via latex) ou HTML (RODRIGUES, n.d).

Figura 42 – Nova interface do diálogo de conexão do Nanook



Fonte: Acervo do autor

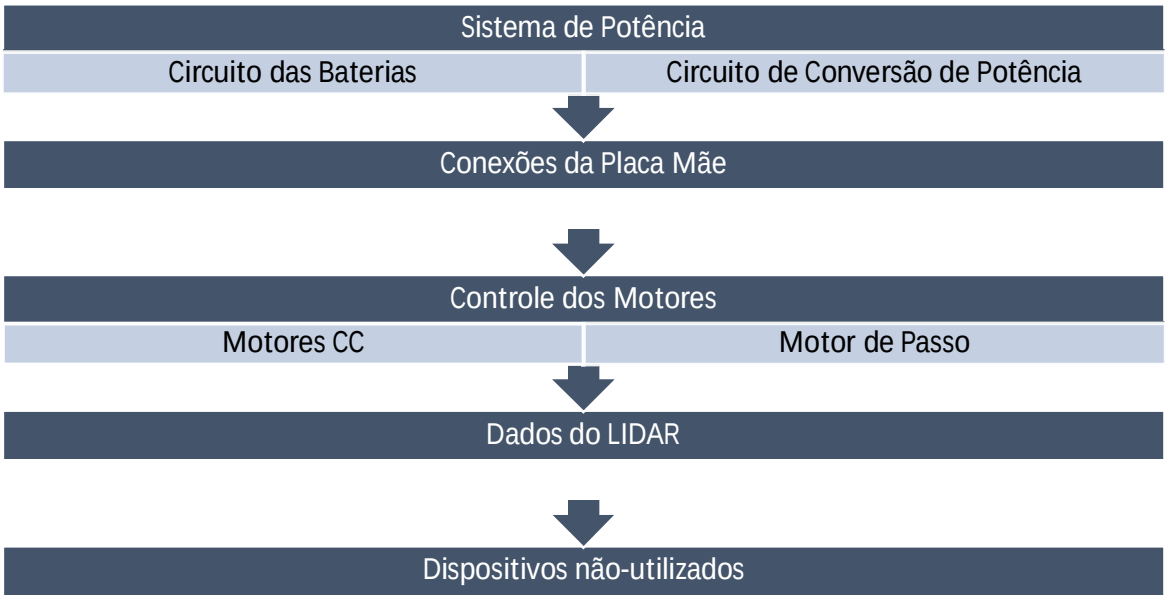
O acesso a essa documentação é possível utilizando o link para a pasta do Nanook ou acessando uma pasta no Github (recomendado) em: <https://github.com/anareboucas/nanook.git>.

5 DOCUMENTAÇÃO DO HARDWARE

Todo o sistema eletrônico do robô foi documentado, de modo que seja possível o entendimento de seus circuitos e sua réplica. Além disso, com a descrição completa do sistema, a proposta de melhoria torna-se uma missão mais amena.

A descrição do hardware foi documentada seguindo a seguinte estrutura: detalhamento do sistema de potência, das conexões da placa mãe, do sistema de controle dos motores e dos dados do LIDAR e esclarecimento dos dispositivos presentes no Nanook que não estavam sendo utilizados, conforme evidenciado na Figura 43.

Figura 43 - Esquema utilizado na documentação do hardware.



Fonte: Elaborado pelo autor.

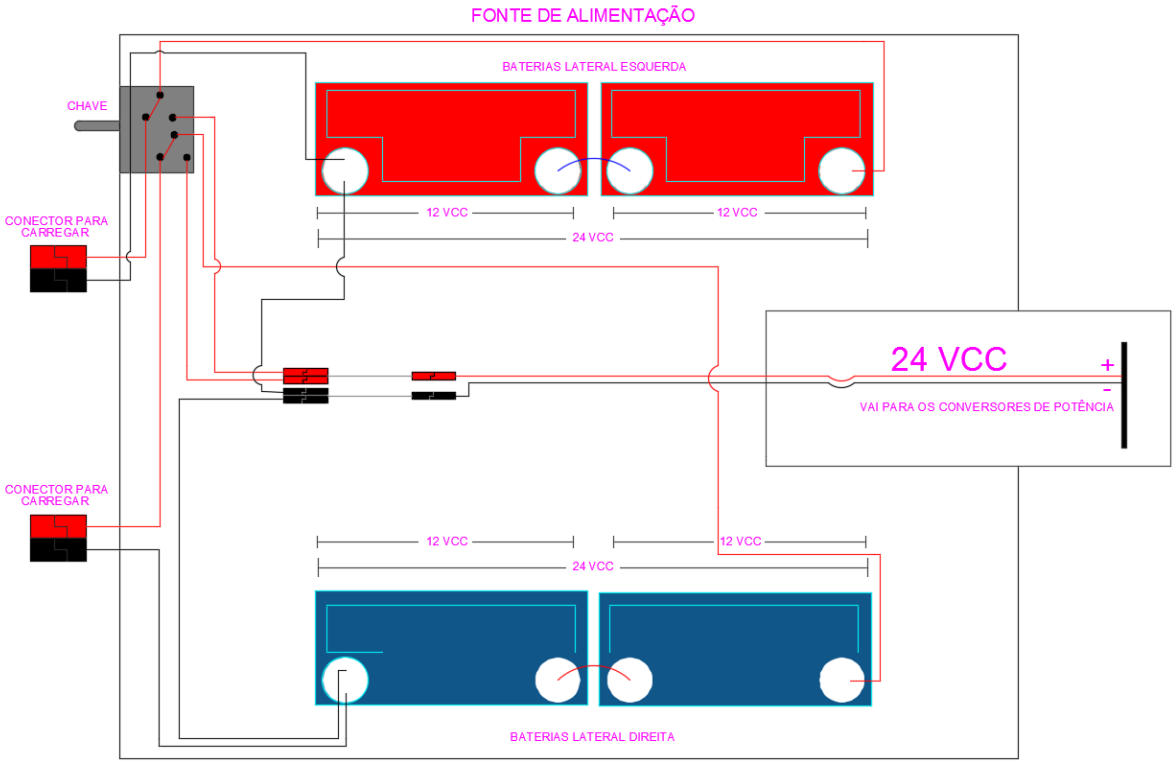
5.1 Sistema de Potência

O sistema de potência do Nanook é composto por dois importantes circuitos. O primeiro deles é o circuito da bateria, que provê a energia para todos os sistemas do robô. O segundo é um sistema de conversão de potência, que é alimentado pelo conjunto de baterias e então distribui os níveis de tensão adequados para cada um dos outros sistemas (+24 V, +12 V ou +5 V).

5.1.1 Circuito das Baterias

Existem quatro baterias de 12 V no robô. Cada lado possui um conjunto de duas baterias conectadas em série fornecendo um total de 24 V. Os dois conjuntos de baterias são conectados em paralelo para fornecer 24 V ao circuito conversor de potência. Os cabos de 24 V (VCC e neutro) vão para a caixa cinza principal como um fio vermelho e um preto. São esses os fios que vão para a placa de conversão de potência. O esquemático elaborado no AutoCad® e exposto na Figura 44 ilustra o circuito das baterias descrito.

Figura 44 - Esquemático elaborado no AutoCad® para o circuito das baterias.

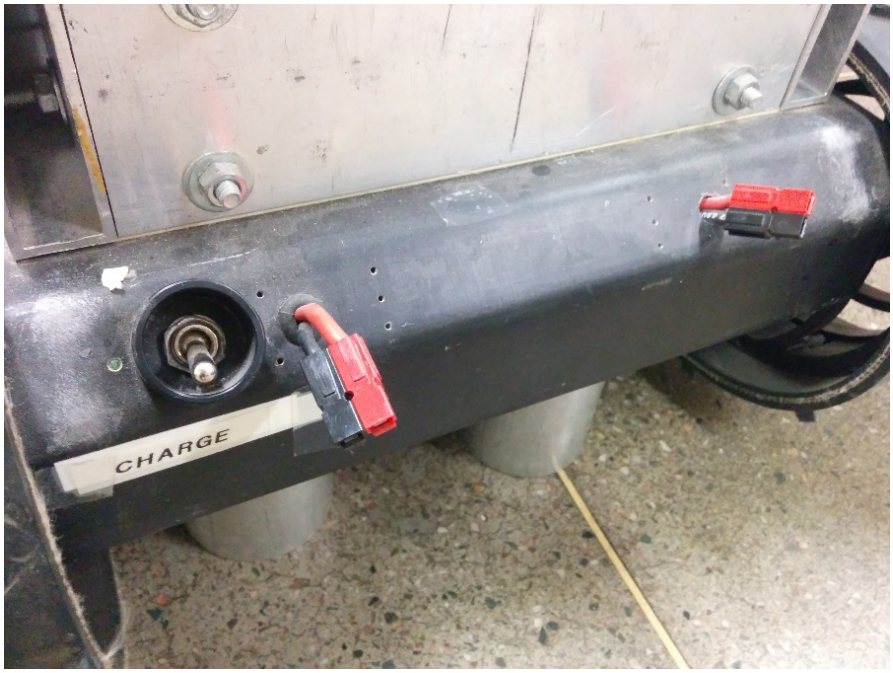


Fonte: Elaborado pelo autor.

As baterias são ainda conectadas a uma chave, de modo que o sistema seja capaz de alternar entre a conexão com o carregador e a conexão com a placa de conversão de energia responsável por fornecer energia ao robô, conforme evidenciado no diagrama da Figura 34. Observando as conexões da chave, constata-se que não é possível que as baterias sejam carregadas ao mesmo tempo que o robô é utilizado e vice-versa.

A Figura 45 ilustra os conectores para carregamento das baterias e a chave que alterna entre o estado de carregamento das baterias e o estado de utilização das baterias pelo robô.

Figura 45 - Conectores para carregamento e chave para alternar entre carregamento e fornecimento de energia para o robô.



Fonte: Acervo do autor.

A Tabela 7 mostra as principais especificações das baterias do Nanook.

Tabela 7 – Especificações das baterias.

Especificação	Bateria azul	Bateria vermelha
Modelo	PSH-12180FR	PC680
Marca	PowerSonic	Odyssey
Tensão nominal	12 V	12 V
Capacidade nominal	21 $A \cdot h$	16 $A \cdot h$
Peso	6 kg	7 kg
Material	Chumbo-ácido	Chumbo-ácido

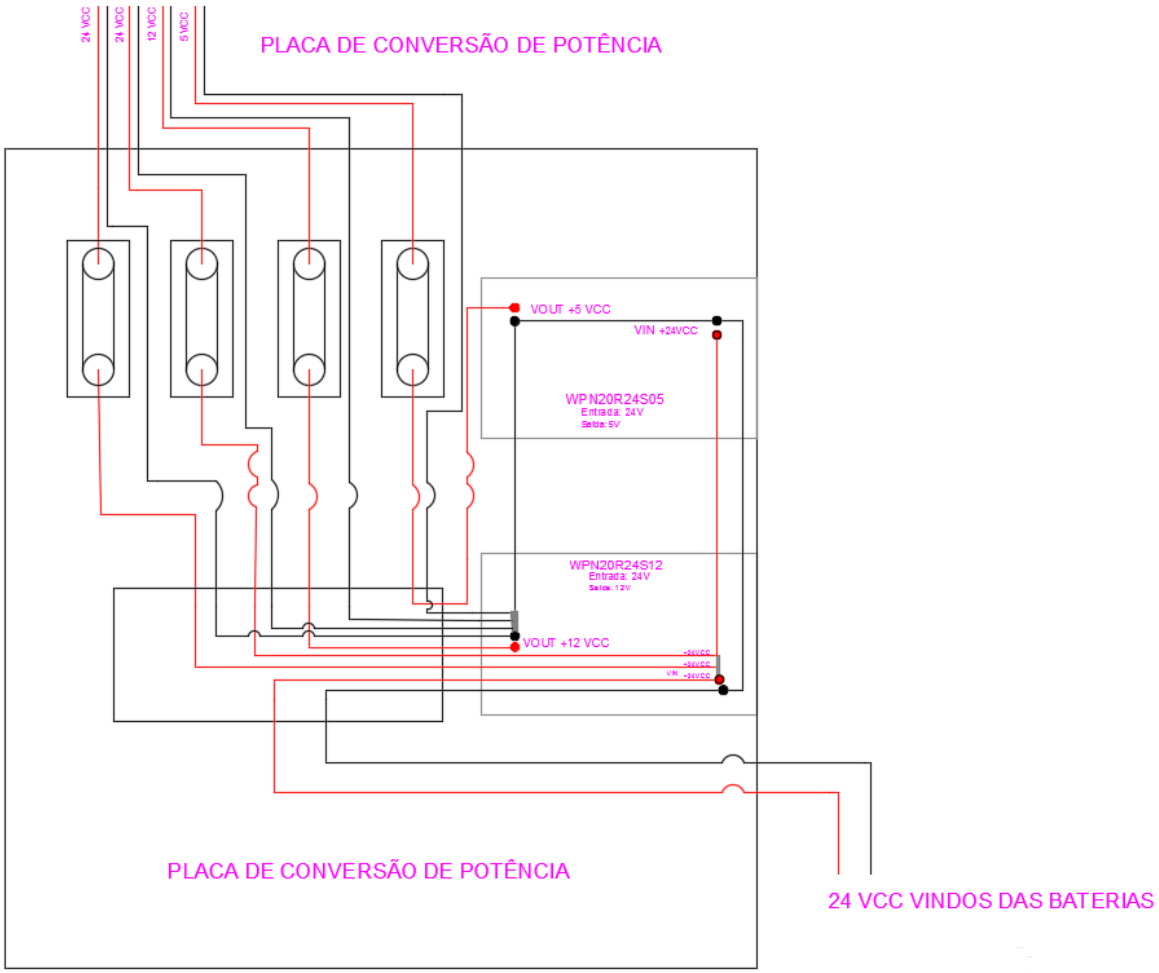
Fonte: Baseeado em Power Sonic (n.d) e West Cost Batteries (n.d.).

Ao realizar uma inspeção mais rigorosa do hardware, percebeu-se que as duas baterias azuis do robô estavam danificadas e precisavam ser substituídas, conforme citado anteriormente e exemplificado na Figura 15 e na Figura 16.

5.1.2 Circuito de Conversão de Potência

O esquemático do circuito de conversão de potência foi elaborado no AutoCad®, conforme a Figura 46.

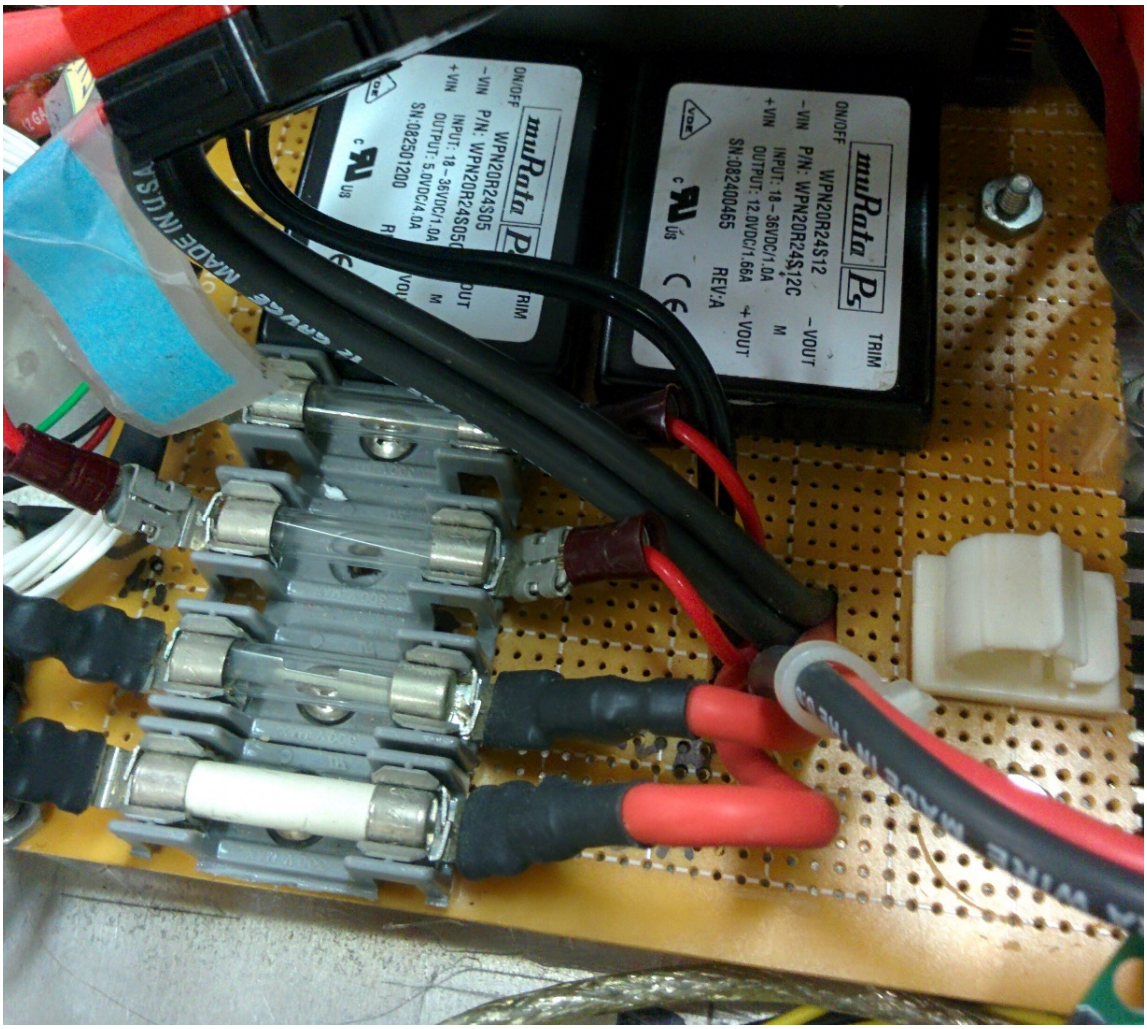
Figura 46 - Esquemático do circuito conversor de potência.



Fonte: Elaborado pelo autor.

A entrada da placa de conversão de potência é de 24 V, energia essa que vem das baterias. Nessa placa, o nível de tensão da bateria é convertido para 12 V e para 5 V nos CIs (circuitos integrados) WPN20R24S12 e WPN20R24S05, respectivamente. Esses componentes citados estão atualmente obsoletos. Há ainda um fio de + 24 V que é dividido em outros dois fios, apenas isolando-os com uma fita. Esse procedimento não é recomendável porque não há garantia do nível de tensão das baterias. Seria mais adequado utilizar um conversor com saída de 24 V, de modo a assegurar que esse fosse o nível de tensão entregue ao LIDAR, à placa mãe e ao Sabertooth. A visualização dessa placa é mostrada na Figura 47.

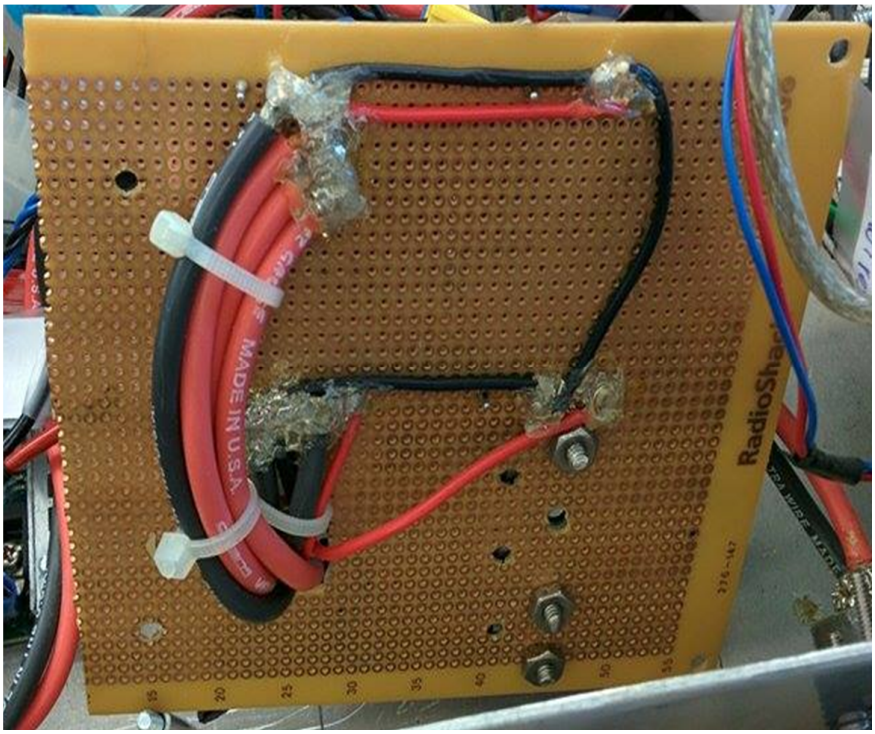
Figura 47 - Placa de conversão de potência.



Fonte: Acervo do autor.

Depois dos conversores, o VCC passa por um fusível. Existe um para o fio de 5 V, um para o fio de 12 V e dois para os fios de 24 V. A tensão de 5 V é identificada na placa com a letra “J” e vai para o conector amarelo marcado com um “5”. A tensão de 12 V é identificada na placa com a letra “M” e vai para o conector azul marcado com um “12”. Ambos fios de 24 V vão para conectores vermelhos após passar por seus fusíveis. A Figura 48 abaixo mostra o verso da placa de conversão de potência. Além das ligações, pode-se observar que essa placa possui potenciais perdas por efeito Joule devido às conexões entre os fios.

Figura 48 - Verso da placa do circuito conversor de potência.



Fonte: Acervo do autor.

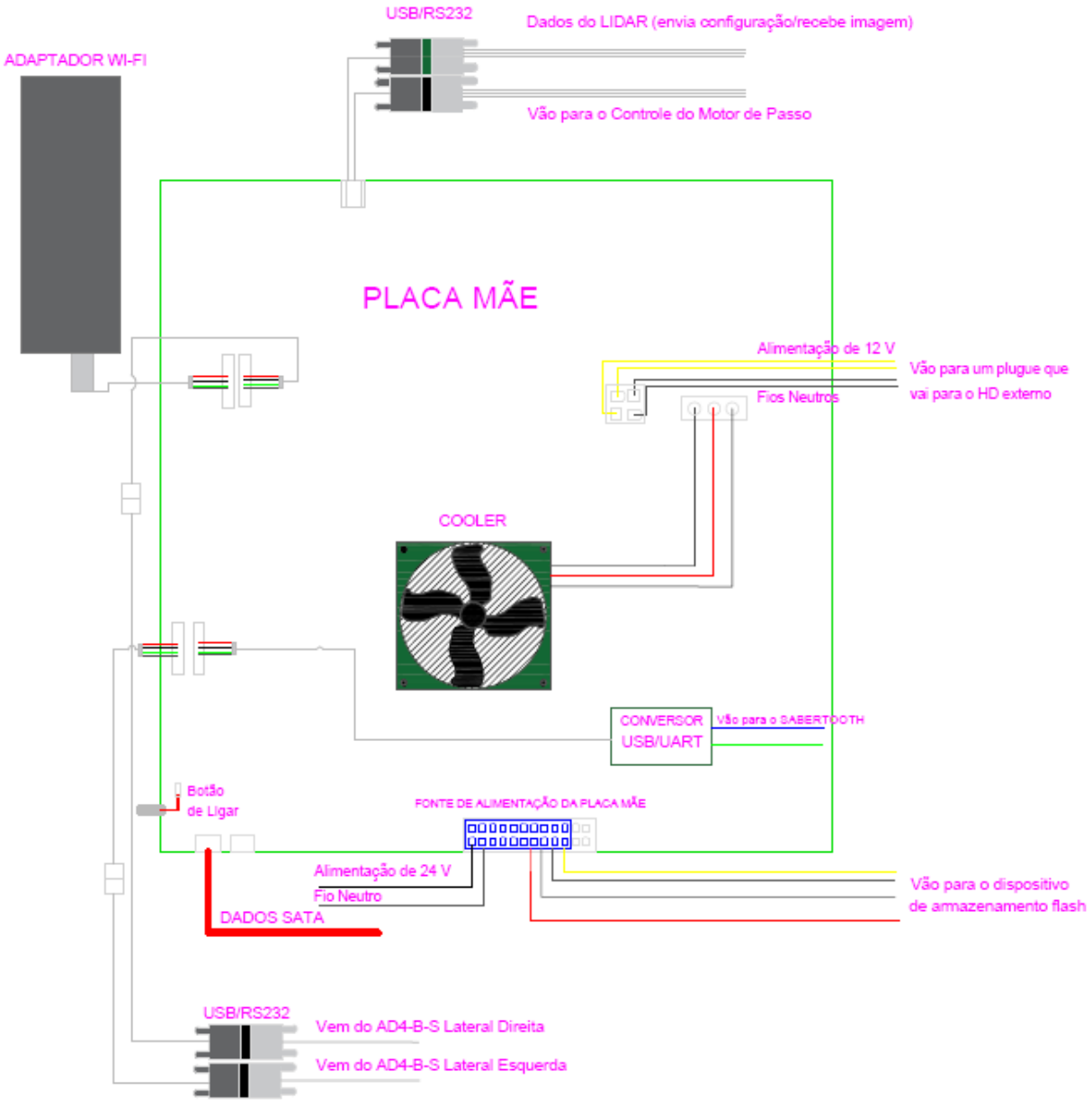
5.2 Conexões da Placa Mãe

A placa mãe, de modelo D945GCLF2 da Intel®, possui oito conexões USB, mas somente seis destas são utilizadas para controlar o sistema. As outras duas entradas USBs provavelmente foram mantidas livres para permitir a conexão de *pen drives* e outros dispositivos similares.

O esquemático das conexões da placa mãe está ilustrado abaixo, na Figura 49, bem

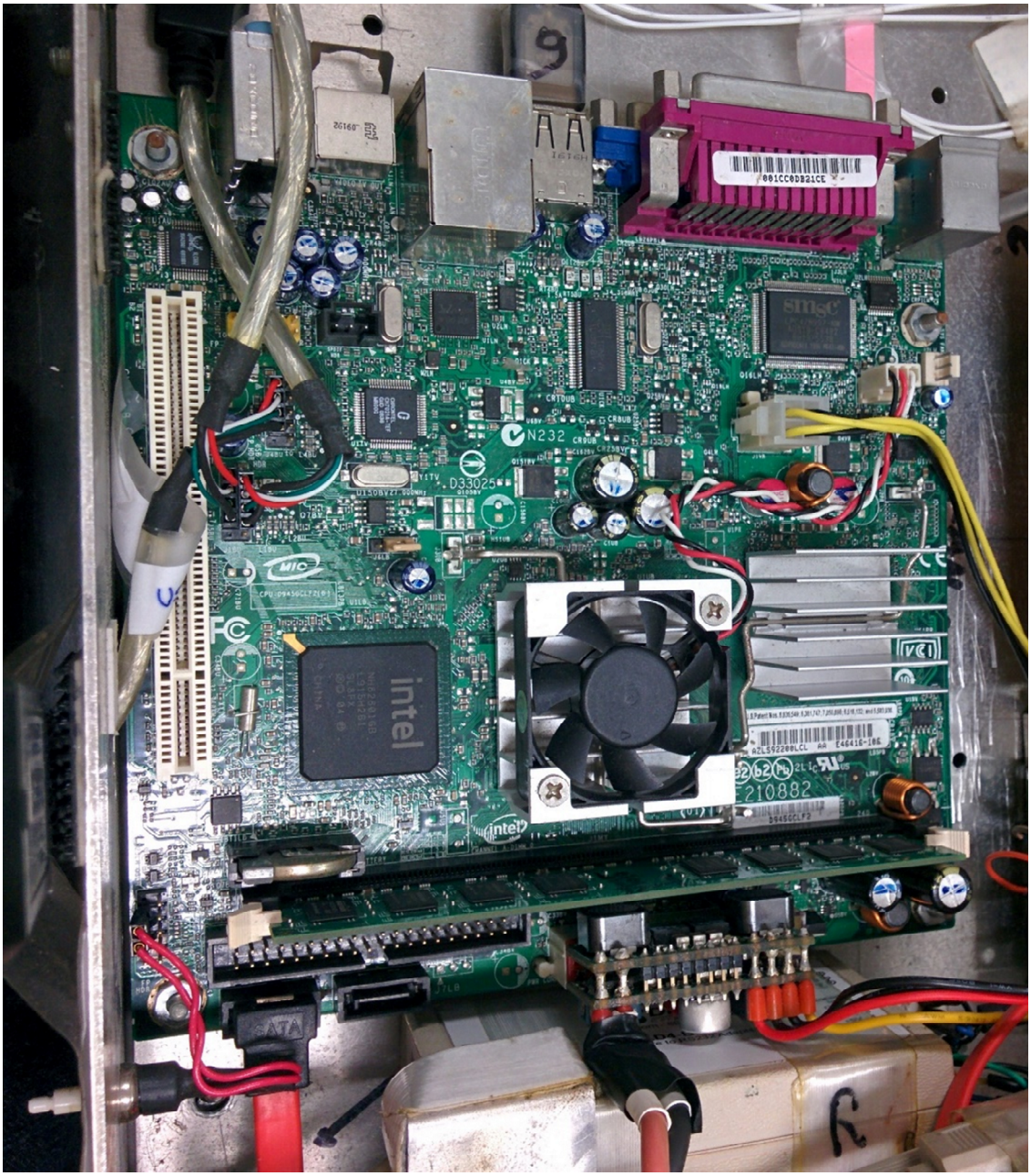
como a visualização desta no Nanook, exposta na Figura 50.

Figura 49 - Esquemático das conexões da placa mãe.



Fonte: Elaborado pelo autor.

Figura 50 - Placa mãe do Nanook: modelo D945GCLF2 da Intel®.



Fonte: Acervo do autor.

As portas USB foram nomeadas seguindo a convenção empregada em uma anotação fixada no verso da tampa da caixa cinza principal, conforme referido na Figura 19.

A Tabela 8 lista a conexão dos USBs da placa mãe e seu destino final, de modo a facilitar seu entendimento.

Tabela 8 - Conexões USB

Identificação do USB	Conexões Intermediárias	Destino Final
USB1	Conector RS232	AD4-B-S Esquerdo
USB2	Conector RS232	AD4-B-S Direito
USB3	CP2102 (conversor USB/UART)	SABERTOOTH 2X25
USB4	--	Adaptador Wi-Fi
USB5	Não utilizado	
USB6	Conector RS232	Sistema do Motor de Passo do LIDAR
USB7	Conector RS232	LIDAR (conexão verde)
USB8	Não utilizado	

Fonte: Elaborado pelo autor.

Os USBs 1 e 2 recebem os dados dos *encoders* e, consequentemente, fornecem os sinais de *feedback* necessários para o controle da posição do robô. O acionamento dos motores CC depende, portanto, desses sinais. Os dados dos *encoders* passam pelo AD4-B-S para que este conte os pulsos enviados, que fornecerão a informação da distância percorrida pelo robô quando convertidos adequadamente na CPU. Como a saída do AD4-B-S é em RS232, esta é convertida em USB antes de chegar à placa mãe. Também vale destacar que, como os sinais do *encoder* são sinais de quadratura, eles podem ser lidos por microcontroladores que já possuam periféricos adequados, como alguns da *Freescale*. No caso do Nanook, o AD4-B-S possui um microcontrolador interno que faz a conversão do sinal. Como o sinal é convertido diretamente para um sinal serial, já se tem um cenário facilmente trabalhável.

O USB 3 é responsável por enviar os sinais de controle dos motores de corrente contínua. O *software* embarcado decide o comando que deve chegar no *driver* do motor, o Sabertooth. Um caractere em ASCII associado a esse comando é enviado pelo USB 3, mas convertido para o protocolo UART pelo CP2102 (conversor USB/UART) antes de finalmente

chegar ao Sabertooth.

O USB 4 conecta a placa mãe ao adaptador Wi-Fi.

O USB 6 é responsável por enviar comandos para o sistema de motor de passo do LIDAR. Esses comandos são associados ao ajuste vertical da posição do LIDAR.

O USB 7 é responsável por compartilhar (enviar e receber) dados com o LIDAR, usando o protocolo RS232. É por ele que a configuração de transmissão de dados serial é definida (*baud-rate* e número de pontos em cada linha de escaneamento, por exemplo) e também por onde os dados da imagem obtida são transferidos.

Além dos já mencionados, a placa mãe tem ainda um *cooler* e um dispositivo de armazenamento flash (HD externo) conectados a ela. Existe uma conexão de dados SATA do drive externo para a placa mãe (SATA0) e fios de energia ligando o conector de alimentação da placa mãe ao HD externo. Esse plugue é também ligado ao conector 2x2 na placa mãe – dois fios neutros (fios pretos) e dois fios de +12 VCC (fios amarelos). O modelo do dispositivo de armazenamento flash SSD, evidenciado na Figura 51, é SNV125-S2/64GB.

Figura 51 - Dispositivo de armazenamento flash do Nanook.

Fonte: Acervo do autor.

O conector de alimentação da placa mãe recebe 24 V e converte isso nos níveis de tensão adequados para alimentar a CPU (+5 V, - 5V, +12 V e -12 V).

Na Figura 17, é possível observar a indicação dos USBs conectados à placa mãe (1, 2, 3, 4, 6 e 7), a conexão do HD externo e a fonte de alimentação da placa mãe vindo do sistema de potência. O cooler não foi representado nesse diagrama.

Por fim, o botão de ligar – botão branco localizado externamente à caixa cinza principal, conforme ilustrado na Figura 52 – é também conectado a placa mãe.

Figura 52 - Botão para inicializar a CPU.

Fonte: Acervo do autor.

5.3 Controle dos Motores CC

Há quatro motores de corrente contínua (CC) no Nanook, do modelo GM9236S027-R1. Eles são motores alimentados por 24 V e capazes de fornecer um torque contínuo de cerca de 3,5 $N \cdot m$. Os motores CC estão ligados em paralelo, sendo dois deles responsáveis por mover as engrenagens do lado direito, enquanto os outros dois motores são responsáveis por mover as engrenagens do lado esquerdo, permitindo um acionamento com apenas dois canais. O Sabertooth, dispositivo usado para acionar os motores, tem dois canais

de saída, o que o torna compatível para essa tarefa. Apesar de haver quatro motores, o robô é um corpo sólido, portanto, em movimentos retos, os dois motores na mesma lateral devem apresentar a mesma velocidade, de modo a não promover nenhuma deformação no robô ou impactar os motores. No caso de manobras de giro, a combinação de movimentos inversos para cada um dos lados, permite a rotação, cuja amplitude irá depender do tempo de manobra. A Figura 53 destaca os motores CC localizados na parte de trás do Nanook.

A relação da velocidade de giro das engrenagens do motor é de 65.5:1 (PITTMAN, n.d.). O *encoder* a eles associados tem três canais e resolução de 500 CPR (ciclos por revolução).

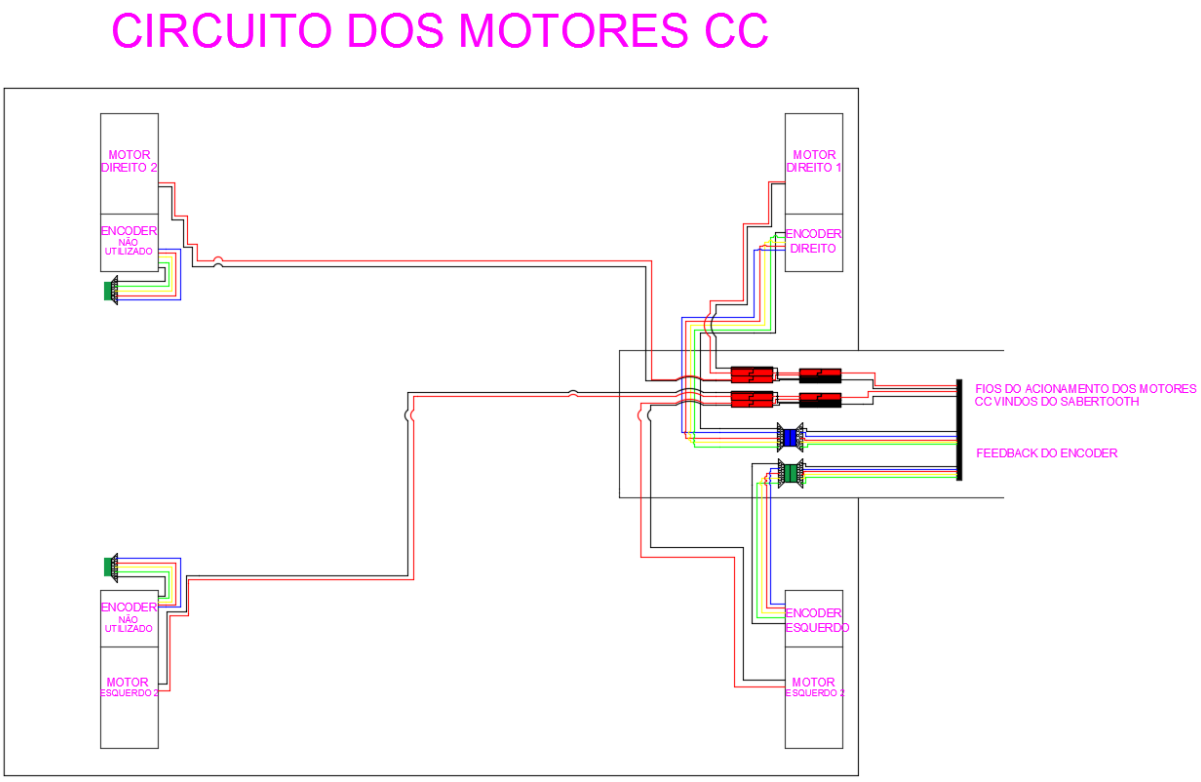
Figura 53 - Motores CC na parte de trás do Nanook.

Fonte: Acervo do autor.

Os motores CC são acionados pelo Sabertooth, mas o comando a eles enviado é ajustado a partir das informações de *feedback* obtidas através dos *encoders*. Considerando o diagrama em blocos do Nanook, ilustrado na Figura 17, e as conexões da placa mãe anteriormente explicadas, o sistema de controle dos motores CC recebem comandos da CPU pelo USB 3 e envia sinais pelos USBs 1 e 2 com informações de *feedback*. A explicação detalhada do controle dos motores CC foi, portanto, dividida em Sistema de Comando e Sistema de *Feedback* para facilitar a compreensão.

A Figura 54 ilustra as conexões finais dos motores CC, que serão explicadas a seguir.

Figura 54 - Esquemático das conexões dos motores CC.



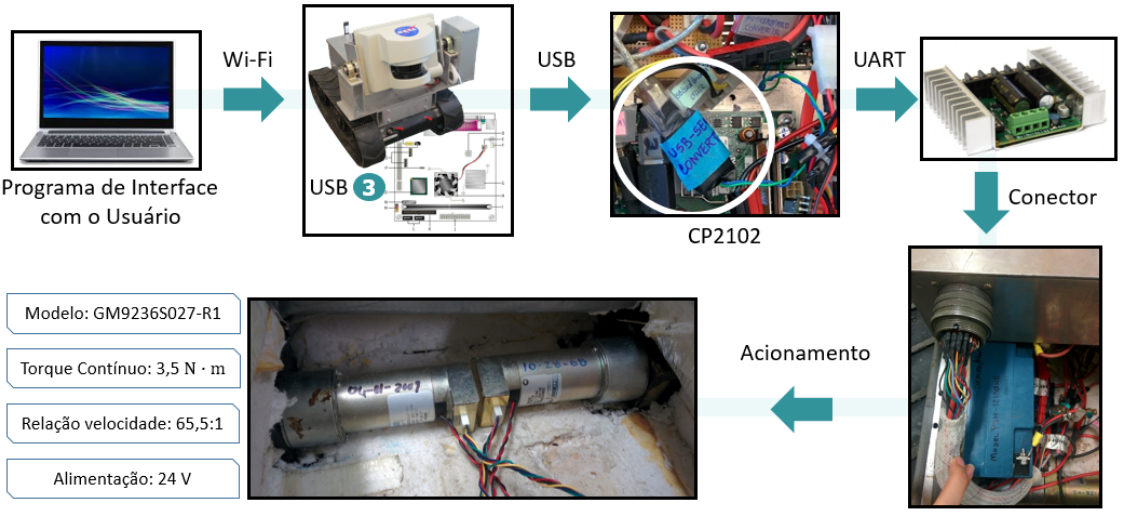
Fonte: Elaborado pelo autor.

5.3.1 Sistema de Comando

O sistema de comando dos motores CC é constituído basicamente pelo USB 3, CP2102 e Sabertooth, conforme evidenciado na Figura 55.

O software do computador envia o comando e o software embarcado processa os dados que vêm do USB 4 via w*i-fi*. Depois do processamento, o software embarcado envia os comandos necessários para os motores CC via USB 3. Como o Sabertooth (*driver* do motor CC) requer uma comunicação no protocolo UART, o USB 3 é conectado no conversor USB-UART, o CP2102. Com isso, a tradução do protocolo USB para o protocolo UART é realizada e o Sabertooth passa a ser capaz de receber o comando do software embarcado como um sinal de entrada.

Figura 55 – Sistema de acionamento dos motores CC.

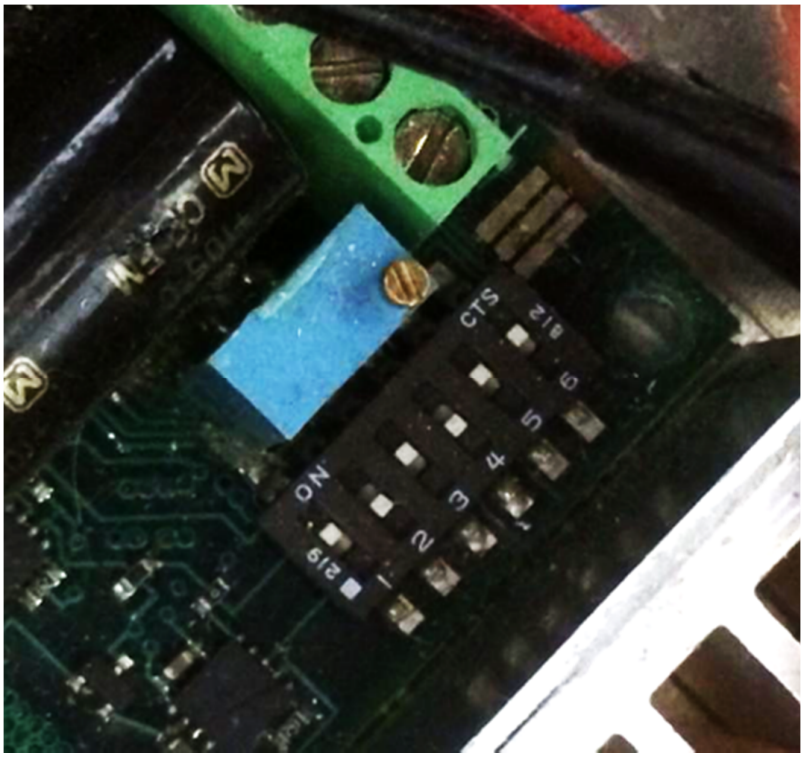


Fonte: Elaborado pelo autor.

O Sabertooth recebe dois fios (um verde e um azul) vindos do CP2102 (o conversor USB/UART). Esses fios correspondem ao fio terra e ao sinal de transmissão de dados, Tx (que será conectado ao sinal de recepção de dados, Rx, do Sabertooth), vindos da placa mãe. A comunicação entre a placa mãe e o Sabertooth é unilateral, por isso só esses fios são requeridos. Vindo da placa de conversão de potência, o Sabertooth recebe uma alimentação de 24 V (VCC e neutro). A saída do Sabertooh é composta de dois pares de fios preto e vermelho que vão para o conector na borda da caixa cinza principal (no canto direito) e então para os motores. Cada par é conectado a um dos dois motores CC localizados no robô (um no lado esquerdo e outro no lado direito do robô). Esses fios correspondem a alimentação enviada aos motores, cujos níveis de tensão dependem do controle determinado pela placa mãe.

Para receber os sinais UART, o Sabertooth trabalha no Modo Serial Simplificado. Esse estado foi detectado pela inspeção de hardware, uma vez que as chaves numeradas de 1 a 6 estavam visíveis, conforme mostrado na Figura 56. Portanto, agregando a indicação visível das chaves e o *datasheet* do Sabertooth (*driver* do motor CC) foi possível afirmar que o *driver* do motor CC estava trabalhando no Modo Serial Simplificado Padrão a uma taxa de transmissão de dados de 9600 *baud-rate*.

Figura 56 - Chaves responsáveis pela seleção do modo e da taxa de transmissão de dados definida.



Fonte: Acervo do autor.

O modo serial simplificado é selecionado quando as chaves 1, 2 e 3 estão no estado para cima (1), para baixo (0) e para cima (1), respectivamente. As chaves 4 e 5 selecionam a taxa de transmissão de dados. No caso da chave 4 e 5 estarem no estado para baixo (0) e para cima (1), respectivamente, o *baud-rate* é de 9600. Selecionando a chave 6 para cima, a opção padrão é mantida, o que significa que os comandos são enviados como bytes únicos.

O *datasheet* do Sabertooth 2x25 expõe ainda sua funcionalidade dependendo do sinal recebido. Esse sinal deve ser um caractere entre 0 e 255, garantindo uma resolução de 7 bits para cada um dos conjuntos de motores (dois no total) controlados pelo Sabertooth.

> *"O envio de um caractere entre 1 e 127 controlará o motor 1. 1 é totalmente reverso, 64 é parado e 127 é totalmente adiante. O envio de um caractere entre 128 e 255 controlará o motor 2. 128 é totalmente reverso, 192 é parado e 255 é totalmente adiante. O caractere 0 (hex 0x00) é um caso especial. Enviá-lo interromperá o movimento dos dois motores"* (DIMENSION ENGINEERING, 2007).

A Tabela 9 resume os comados enviados e as ações consequentes dos motores.

Tabela 9 - Sinais de controle dos motores CC.

Motores	Função	Comandos em Binário (Hexadecimal/Decimal)
Motor 1 e Motor 2	Parado	0b00000000 (0x00/000)
Motor 1	Totalmente Reverso	0b00000001 (0x01/001)
	Parado	0b01000000 (0x40/064)
	Totalmente Adiante	0b01111111 (0x7F/127)
Motor 2	Totalmente Reverso	0b1000000 (0x80/128)
	Parado	0b11000000 (0xC0/192)
	Totalmente Adiante	0b11111111 (0xFF/255)

Fonte: Elaborado pelo autor.

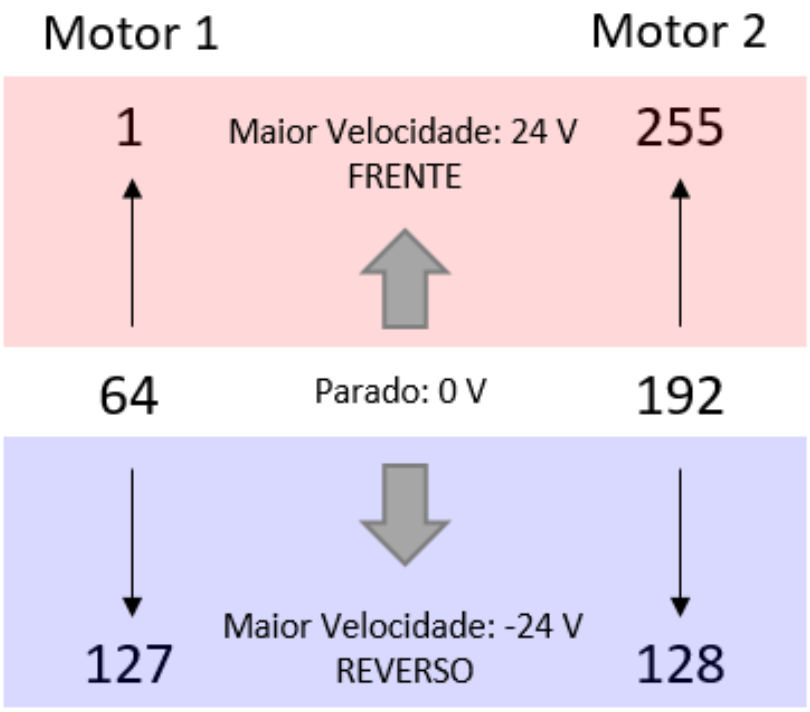
Desse modo, o sistema embarcado envia um comando para o Sabertooth (conforme explicado, esse comando é um caractere entre 0 e 255). Como a placa mãe envia esse comando em uma comunicação serial pelo USB 3, é preciso primeiramente convertê-lo para UART no CP2102, consoante já exposto. Quando esse caractere chega ao Sabertooth, de acordo com as informações destacadas na tabela abaixo, tem-se uma ação para os motores, que pode ser mover para frente, para trás ou parar. Somente um canal (associado a um conjunto de dois motores) é acionado de cada vez, mas como essa comunicação é rápida, isso não é percebido na prática. Pode-se, no entanto, parar os dois conjuntos de motores simultaneamente enviando o caractere 0. Para o conjunto de motores acionados quando um caractere entre 1 e 127 é recebido pelo Sabertooth, quanto maior for a diferença desse número para 64 (convergindo para 1 ou 127), maior será a velocidade do motor. Para o caractere 64, o motor permanecerá parado. Valores acima disso implicam em um movimento das rodas para a frente, enquanto valores inferiores implicam em um movimento reverso nesse lado. Para o conjunto de motores acionados quando

um caractere entre 128 e 255 é recebido no Sabertooth, raciocínio análogo é aplicado, sendo 192 o caractere associado à sua parada. Valores acima deste implicam em um movimento para a frente, enquanto valores inferiores implicam em um movimento reverso. Quanto mais o valor do caractere se distanciar de 192 (convergindo para 128 ou 255), maior será a velocidade do motor.

Apesar de nas especificações acima evidenciar-se que, no modo serial simplificado padrão, o Sabertooth só é capaz de acionar dois motores, isso precisa ser entendido como dois canais de acionamento. O Sabertooth pode acionar os quatro motores CC do Nanook, uma vez que, como visto, eles estão organizados em dois conjuntos paralelos. Portanto, os sinais relacionados a cada um dos lados são enviados para os dois motores presentes no lado correspondente.

O Sabertooth aciona esses motores enviando uma tensão entre -24 V e 24 V. O nível de tensão depende do sinal recebido pelo Sabertooth. Quanto maior o módulo da tensão, mais rápido será a velocidade dos motores. Essa variação de tensão é possível por uma variação do *duty cycle*. A Figura 57 apresenta um resumo do movimento dos motores (direção e velocidade) de acordo com o caractere enviado ao Sabertooth.

Figura 57 - Movimento dos motores de acordo com o caractere enviado ao Sabertooth.

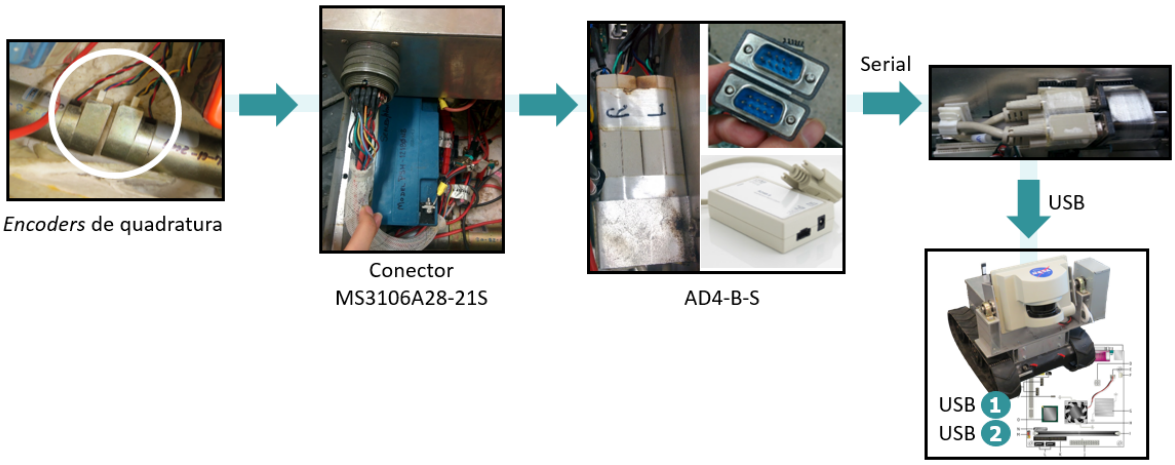


Fonte: Elaborado pelo autor.

5.3.2 Sistema de Feedback

O sistema de feedback consiste principalmente dos *encoders* do lado direito e esquerdo, do AD4-B-S associado a cada lado e dos USBs 1 e 2, conforme Figura 58.

Figura 58 –Sistema de feedback dos motores CC



Fonte: Elaborado pelo autor.

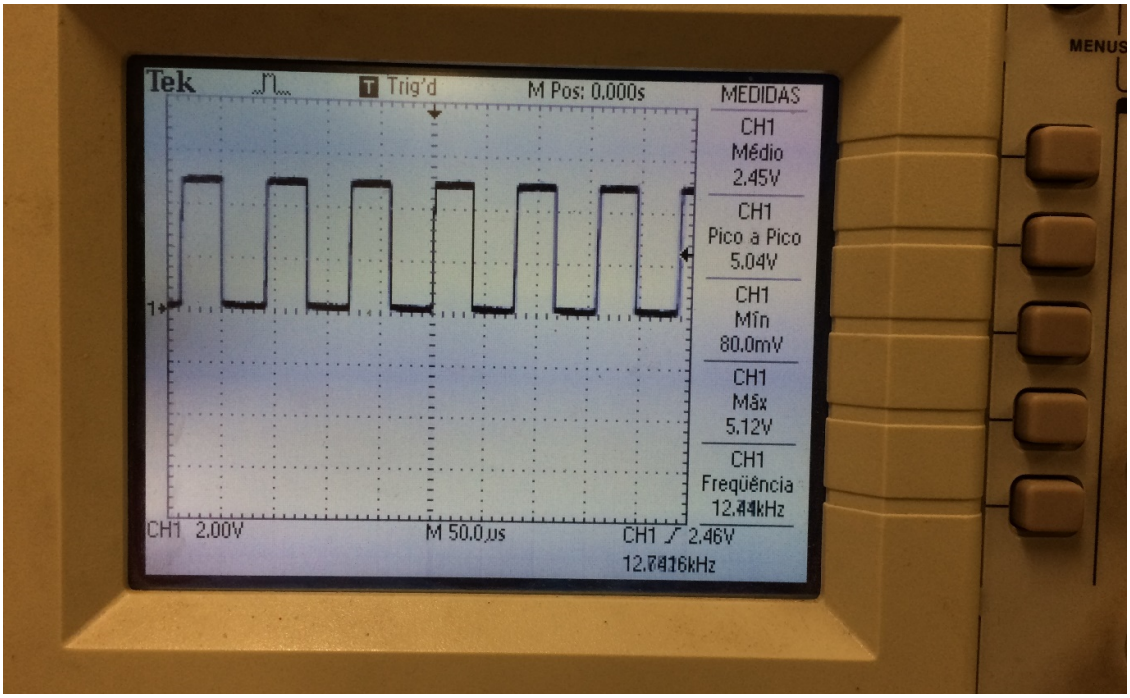
Os dados de feedback vêm dos *encoders* conectados nos motores CC. Existem quatro *encoders* no total, mas dois deles – localizados na frente do Nanook – não estão sendo utilizados. Os dois *encoders* localizados na parte de trás do robô, mostrados na Figura 59, sendo um no lado direito e um no lado esquerdo, são os *encoders* que estão sendo empregados.

Os *encoders* do Nanook são *encoders* de quadratura. O termo quadratura está associado a capacidade do *encoder* determinar a direção do movimento pelo monitoramento da diferença de fase entre dois canais - saída de incremento único -, no caso, os canais A e B (DYNAPAR, n.d.). A resolução do *encoder* determina o movimento mínimo detectável por ele, que para os *encoders* do Nanook é de 500 CPR. Um exemplo do sinal de um dos canais do *encoder*, obtido com o auxílio de um osciloscópio, é ilustrado na Figura 60.

Figura 59 - Encoders do Nanook.

Fonte: Acervo do autor.

Figura 60 – Forma de onda de um dos canais do encoder.



Fonte: Acervo do autor.

O nível de tensão do sinal do encoder é de 5 V e, ao contrário do Sabertooth que variava o *duty cycle*, nesse sinal a frequência que é variável à medida que a velocidade dos motores é modificada.

Existem dois conjuntos de fios finos coloridos que vêm dos *encoders* de quadratura conectados aos motores CC. Eles interligam os *encoders* aos seus respectivos AD4-B-S, passando pelo conector MS3106A28-21S, que está localizado no canto direito da caixa cinza. O *encoder* de quadratura captura a posição de fase dos motores CC nos canais A e B e então essa informação é processada no AD4-B-S, onde será convertida em comunicação serial (RS232). No *datasheet* do AD4-B-S é possível encontrar a correspondência entre as entradas do AD4-B-S e as saídas do *encoder* de quadratura. Essa relação está exposta na Tabela 10.

Tabela 10 - Relação entre os fios do *encoder* e as entradas do AD4-B-S

Entradas do AD4-B-S (Número do fio/Cor)	**Saídas do *Encoder* de Quadratura**
1/Preto	Terra
2/Verde	Index
3/Amarelo	Canal A
4/Vermelho	+5 VCC
5/Azul	Canal B

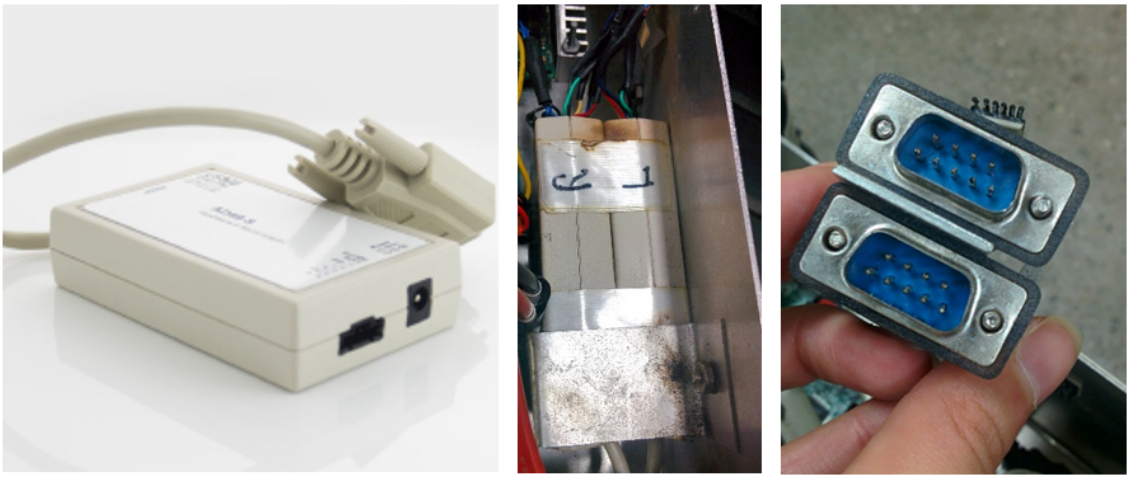
Fonte: Elaborado pelo autor

O AD4-B-S, que é alimentado por 12 V vindos da placa de conversão de potência, conta o número de pulsos do *encoder*. O AD4-B-S possui internamente um contador de quadratura, o LFLS7166, que consegue rastrear a posição do *encoder*. Um microprocessador também interno ao AD4-B-S fornece a interface serial necessária para enviar essa informação quando requisitado. A comunicação do AD4-B-S como a CPU é feita usando uma palavra binária com 8 bits, porque é um formato rápido e eficiente. Quando a placa mãe envia um caractere específico (binário correspondente à letra ‘A’ maiúscula em ASCII) para o AD4-B-S, ele retorna o valor da contagem de pulsos. A placa mãe processa essa informação fazendo a

conversão para a distância percorrida. A CPU é ainda responsável por se comunicar previamente com o AD4-B-S determinando as configurações da transmissão serial.

Como a comunicação do AD4-B-S é no formato serial (RS232), há um conector que adequa o plugue de saída, destacado na Figura 61, para uma conexão para USB, requerida pela placa mãe.

Figura 61 - Ilustração do AD4-B-S (esquerda), sua utilização no Nanook (centro) e destaque do plugue de saída (direita).

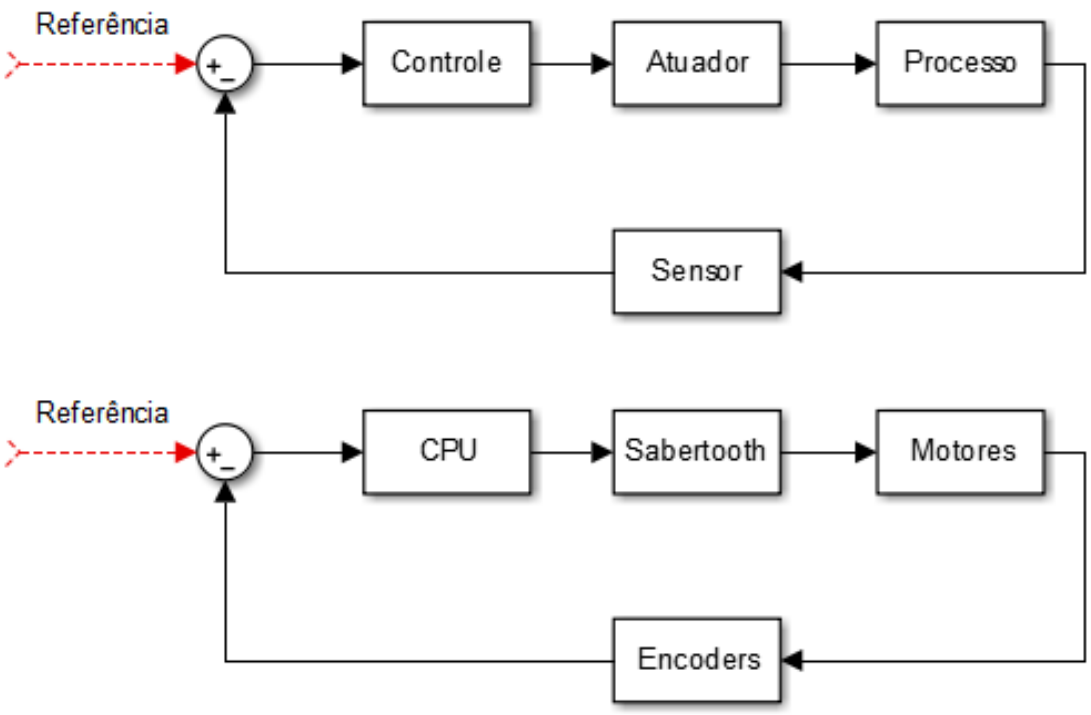


Fonte: US DIGITAL (n.d.) e acervo do autor.

Como os *encoders* fornecem o sinal de realimentação do sistema, o feedback enviado torna o controle dos motores um controle em malha fechada. Dentre os benefícios de um controle em malha fechada, destaca-se que o aumento da precisão do sistema, a rejeição ao efeito de perturbações externas e a diminuição da sensibilidade a variações dos parâmetros do processo contribuem para um sistema mais robusto (SILVA, 2000).

A Figura 62 ilustra uma versão simplificada de um diagrama em blocos comparando um sistema em malha fechada com o sistema de acionamento dos motores CC do Nanook.

Figura 62 – Diagrama de blocos do sistema em malha fechada de controle dos motores CC



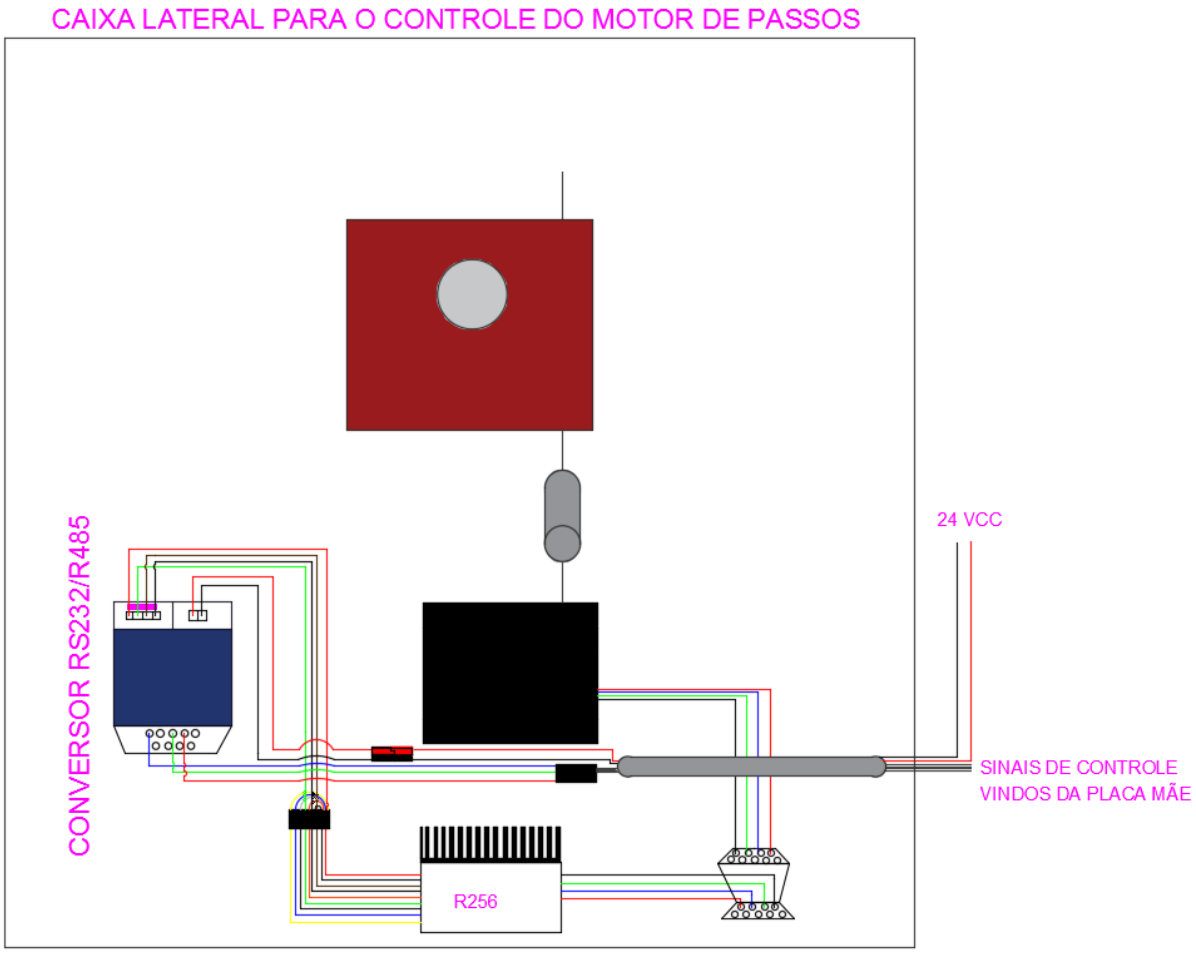
Fonte: Elaborado pelo autor.

5.4 Motor de Passos

O esquemático do circuito do controle do motor de passos, englobando o conversor RS232/R485, o R256, o motor e as engrenagens, está ilustrado na Figura 63.

O circuito para controlar o ângulo de inspeção (vertical) do LIDAR localiza-se, principalmente, na caixa lateral cinza. Nessa caixa, existe um controlador de micro passos R256 com um kit de comunicação RS232, que é responsável por controlar um motor de passos. Esse motor está associado a um sistema de engrenagens de modo a que sua rotação seja transmitida para o LIDAR, determinando seu movimento vertical. O motor de passos está em destaque na Figura 64 onde também podem ser observados os outros elementos do sistema responsável pelo movimento do LIDAR.

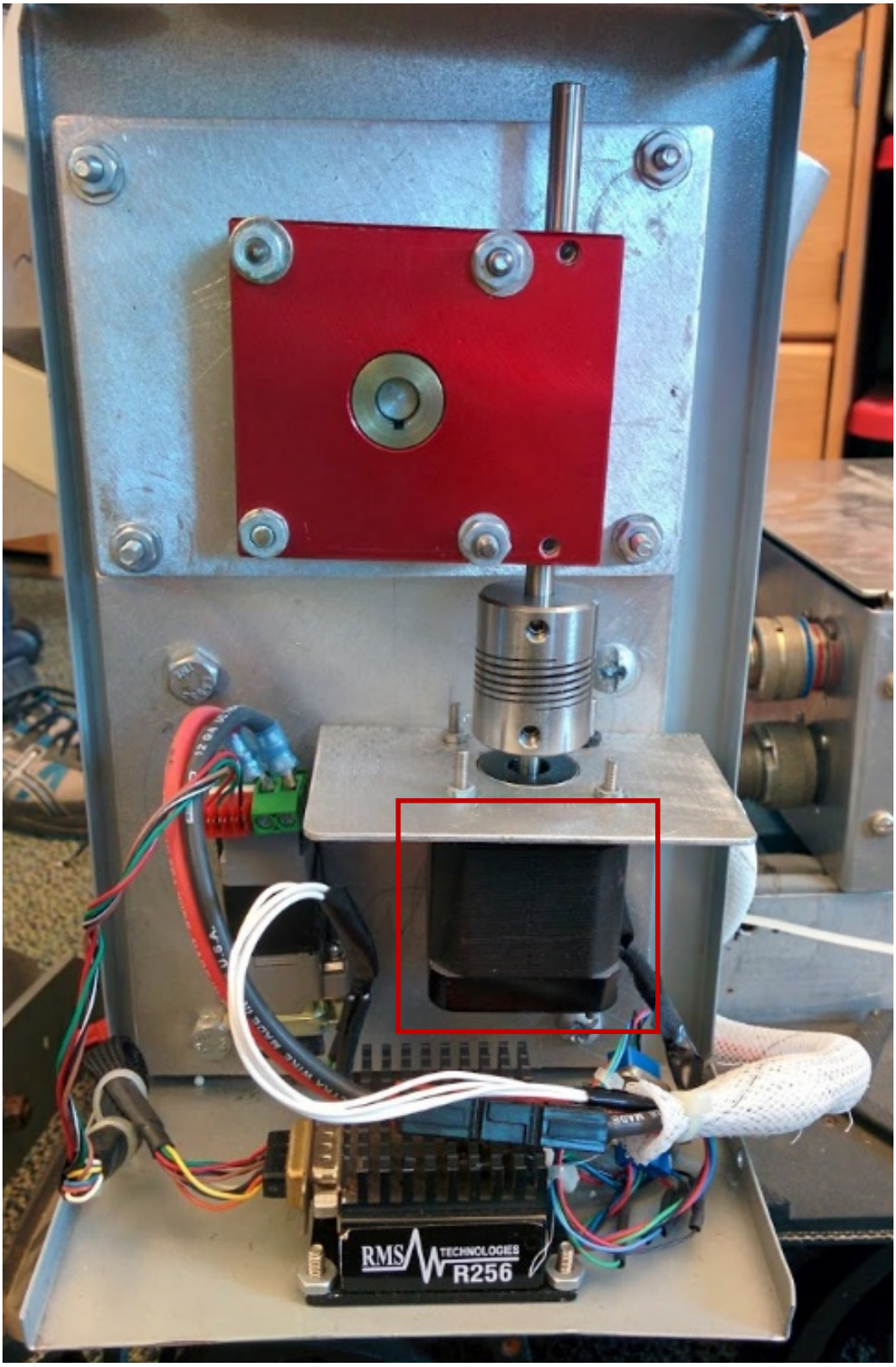
Figura 63 - Esquemático do circuito de controle do motor de passos.



Fonte: Elaborado pelo autor.

De modo a controlar esse ajuste vertical do LIDAR, a CPU envia sinais pelo USB 6, que serão convertidos na comunicação RS232. Esses sinais vão para a caixa cinza lateral, onde os comandos são traduzidos em outro protocolo, o RS485, pelo conversor RS232-RS485 (evidenciado na Figura 65). O conversor RS232-RS485 é ligado na CPU usando um cabo DB-9 padrão macho-fêmea. O conversor RS232-RS485 tem como saída uma fase “A” e uma fase “B”. A alimentação + 12 VCC é suprida tanto para o conversor quanto para o RS256.

Figura 64 - Caixa lateral cinza com sistema de controle do movimento vertical do LIDAR.



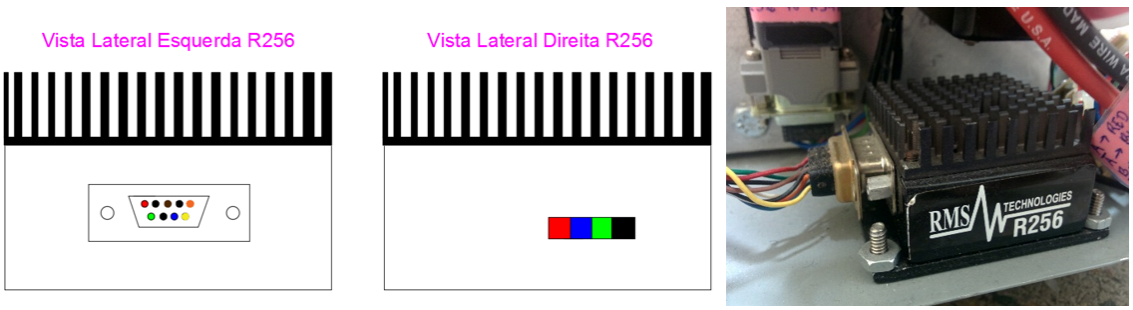
Fonte: Acervo do autor.

Figura 65 - Conversor RS232/R485.

Fonte: Acervo do autor.

Os sinais de saída estão conforme o padrão de comunicação RS485 e vão para o controlador de passos R256. A saída do R256 consiste em quatro sinais que vão para o motor de passo. O fio vermelho é A, o azul é $\bar{A}$, o verde é B e o preto é $\bar{B}$, enquanto suas respectivas conexões no R256 são destacadas na Figura 66.

Figura 66 - Destaque para o R256: vistas laterais e modelo utilizado no Nanook.

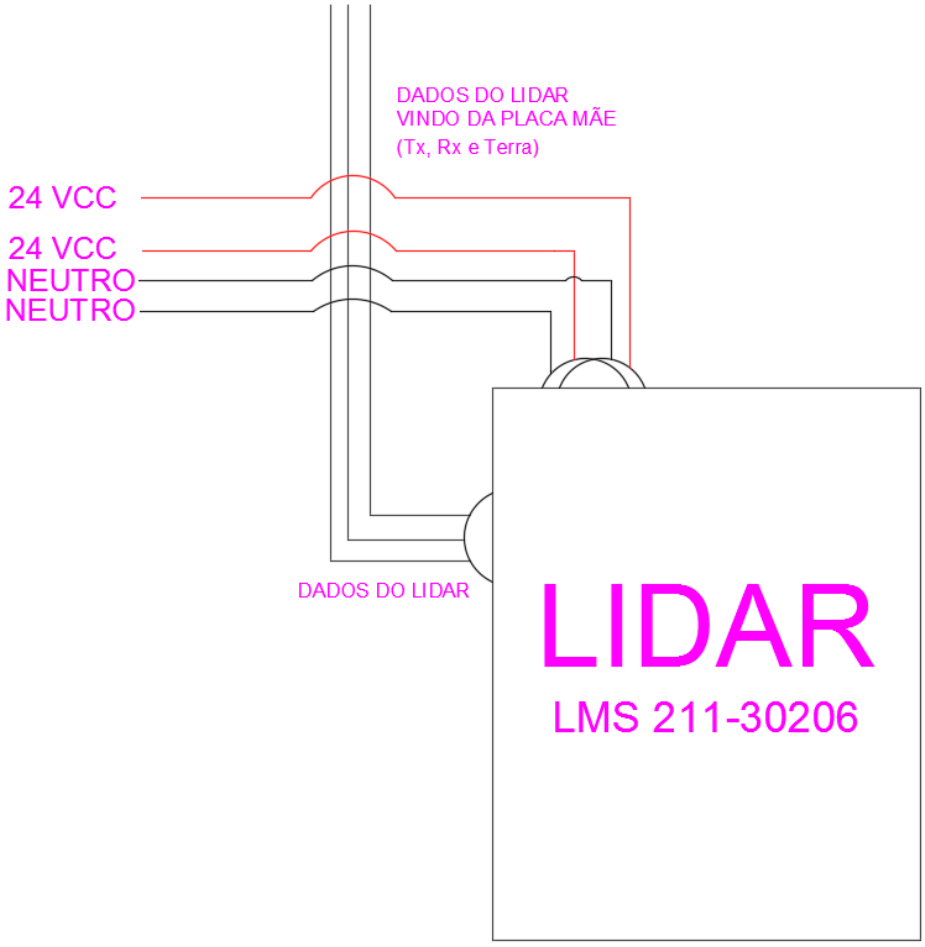


Fonte: Elaborado pelo autor.

5.5 Dados do LIDAR

Para a troca de dados entre o LIDAR e a CPU, é necessário usar o protocolo RS232 ou o RS422. No Nanook, o protocolo RS232 foi escolhido. A CPU é conectada ao LIDAR pelo USB 7. As conexões do LIDAR são ilustradas no esquemático apresentado na Figura 67.

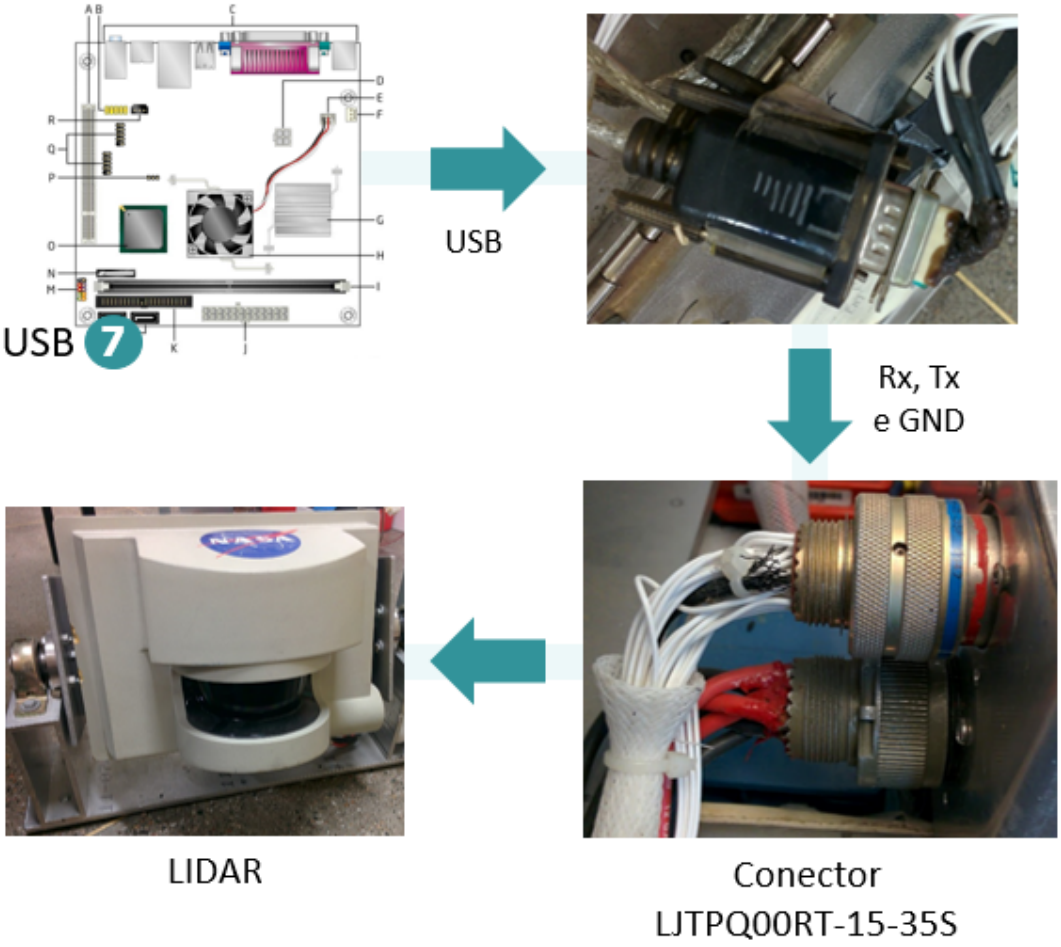
Figura 67 - Esquemático do circuito de dados do LIDAR.



Fonte: Elaborado pelo autor.

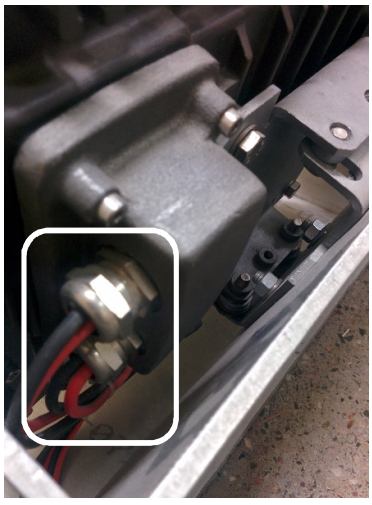
O USB 7 é convertido na comunicação RS232 e então três fios brancos passam pelo conector de modelo LJTPQ00RT-15-35S. Eles saem da caixa envoltos por uma fita preta e vão diretamente para a caixa de conexões do LIDAR. Esses fios correspondem aos fios de transmissão de dados (Tx), recepção de dados (Rx) e terra vindos da placa mãe. O LIDAR, que é do modelo LMS221, recebe ainda alimentação de 24 V, sendo dois fios de fase (24 V) e dois neutros, pois são duas entradas de alimentação, conforme a Figura 69.

Figura 68 – Conexões para transmissão dos dados da imagem obtida pelo LIDAR



Fonte: Elaborado pelo autor.

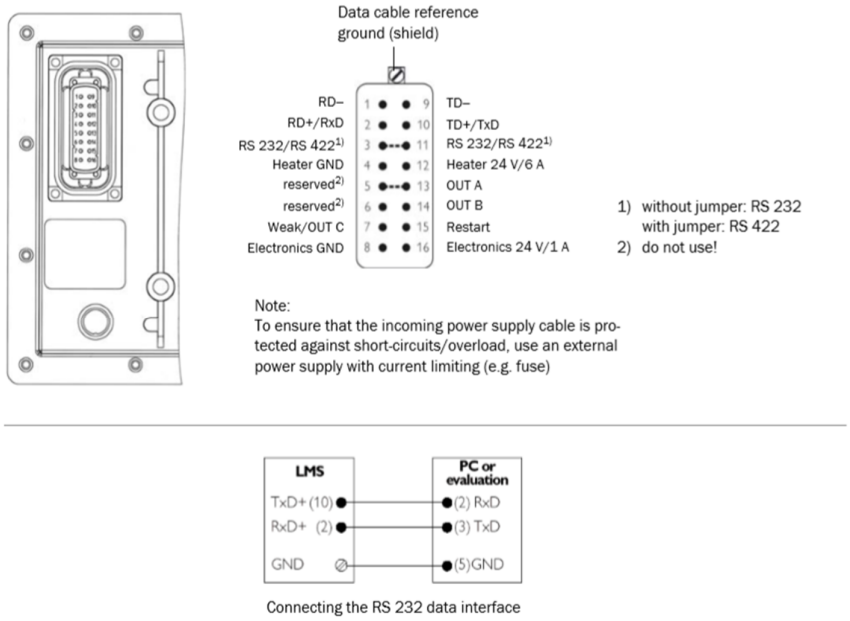
Figura 69 – Destaque para as duas entradas de alimentação no LIDAR.



Fonte: Acervo do autor.

Pela leitura do *datasheet* do LIDAR (SICK AG Waldkirch, 2006), é possível entender as conexões do LIDAR com o computador embarcado. Abaixo, na Figura 70, segue uma ilustração explicando como o LIDAR deve ser conectado ao RS232.

Figura 70 - Conexões do LIDAR.



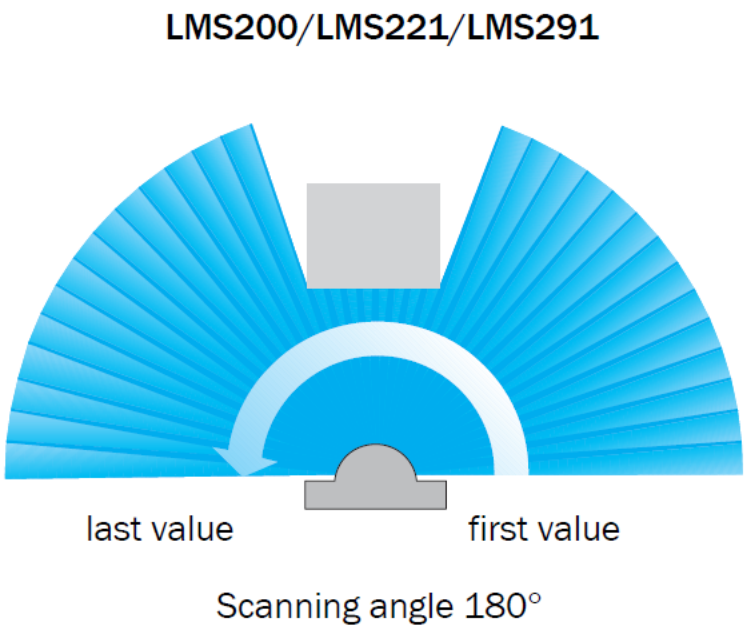
Fonte: SICK AG Waldkirch (2006).

O LMS221 consegue fazer um escaneamento de 180º, evidenciado pela Figura 71, com resolução angular de 0,25º, 0,5º ou 1º e alcance de 80 m, com correção de neblina, sendo seu alcance máximo com 10 % de refletividade equivalente a 30 m. O tempo de resposta mínimo do LIDAR é de 13 ms, sendo possível detectar praticamente qualquer formato de objeto. Seu erro sistemático é de ±35 mm e o erro estatístico é de ± 10 mm. Seu peso é de aproximadamente 9 kg, sendo alimentado por 24 V e consumindo uma potência média de 30 W. A temperatura do ambiente de operação pode variar entre -30º C e 50 º C, o que o torna adequado mesmo em lugares de temperatura hostil. O laser utilizado (luz infravermelha de 905 nm) é classificado como classe 1 (EM/IEC 60825-1), não sendo prejudicial à visão humana. (SICK AG WALDKIRCH, 2006)

Os parâmetros de configuração do LIDAR são especificados pela CPU que transmite serialmente para o LIDAR, conforme detalhado sobre essa conexão. O LIDAR envia

para a CPU os dados da imagem obtida. A taxa de transmissão é definida pela CPU e pode ser 9,6 kBd ou 19,2 kBd para a comunicação RS232. No caso, da comunicação RS422, além dessas taxas mencionadas, pode-se ainda transmitir dados em 38,4 kBd e 500 kBd. O padrão estabelecido foi de 9,6 kBd na comunicação RS232.

Figura 71 - Direção de transmissão e máximo ângulo do escaneamento horizontal.



Fonte: SICK AG Waldkirch (2006).

5.6 Dispositivos Não-Utilizados

Haviam alguns dispositivos presentes no Nanook que não estavam sendo utilizados para seu funcionamento, conforme exposto no decorrer desta seção. Sua presença no circuito eletrônico do robô deve-se à constante alteração e aperfeiçoamento do Nanook ao longo dos anos. Isso ocasionou a substituição de dispositivos que antes eram utilizados por novos, sem que isso significasse a retirada da peça, implicando apenas na mudança de ligações.

5.6.1 Driver de motor Parallax

O driver de motor *Parallax* é um componente obsoleto que está na caixa cinza onde fica a CPU. Conforme identificado na inspeção de hardware, ele não está mais sendo utilizado.

Apesar disso, ainda existem fios conectados a ele e, portanto, ele foi incluído no esquemático dos circuitos do robô. Há dois fios (um azul e um vermelho) vindo da placa de conversão de potência e há seis outros fios saindo do driver *parallax*. Desses fios de saída, dois são vermelhos, dois verdes e dois pretos, mas todos eles se tornam fios brancos após passar por um conector. Por fim, eles vão para uma junção de fios que não está sendo utilizada.

5.6.2 Encoders Frontais

Como explicado na seção sobre o controle dos motores CC, somente dois *encoders* na parte de trás do Nanook estão sendo utilizados. O dois *encoders* frontais não estão conectados ao restante do sistema.

5.6.3 Conector para hub (identificado como "For hub")

Do conector amarelo presente logo após a placa de conversão de potência, que tem como saída 5 V, saem também dois fios adicionais, um preto e um vermelho, sendo ambos de 5 V (mesmo o fio preto, que normalmente é usado como terra ou neutro). O fio preto se junta com outro fio preto, que dessa vez é realmente um fio neutro, e vai para um conector etiquetado com "*for hub*" (para hub). Esse conector não estava sendo utilizado.

A escolha das cores dos fios não foi a mais recomendável, uma vez que pode causar confusão por não estar em conformidade com o padrão comumente utilizado por engenheiros eletricistas.

6 DOCUMENTAÇÃO DO SOFTWARE

A documentação do software, tanto do código embarcado como o do programa de interface com o usuário, foi elaborada de modo que pessoas interessadas possam ser capazes de entender o código e replicá-lo. Além disso, com a descrição completa do código, a proposta de melhorias é facilitada.

Essa documentação pode ser estudada em paralelo com a análise do hardware. Dessa maneira, futuras equipes que trabalharem com o Nanook ou mesmo tiverem interesse nele podem ter um melhor entendimento da interface do hardware e do software, de modo a terem um começo mais ameno, o que promove o desenvolvimento de futuras aplicações.

6.1 Software do Programa do Computador

Como C# é uma linguagem orientada ao objeto, o código foi organizado em classes e métodos. Na Figura 72, encontra-se um diagrama de classes que resume a estrutura global do código.

Figura 72 - Diagrama de classes do programa do Nanook de interface com o usuário.

Fonte: Elaborado pelo autor.

Cada bloco nesse diagrama representa alguma classe, *struct* ou *enum* e várias destas classes representam *Forms* que são usados na interação com o usuário. As próximas seções explicarão com detalhes cada um desses blocos.

6.1.1 Form

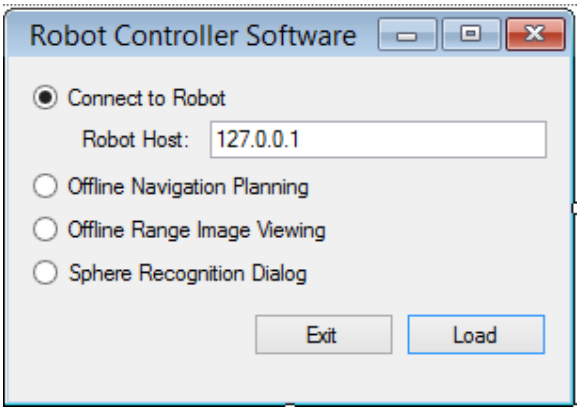
A classe *Form* representa uma janela ou caixa de diálogo utilizada na interface GUI (*Graphical User Interface*) da aplicação. Basicamente, ela permite a adição de botões, caixas de texto, tabulações e outros componentes gráficos importantes no desenvolvimento de um software.

No caso específico do software controlador do Nanook, foram criadas as classes *ConnectionDialog* (Diálogo de Conexão), *Control* (Controle), *Debug* (Depurador), *DroneCommandList* (Lista de Comandos do Drone), *MainFrame* (Quadro Principal), *RangeImageViewer* (Visualizador do Alcance da Imagem), *ScanParametersViewer* (Visualizador dos Parâmetros de Escaneamento) e *SphereRecognitionView* (Visualizador do Reconhecimento de Esferas) que herdam de *Form* e, portanto, possuem todas as suas propriedades e podem ter novas características adicionadas.

6.1.1.1 Form Connection_Dialog

Este *Form*, destacado na Figura 73, é a primeira camada de interação com o robô. Quando o usuário inicia o programa, esta janela é aberta e então é possível escolher o modo de funcionamento que será utilizado dentre as seguintes opções: *Connect to Robot* (Conectar com o Robô), *Offline Navigation Planning* (Planejamento de Navegação Offline), *Offline Range Image Viewing* (Visualização de Imagem Offline) e *Sphere Recognition Dialog* (Diálogo de Reconhecimento de Esfera). A escolha de um destes modos leva a uma experiência diferente depois que o botão *Load* (Carregar) é clicado.

Figura 73 - Form Connection_Dialog.



Fonte: Acervo do autor.

6.1.1.1.1 Conectar com o Robô (*Connect to Robot*)

Por padrão, a opção ‘Conectar com o Robô’ está selecionada, com o endereço de IP padrão 127.0.0.1, que é o endereço local (*localhost*). Entretanto, é possível especificar um IP diferente se necessário (para conectar este software ao robô remotamente). Quando o botão *Load* é clicado, o código entra em seu *loop* principal, o *RobotController*, e a caixa de diálogo *Control* aparece na tela.

6.1.1.1.2. Planejamento de Navegação Offline (*Offline Navigation Planning*)

Este modo possui um problema que impede o seu funcionamento, por conta de erros no código. Portanto, ele está sem funcionalidade.

6.1.1.1.3 Visualização de Imagem Offline (*Offline Range Image Viewing*)

Este modo é utilizado para carregar uma “*RangeImage*”, que é a imagem de um escaneamento anterior. Após carregar este arquivo, é possível dar zoom na imagem e escolher entre quatro possíveis estilos de visualização: ‘nuvem de pontos’, ‘colorido por altura’, ‘colorido por distância’ ou ‘imagem plana’. Todavia, apenas as opções ‘colorido por distância’ e ‘colorido por altura’ estavam funcionando corretamente.

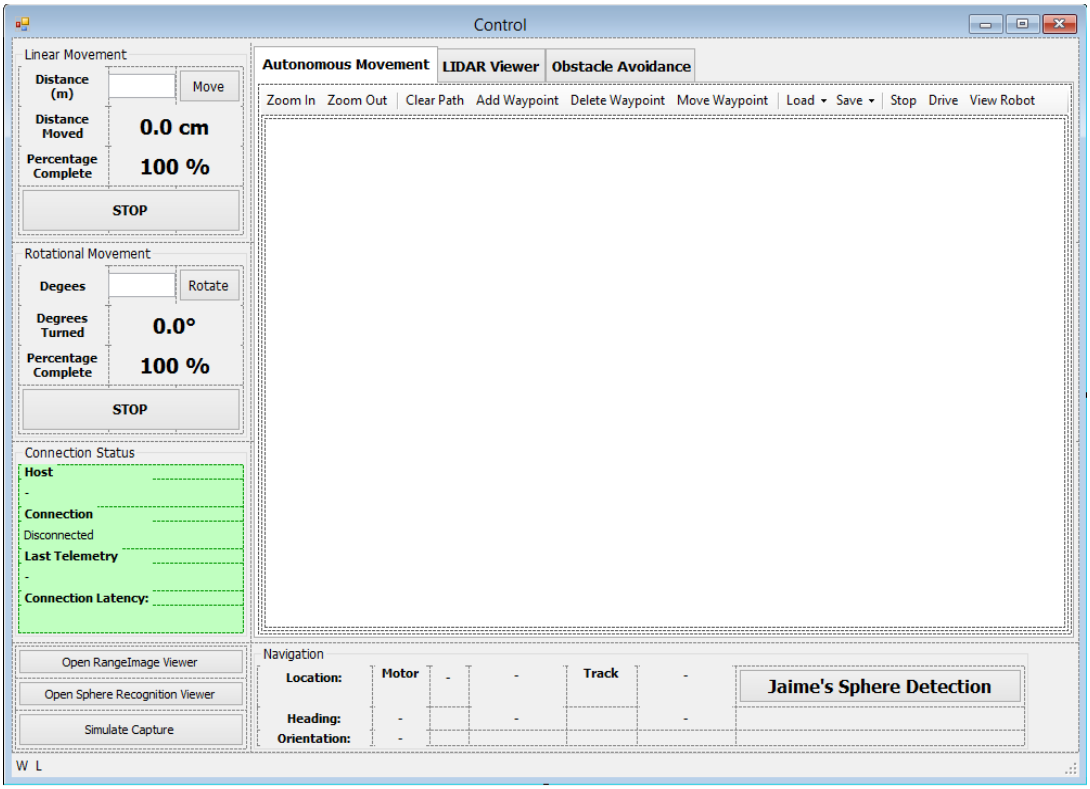
6.1.1.1.4. Diálogo de Reconhecimento de Esfera (*Sphere Recognition Dialog*)

Esta caixa de diálogo é um modo off-line que carrega a "*RangeImage*" que foi adquirida a partir da realização de um escaneamento anterior. O Reconhecimento de Esfera possui um algoritmo para encontrar formatos esféricos nesta imagem. Há também algumas caixas de texto que permitem que o usuário escolha as diferentes configurações para o algoritmo e um botão para testar no último escaneamento do Nanook.

6.1.1.2. Form Control

No *Form Control*, destacado na Figura 74, é possível mover o robô usando o movimento linear ou o movimento rotacional. Inicialmente, por padrão, os campos "*Distance Moved*" (Distância Percorrida), "*Degree Turned*" (Ângulo Rotacionado) e "*Percentage Complete*" (Porcentagem Concluída) são configurados conforme a Tabela 11.

Figura 74 - Form Control.



Fonte: Acervo do autor.

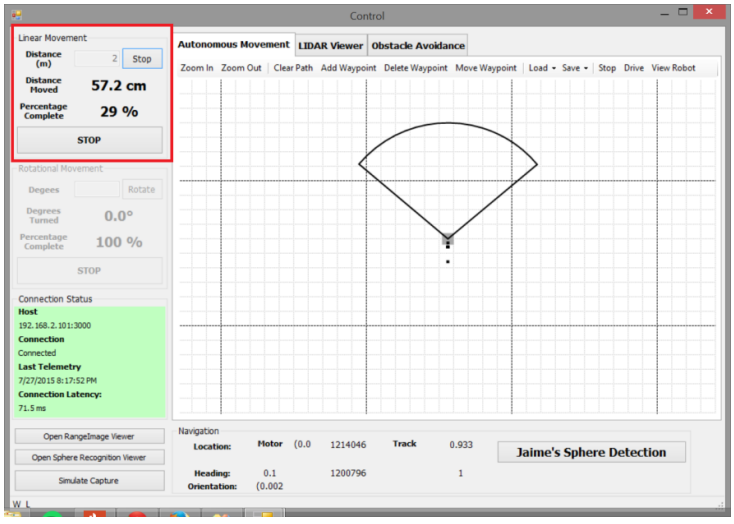
Tabela 11 - Padrão para o os campos do *Form Control*.

Antes de mover ou girar	
Movimento Linear	**Movimento Rotacional**
Distance Moved = 0.0 cm	*Degree Turned* = 0.0°
Percentage Complete = 100%	*Percentage Complete* = 100%
Depois de mover ou girar	
Movimento Linear	**Movimento Rotacional**
Distance Moved = 0.0 cm	*Degree Turned* = 0.0°
Percentage Complete = 0%	*Percentage Complete* = 0%

Fonte: Elaborado pelo autor.

No movimento linear, o usuário pode escrever o valor da distância que ele quer mover o robô na caixa de texto do lado do botão '*Move*' (Mover). Ao clicar neste botão, o Nanook começa o seu movimento e os valores de "*Distance Moved*" e "*Percentage Complete*" no Form Control mudam enquanto o usuário vê algo como o ilustrado na Figura 75.

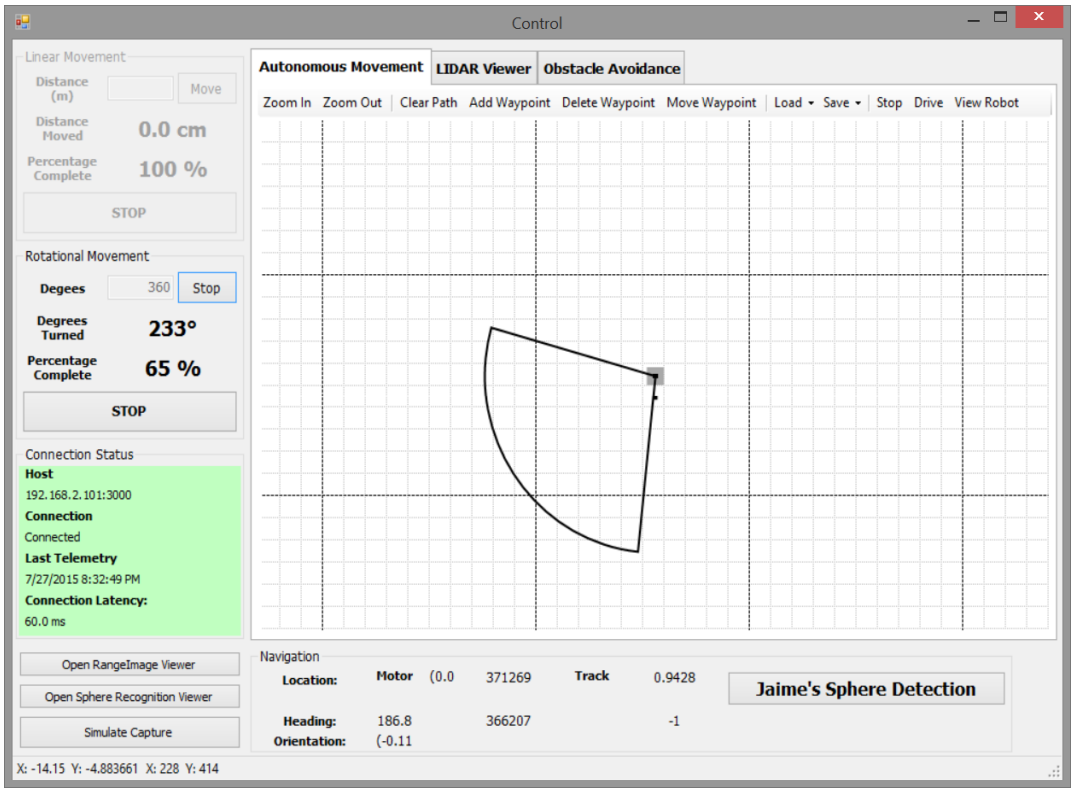
Figura 75 - Alteração do GroupBox de movimento linear durante o deslocamento do Nanook.



Fonte: Acervo do autor.

O usuário pode observar o mesmo comportamento no movimento rotacional. Por exemplo, a Figura 76 ilustra o momento em que o Nanook completou 65% do seu giro de 360°, o que corresponde a um ângulo rotacionado de 233º.

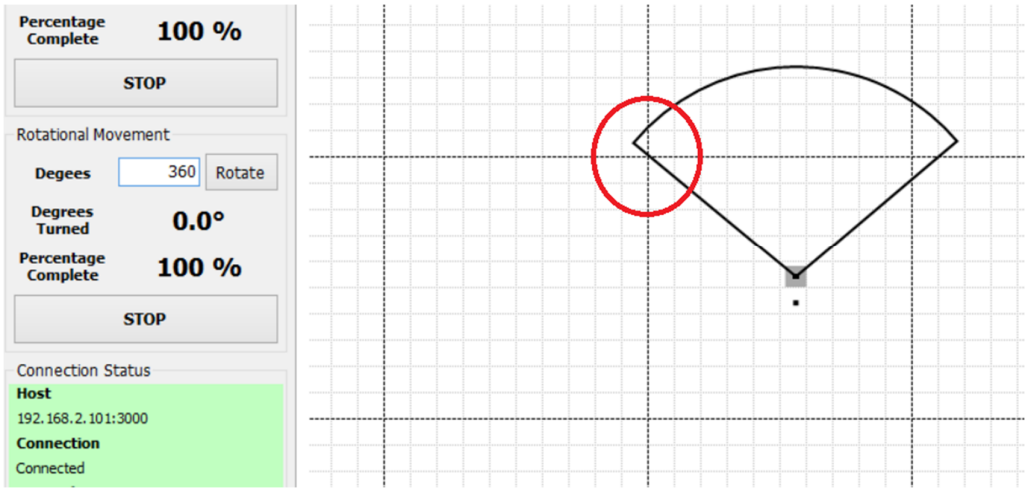
Figura 76 - Janela do Form Control durante um giro do Nanook.



Fonte: Acervo do autor.

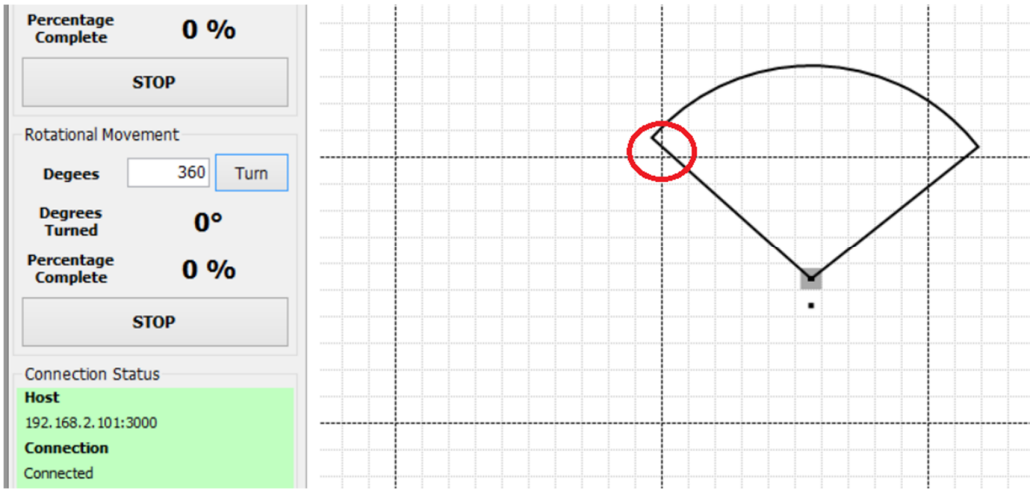
Para o movimento linear, é difícil verificar se existe algum erro de calibração somente analisando a tela. No entanto, para o movimento rotacional, é mais fácil identificar um erro evidente, porque é possível visualizar a posição do ângulo na tela. As próximas duas imagens, a Figura 77 e a Figura 78, ilustram o erro de calibração encontrado, pois exibem uma pequena diferença na posição do robô mesmo com um giro de 360º:

Figura 77 - Visualização antes do giro de 360° do Nanook.



Fonte: Acervo do autor.

Figura 78 - Visualização depois do giro de 360° do Nanook.

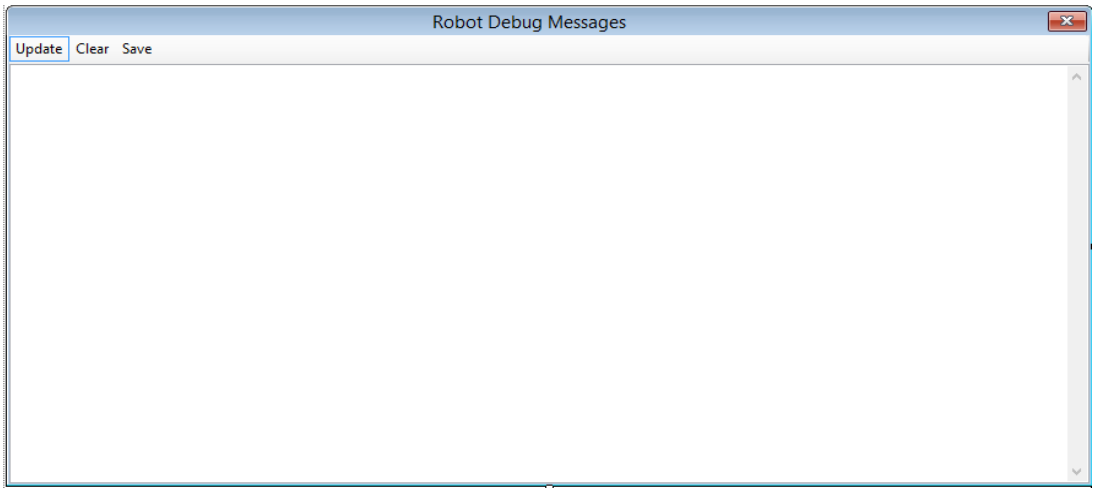


Fonte: Acervo do autor.

6.1.1.3 Form Debug

O Form *Debug* (depurador), ilustrado na Figura 79, foi usado por programadores como o objetivo de servir como uma interface para checar os dados recebidos do software embarcado e então analisar se esses dados estavam conforme esperados.

Figura 79 - Form Debug.

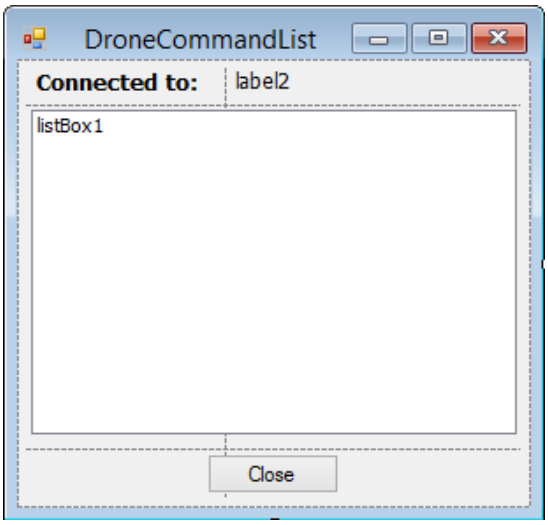


Fonte: Acervo do autor.

6.1.1.4. Form DroneCommandList

O DroneCommandList é um *Form* no programa do computador que estava sem funcionalidade. Durante o tempo de desenvolvimento do Nanook, esse *Form* provavelmente não foi finalizado, o que é evidenciado na Figura 80. Sua proposta não estava bem especificada no código, mas supõe-se que esse *Form* foi projetado, ou pelo menos arquitetado, para permitir a comunicação entre o Nanook e os seus robôs auxiliares: Pinguim 1 e 2.

Figura 80 - Form DroneCommandList.

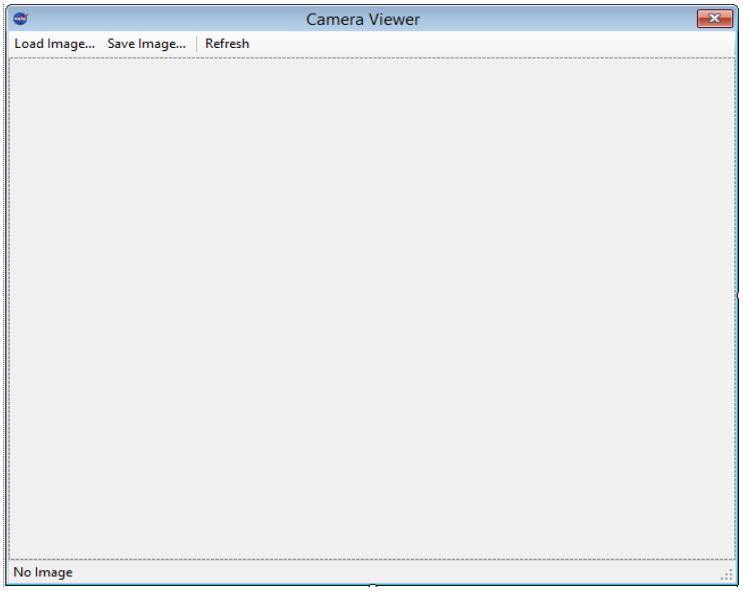


Fonte: Acervo do autor.

6.1.1.5 Form ImageViewer

O *ImageViewer* também não estava sendo utilizado. Ele provavelmente foi projetado para ajudar em uma possível aplicação de desvio de obstáculos, que acabou não sendo desenvolvida. Como o *Form ImageViewer* foi nomeado como "*Camera Viewer*" (Visualizador da Câmera), conforme pode-se visualizar na Figura 81, ele provavelmente deveria mostrar imagens da câmera em tempo real na tela.

Figura 81 - Form ImageViewer.

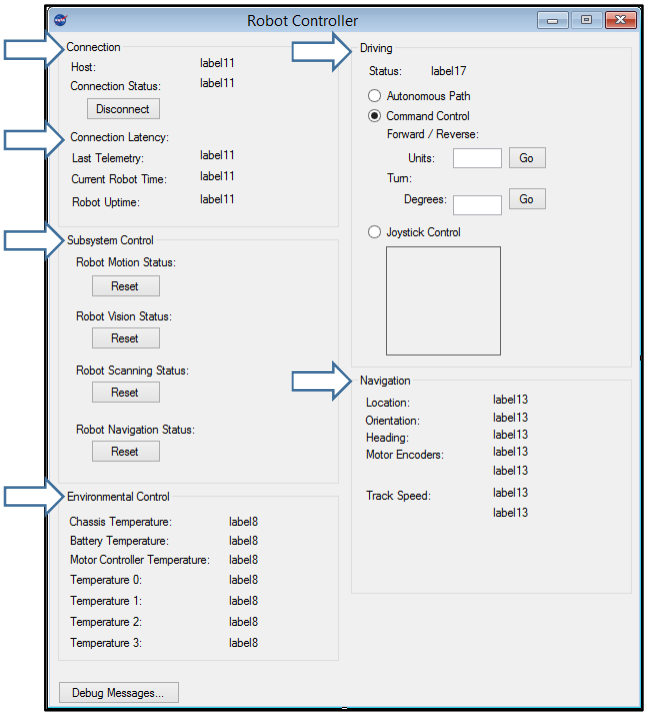


Fonte: Acervo do autor.

6.1.1.6 Form Main

O Form Main é usado para observar efeitos do ambiente no Nanook (temperatura), para checar a conexão wi-fi e sua qualidade e também para checar e controlar os movimentos linear e rotacional. Nesse Form, existem seis GroupBoxes, os quais estão destacados na Figura 82. Eles são: *Connection* (conexão), *Connection Latency* (latência da conexão), *Subsystem Control* (controle do subsistema), *Environmental Control* (controle do ambiente), *Driving* (direção) e *Navigation* (navegação).

Figura 82 – Form Main.

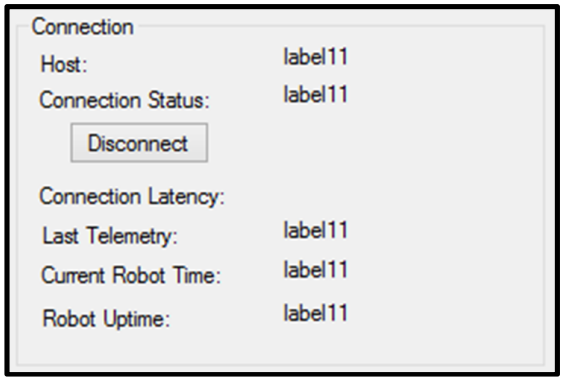


Fonte: Acervo do autor.

6.1.1.6.1 Connection

Connection, ilustrada na Figura 83, é responsável por conectar ou desconectar o programa do robô, informando ao usuário sobre o status da conexão, o endereço IP do servidor e a latência.

Figura 83 - GroupBox Connection.



Fonte: Acervo do autor.

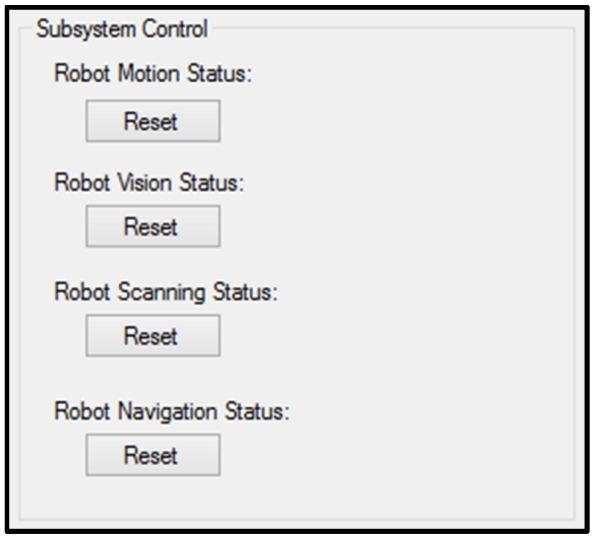
A Latência da conexão (*Connection Latency*) é responsável por informar ao usuário a qualidade da conexão, fornecendo o tempo de resposta da troca de dados entre o Nanook e o programa do computador.

6.1.1.6.2 Subsystem control

Subsystem Control é o GroupBox que contém os botões de reiniciar o movimento, a visão, o escaneamento e a navegação. No entanto, o clique nesses botões não leva a nenhum evento, pois não havia nenhuma instrução programada para essas ações.

De fato, os botões do *GroupBox* não foram implementados. Durante a inspeção do software do computador, entendeu-se que os botões foram apenas criados e mantidos sem a adição de código específico algum para eles. Apenas o nome dos botões havia sido atualizado para "*Reset*" após sua criação, conforme evidenciado na Figura 84.

Figura 84 - GroupBox Subsystem Control.



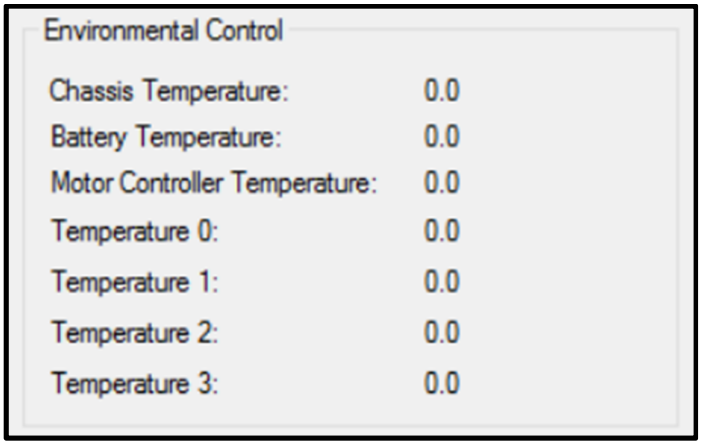
Fonte: Acervo do autor.

6.1.1.6.3. Environmental Control

As referências de temperatura para o Nanook não haviam sido implementadas. O *GroupBox Environmental Control* tem sete tipos diferentes de temperatura que não haviam sido atualizados com as medidas, conforme mostrado na Figura 85. A razão para isso é que não

havia nenhum código para processar a informação que deveria vir do Nanook e este não possuía nenhum hardware para realizar essas medições. O programa deveria reunir essas informações sobre temperatura vindas do Nanook por uma conexão wi-fi. Essa funcionalidade poderia até mesmo ser implementada usando Arduíno. O Arduíno poderia enviar a informação para uma CPU embarcada e então enviá-la para um computador por conexão wi-fi. Essa é uma oportunidade de melhoria do robô que pode vir a ser implementada.

Figura 85 - GroupBox Environmental Control.



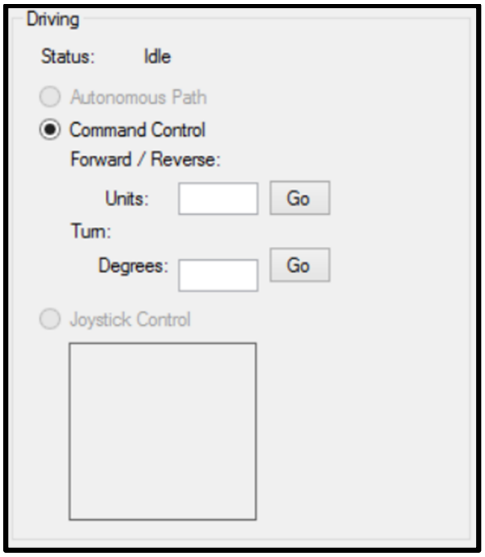
Fonte: Acervo do autor.

7.1.1.6.4. Driving

O *GroupBox Driving*, ilustrado na Figura 86, trabalha similarmente ao Movimento Linear e Rotacional do *Form Control*.

Existem alguns botões de opções que selecionam o tipo de direção. Contudo, somente o Controle por Comandos (*Command Control*) estava implementado. O caminho autônomo (*Autonomous Path*) nunca foi realmente desenvolvido e o controle por joystick (*Joystick Control*) estava desativado possivelmente porque foi projetado para trabalhar com o *ImageViewer* que poderia fornecer imagens em tempo real obtidas pelo robô.

Figura 86 - GroupBox Driving.

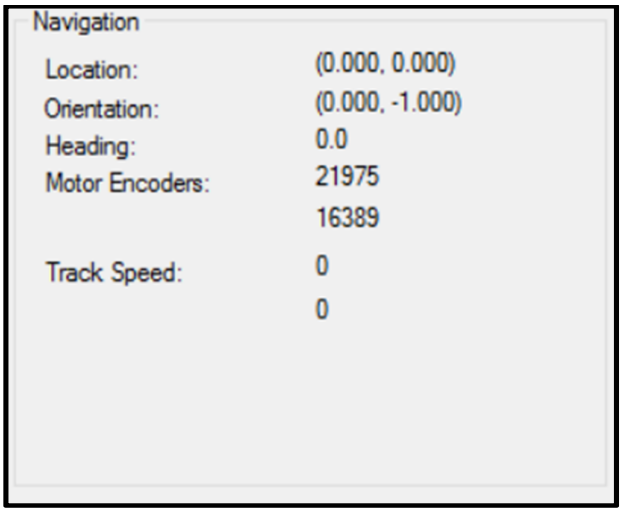


Fonte: Acervo do autor.

6.1.1.6.5. Navigation

O *GroupBox Navigation*, ilustrado na Figura 87, é relevante para informar ao usuário sobre os dados de navegação, por exemplo, as coordenadas do robô e a orientação, tanto para o movimento linear quanto rotacional no *Form Control*.

Figura 87 - GroupBox Navigation.

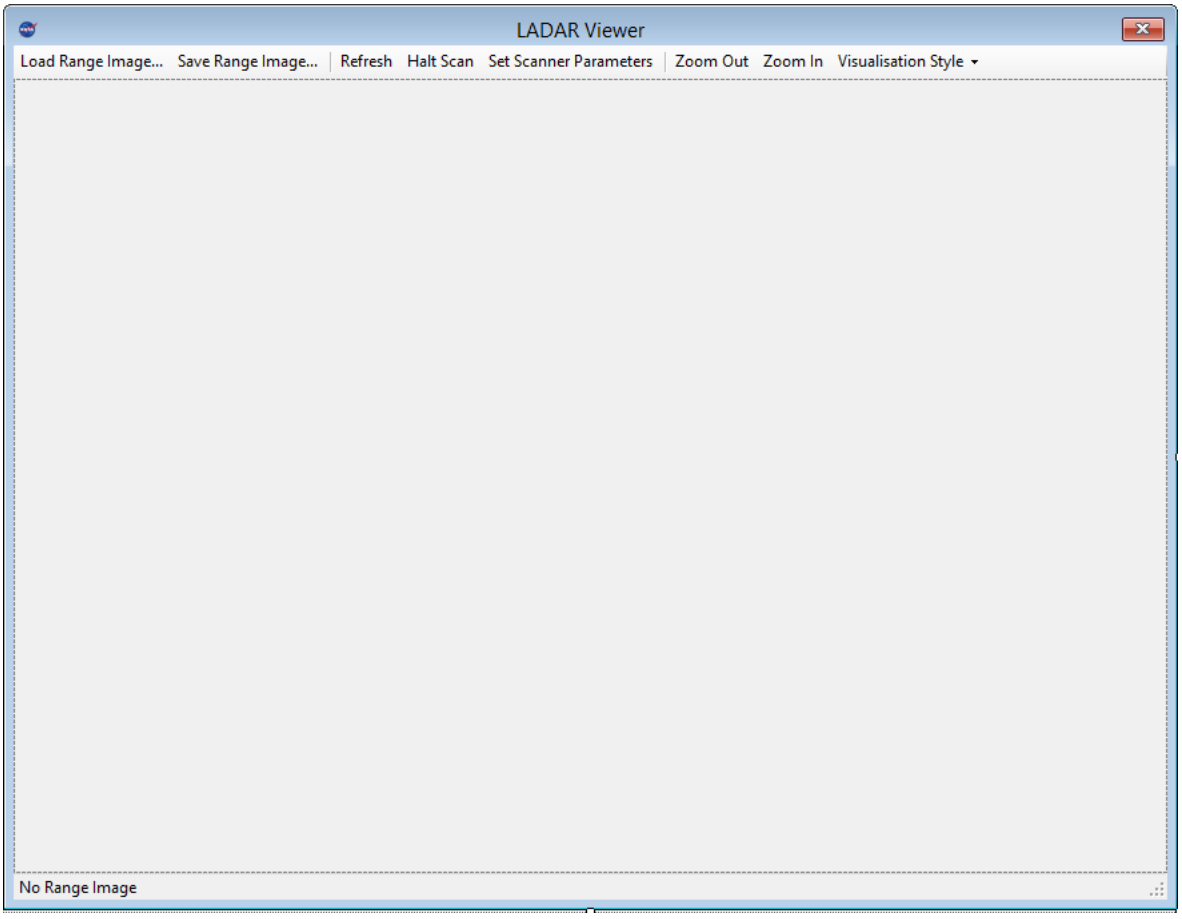


Fonte: Acervo do autor.

6.1.1.7 Form RangeImageViewer

O RangeImageViewer, evidenciado na Figura 88, é a interface responsável por mostrar aos usuários a imagem obtida pelo LIDAR. Existem oito botões nessa janela, são eles: *Load Range Image* (Carregar Imagem), *Save Range Image* (Salvar Imagem), *Refresh* (Atualizar), *Halt Scan* (Parar Varredura), *Set Scanner Parameters* (Definir Parâmetros do Escaneamento), *Zoom Out* (Diminuir), *Zoom In* (Ampliar) e *Visualisation Style* (Estilo de Visualização).

Figura 88 - Form RangeImageViewer.



Fonte: Acervo do autor.

O botão de *Load Range Image* permite ao usuário abrir uma imagem salva previamente.

O botão *Save Range Image* permite ao usuário salvar uma figura obtida do escaneamento.

O botão de *Refresh* executa o processo de escaneamento novamente.

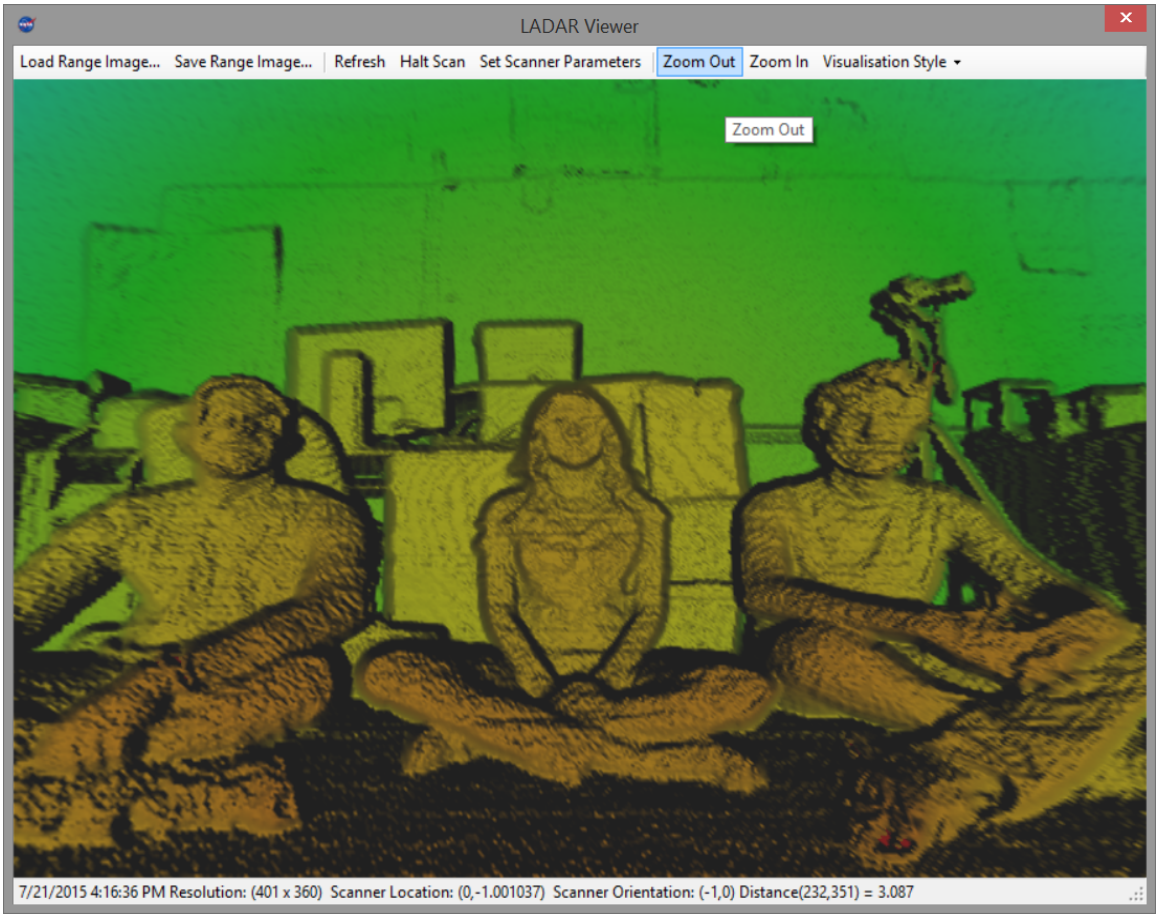
O botão de *Halt Scan* para a atividade de escaneamento do LIDAR que esteja em processo.

O botão de *Set Scanner Parameters* abre o Form chamado de ScanParametersViewer, onde o usuário pode escolher os parâmetros de escaneamento.

Os botões de *Zoom Out* e *Zoom In* permitem a interação do usuário com a imagem, aproximando-a ou afastando-a.

Na Figura 89, encontra-se um exemplo de como a janela é atualizada após o clique no botão *Refresh*, de modo a mostrar a imagem obtida pelo novo escaneamento realizado.

Figura 89 - Imagem obtida pelo escaneamento do LIDAR.

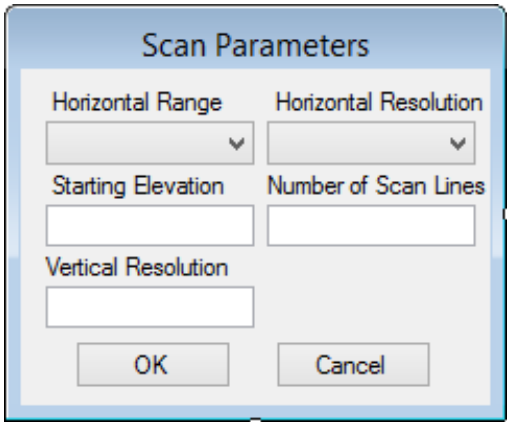


Fonte: Acervo do autor.

6.1.1.8 Form ScanParametersViewer

O ScanParametersViewer, ilustrado na Figura 90, é a interface responsável por determinar os parâmetros de escaneamento do LIDAR. Esses parâmetros são importantes para que o usuário selecione a resolução desejada. Os parâmetros são: *Horizontal Range* (alcance horizontal), *Horizontal Resolution* (resolução horizontal), *Starting Elevation* (elevação inicial), *Number of Scan Lines* (número de linhas do escaneamento) e *Vertical Resolution* (resolução vertical). Seus valores padrões estão expostos na Tabela 12.

Figura 90 - Form ScanParametersViewer.



Fonte: Acervo do autor.

Tabela 12 - Parâmetros padrão

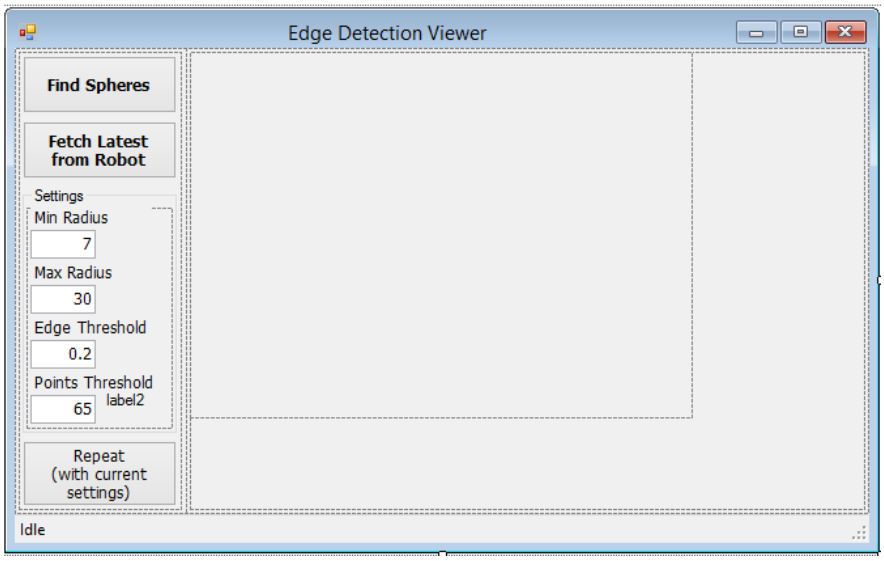
Configuração Padrão			
Parâmetros	**Valor**	**Parâmetros**	**Valor**
Alcance Horizontal	100	**Número de Linhas do Escaneamento**	360
Resolução Horizontal	0,25	**Resolução Vertical**	0,25
Elevação Inicial	-45°	**----**	**----**

Fonte: Elaborado pelo autor.

6.1.1.9 Form SphereRecognitionView

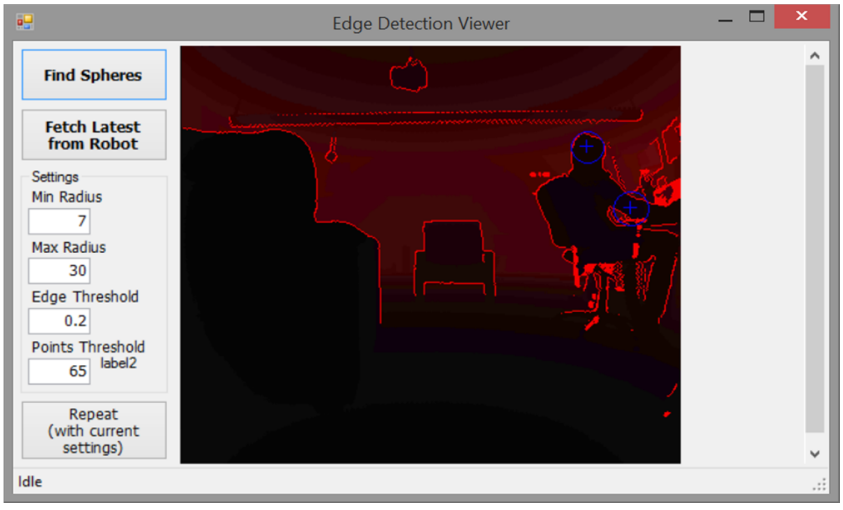
O *SphereRecognitionView*, ilustrado na Figura 91, é a interface responsável pela identificação dos robôs auxiliares do Nanook. Nessa interface, o programa pode até mesmo mostrar ao usuário a localização das esferas, circulando-as como um alvo na imagem. No exemplo da Figura 92, um círculo azul destaca na figura o que foi detectado como esfera.

Figura 91 - Form Sphere Recognition View.



Fonte: Acervo do autor.

Figura 92 - Indicação das esferas detectadas, destacadas com um círculo azul.



Fonte: Acervo do autor.

6.1.2 Classes

Uma classe é um componente do código que permite que alguém crie seu próprio tipo customizado de modo a agrupar variáveis, métodos e eventos que caracterizam o grupo. Classes, portanto, definem dados e comportamentos.

Como mencionado em (DONETPERLS, “Class”, n.d.), classes são um dos componentes mais importantes em uma linguagem orientada a objetos: “Um programa é uma abstração de uma máquina. À medida em que é executado, suas partes estão em movimento [...] constante [...], movendo-se em direção a um resultado. Essas partes são denominadas classes”.

No programa do Nanook, foram identificadas as seguintes classes: *ConnectionDialog* (Diálogo de Conexão), *Control* (Controle), *Debug* (Depuração), *DebugTimer* (Depuração de Tempo), *DebugTrace* (Depuração do Traço), *Drone* (Drone), *DroneCommandList* (Lista de Comando do Drone), *HeightMap* (Mapa de Altura), *Image* (Imagem), *ImageViewer* (Visualizador da Imagem), *MainFrame* (Quadro Principal), *Map* (Mapa), *Navigation* (Navegação), *NavigationBAK* (NavegaçãoBAK), *Path* (Caminho), *PathWaypoint* (Caminho pelo Ponto de Referência), *PointCloud* (Nuvem de Pontos), *Program* (Programa), *RangeImage* (Alcance da Imagem), *RangeImageViewer* (Visualizador do Alcance da Imagem), *Robot* (Robô), *ScanParameters* (Parâmetros do Escaneamento), *ScanParametersViewer* (Visualizador dos Parâmetros do Escaneamento), *Sphere* (Esfera), *SphereRecognition* (Reconhecimento de Esferas), *SphereRecognition2* (Reconhecimento de Esferas 2), *SphereRecognitionView* (Visualizador do Reconhecimento de Esferas). No apêndice B, essas classes estão evidenciadas em um diagrama UML.

6.1.2.1 Classe Parcial

Classes parciais são uma metodologia escolhida por programadores de projetos grandes para separar fisicamente a definição de certas classes em múltiplos arquivos fonte (DONETPERLS, “Partial”, n.d.). Cada arquivo fonte contém uma seção da definição do tipo ou método, mas todas as partes são combinadas quando a aplicação é compilada (MICROSOFT, 2015c).

Os forms do Windows são um exemplo de Classe Parcial porque não há necessidade de recriar o arquivo fonte para adicionar código a eles. O usuário pode usar essas classes em seu código sem precisar modificar o arquivo criado pelo Visual Studio.

Como o Nanook foi um projeto resultante da colaboração de diversas pessoas, classes parciais foram extensivamente usadas. Algumas das classes que foram definidas usando classes parciais estão descritas a seguir.

6.1.2.1.1 ConnectionDialog

O ConnectionDialog foi declarado em ConnectionDialog.cs e ConnectionDialog.Designer.cs, como mostrado na Tabela 13.

Tabela 13 - ConnectionDialog Class

Classe	Sintaxe	Arquivo

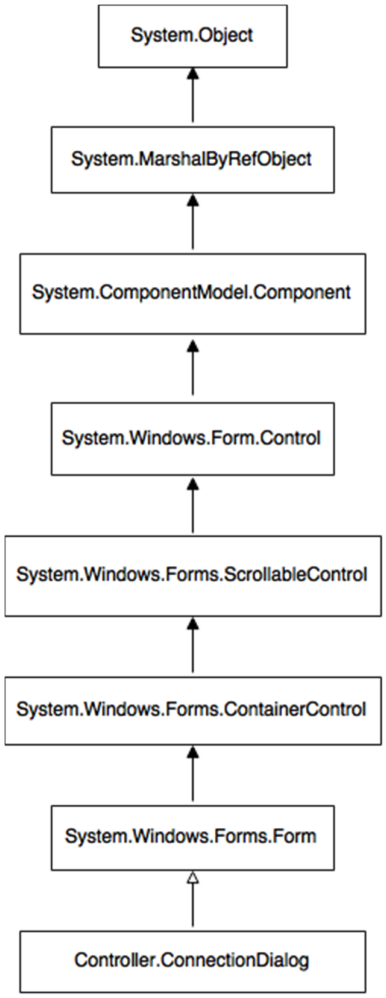
ConnectionDialog	public partial class ConnectionDialog : Form
	{
	…
	}
		ConnectionDialog.cs

	partial class ConnectionDialog
	{
	…
	}
		ConnectionDialog.Designer.cs

Fonte: Elaborado pelo autor.

A classe é estendida de Form, seguindo a herança completa da hierarquia descrita na Figura 93.

Figura 93 - Hierarquia do ConnectionDialog.



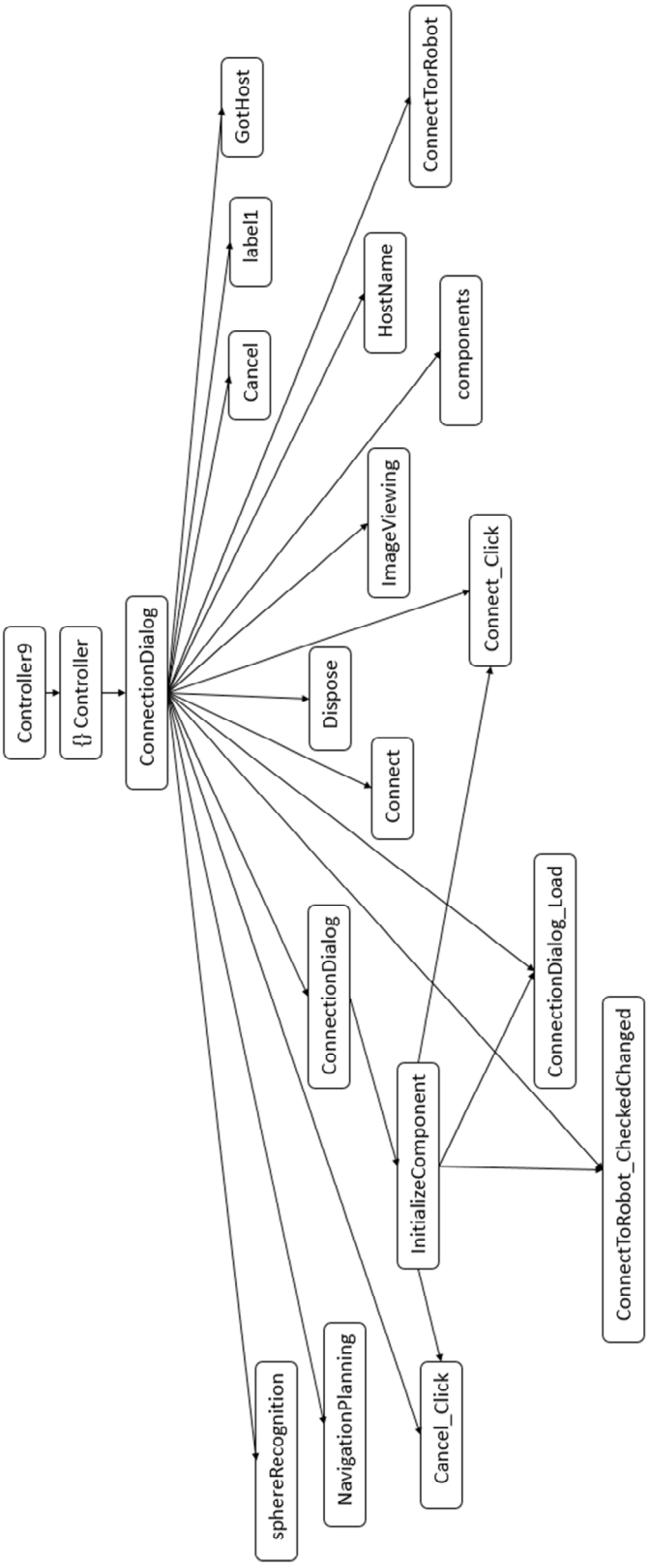
Fonte: Acervo do autor.

O termo classe estendida significa que a classe herda as características (métodos e objetos) da sua classe mãe, mas pode adicionar algumas especificidades em sua definição, o que a torna uma classe diferente.

A classe *ConnectionDialog*, cujo diagrama UML é ilustrado na Figura 94, está relacionada às opções de conectividade. Existem quatro modos disponíveis para conexão: Conectar ao Robô fornecendo seu IP, Planejamento de Navegação Off-line, Visualização de Imagem Off-line e Diálogo de Reconhecimento de Esfera.

Usando o *Event Handler*, essa classe define a ação que será tomada após o usuário selecionar uma das opções disponíveis, bem como o que acontecerá caso o botão de Cancelar (*Cancel*) ou Carregar (*Load*) sejam selecionados.

Figura 94 - Diagrama UML do ConnectionDialog.



Fonte: Acervo do autor.

6.1.2.1.2 Control

O Form Control também é uma Classe Parcial. Ele é declarado em Control.cs e Control.Designer.cs, conforme mostrado na Tabela 14.

Tabela 14 - Classe Control

Classe	Sintaxe	Arquivo

Control	public partial class Control : Form
	{
	…
	}
		Control.cs

	partial class Control
	{
	…
	}
		Control.Designer.cs

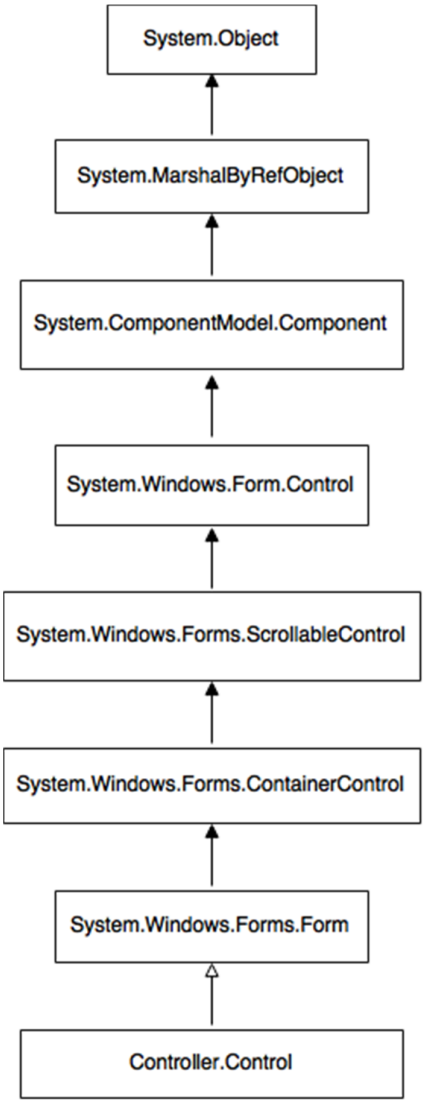
Fonte: Elaborado pelo autor.

Essa classe é estendida de Form, seguindo a herança completa da hierarquia descrita na Figura 95.

A classe *Control* é relacionada à navegação do robô. Nessa classe, o usuário pode controlar o robô e visualizar sua posição no ambiente. Existem métodos relacionados à navegação autônoma, solicitação e processamento do escaneamento do LIDAR e controle de navegação. Essa é uma das classes mais importantes no projeto, por ser onde estão os principais algoritmos.

Usando o *event handler*, essa classe define as ações que serão tomadas depois de o usuário clicar em um dos botões de sua janela.

Figura 95 - Hierarquia do Control.



Fonte: Acervo do autor.

6.1.2.1.3 Debug

Debug é declarado em Debug.cs e Debug.Designer.cs, como mostrado na Tabela 15.

Essa classe também é estendida do *Form*, seguindo a herança de hierarquia mostrada na Figura 96.

Tabela 15 - Classe Debug

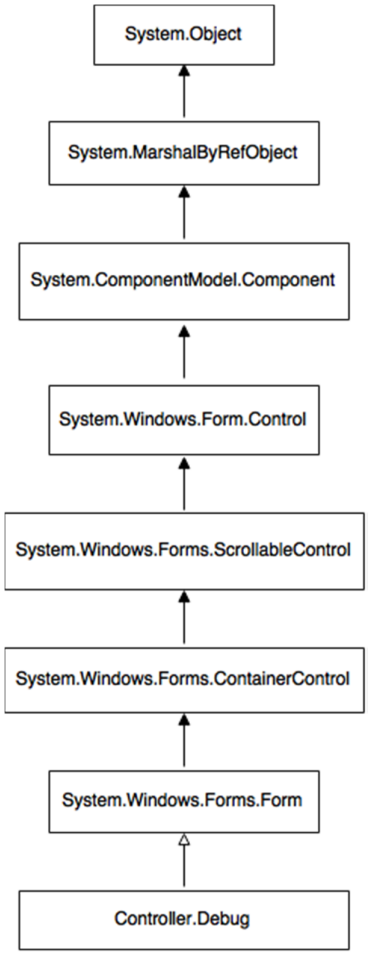
Classe	Sintaxe	Arquivo

Debug	public partial class Debug : Form
	{
	…
	}
		Debug.cs

	partial class Debug
	{
	…
	}
		Debug.Designer.cs

Fonte: Elaborado pelo autor.

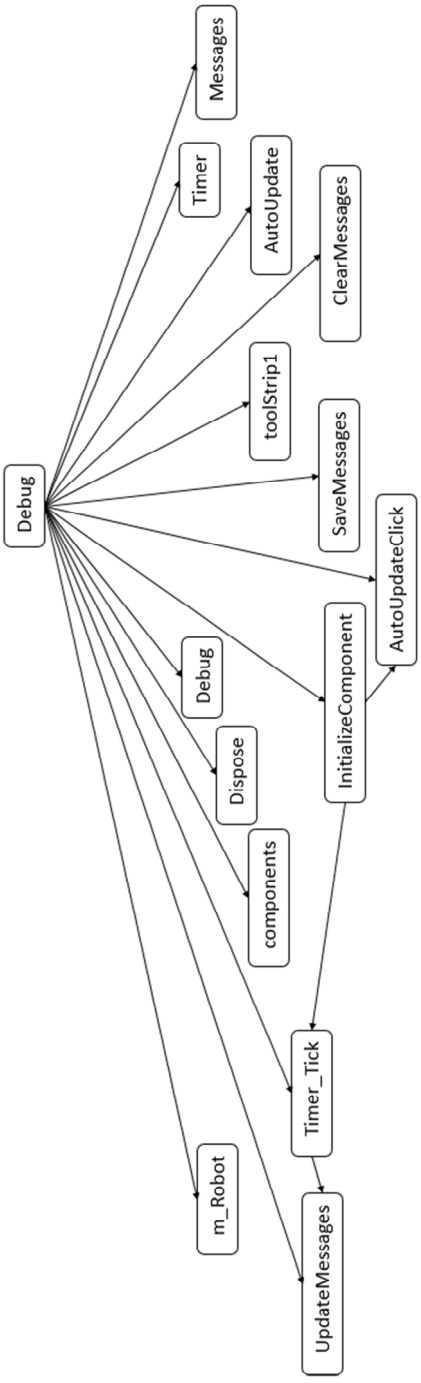
Figura 96 - Hierarquia do Debug.



Fonte: Acervo do autor.

A classe *Debug*, cujo diagrama UML é mostrado na Figura 97, foi usada para ajudar os programadores a receberem mensagens do software embarcado e mostrá-las na tela do computador quando necessário.

Figura 97 - Diagrama UML para a Classe Debug.



Fonte: Acervo do autor.

6.1.2.1.4 DroneCommandList

DroneCommandList é declarado em DroneCommandList.cs e DroneCommandList.Designer.cs, conforme mostrado na Tabela 16.

Tabela 16 - Classe DroneCommandList

Classe	Sintaxe	Arquivo

DroneCommandList	public partial class DroneCommandList : Form
	{
	…
	}
		DroneCommandList.cs

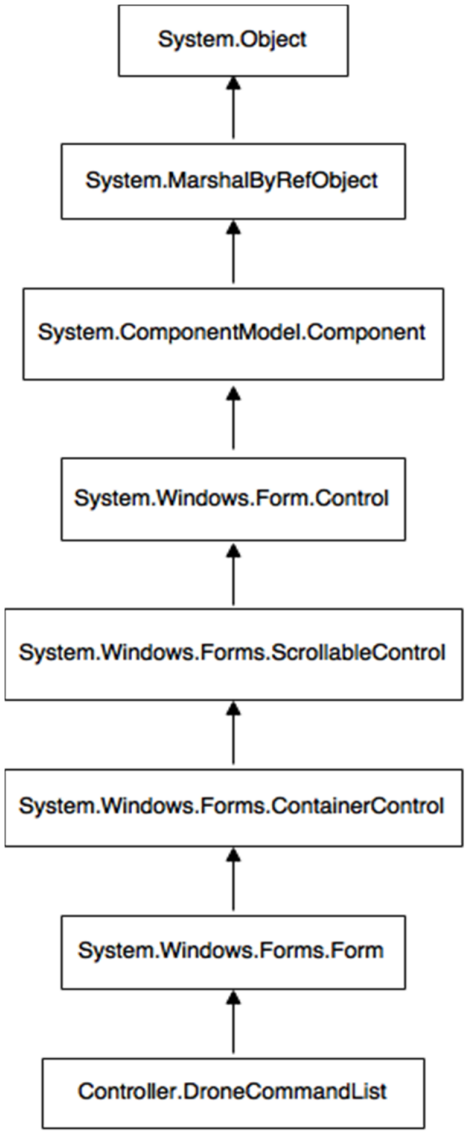
	partial class DroneCommandList
	{
	…
	}
		DroneCommandList.Designer.cs

Fonte: Elaborado pelo autor.

O Form DroneCommandList segue o mesmo padrão das classes parciais anteriores. Seu propósito é mostrar na tela o IP dos outros drones e seus respectivos comandos.

A classe é estendida de Form, seguindo a herança de hierarquia mostrada na Figura 98.

Figura 98 - Hierarquia do DroneCommandList.



Fonte: Acervo do autor.

6.1.2.1.5 ImageViewer

ImageViewer é declarado em ImageViewer.cs e ImageViewer.Designer.cs, como mostrado na Tabela 17.

Tabela 17 - Classe ImageViewer

Classe	Sintaxe	Arquivo

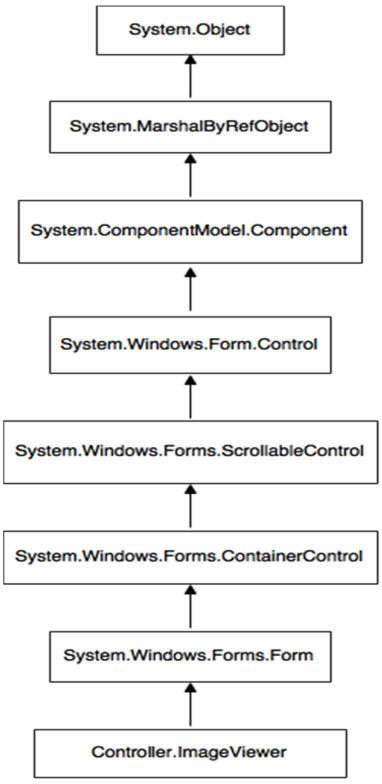
ImageViewer	public partial class ImageViewer : Form
	{
	…
	}
		ImageViewer.cs

	partial class ImageViewer
	{
	…
	}
		ImageViewer.Designer.cs

Fonte: Elaborado pelo autor.

A classe é estendida de Form, seguindo a herança de hierarquia mostrada na Figura 99.

Figura 99 - Hierarquia do ImageViewer.



Fonte: Acervo do autor.

6.1.2.1.6 MainFrame

O MainFrame é declarado em Main.cs e Main.Designer.cs, como mostrado na Tabela 18.

Tabela 18 - Classe MainFrame

Classe	Sintaxe	Arquivo

MainFrame	public partial class MainFrame : Form
	{
	…
	}
		Main.cs

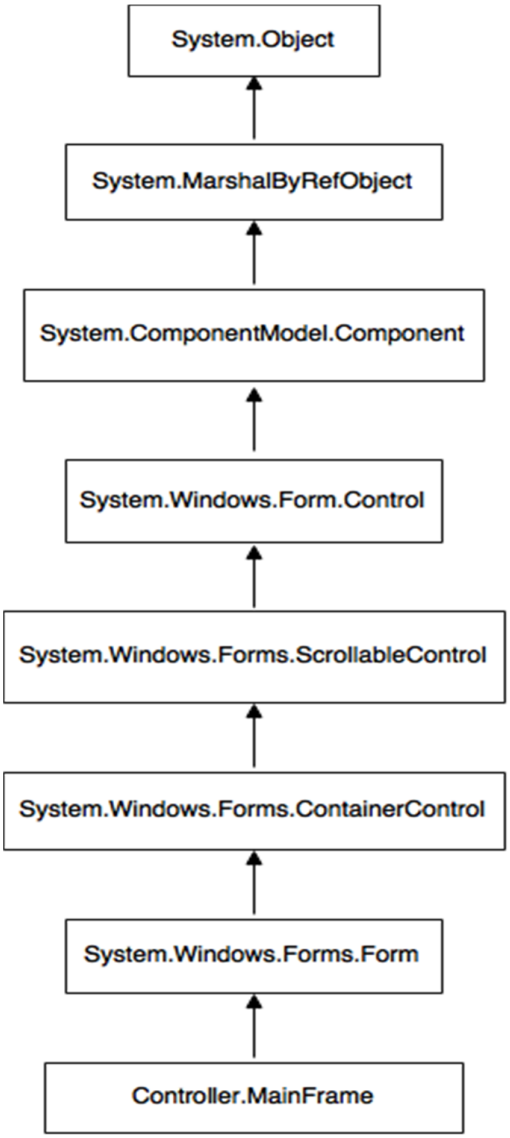
	partial class MainFrame
	{
	…
	}
		Main.Designer.cs

Fonte: Elaborado pelo autor.

Essa classe é uma extensão do Form e segue a herança de hierarquia mostrada na Figura 100.

O *Form MainFrame* é uma classe parcial com informações sobre o robô e também sobre o espaço de controle, por exemplo, a temperatura do ambiente, velocidade do robô, status da conexão e direção por joystick, autônomo ou movimento por comandos de seguir/girar.

Figura 100 - Hierarquia do MainFrame.



Fonte: Acervo do autor.

6.1.2.1.7 RangeImageViewer

RangeImageViewer é declarado em RangeImageViewer.cs e RangeImageViewer.Designer.cs, conforme mostrado na Tabela 19.

Tabela 19 - Classe RangeImageViewer

Classe	Sintaxe	Arquivo

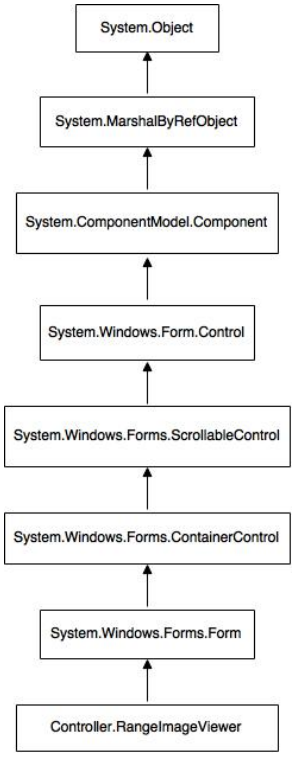
RangeImageViewer	public partial class RangeImageViewer : Form
	{
	…
	}
		RangeImageViewer.cs

	partial class RangeImageViewer
	{
	…
	}
		RangeImageViewer.Designer.cs

Fonte: Elaborado pelo autor.

Essa classe é estendida de *Form*, seguindo a herança de hierarquia mostrada na Figura 101.

Figura 101 - Hierarquia do RangeImageViewer.



Fonte: Acervo do autor.

O *RangeImageViewer* é uma classe parcial responsável por construir e mostrar a imagem 3D obtida a partir dos dados do LIDAR, usando como auxílio a classe Canvas.

6.1.2.1.8 ScanParametersViewer

ScanParameterViewer é declarado em ScanParameterViewer.cs e ScanParameterViewer.Designer.cs, como mostrado na Tabela 20.

Tabela 20 - Classe ScanParametersViewer

Classe	Sintaxe	Arquivo

ScanParameter Viewer	public partial class ScanParametersViewer
	: Form
	{
	…
	}
		ScanParameterViewer.cs

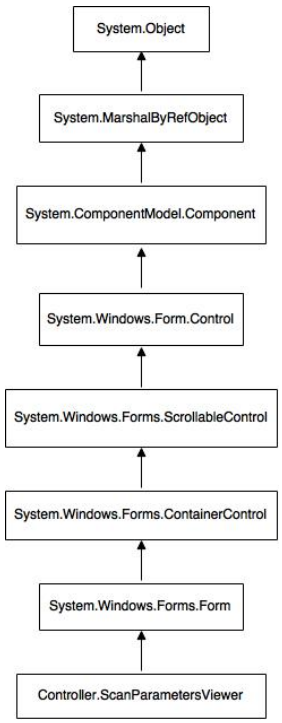
	partial class ScanParametersViewer
	{
	…
	}
		ScanParameterViewer.Designer.cs

Fonte: Elaborado pelo autor.

Essa classe é estendida do *Form*, seguindo a herança de hierarquia mostrada na Figura 102.

O *ScanParametersViewer* é uma classe parcial que ajuda o usuário a selecionar os parâmetros de escaneamento.

Figura 102 - Hierarquia do ScanParametersViewer.



Fonte: Acervo do autor.

6.1.2.1.9 SphereRecognitionView

SphereRecognitionView é declarado em SphereRecognitionView.cs e SphereRecognitionView.Designer.cs, conforme mostrado na Tabela 21.

Tabela 21 - Classe SphererRecognitionView

Classe	Sintaxe	Arquivo

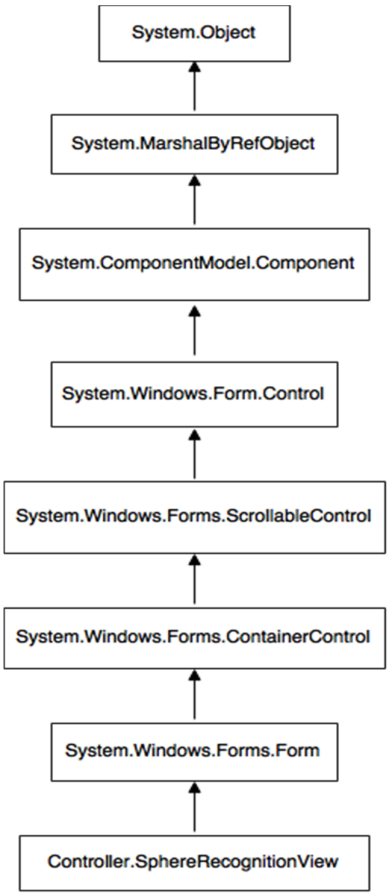
SphereRecognitionView	public partial class SphereRecognitionView : Form
	{
	…
	}
		SphereRecognitionView.cs

	partial class SphereRecognitionView
	{
	…
	}
		SphereRecognitionView.Designer.cs

Fonte: Elaborado pelo autor.

Essa classe é estendida do *Form*, seguindo a herança de hierarquia mostrada na Figura 103.

Figura 103 - Hierarquia do SphereRecognitionView.



Fonte: Acervo do autor.

O *Form SphereRecognitionView* é uma classe parcial que é responsável por detectar os objetos com formato esférico. Nessa classe parcial e nas classes *SphereRecognition* e *SphereRecognition2*, encontra-se o código que objetiva encontrar e identificar os robôs auxiliares, Pinguim 1 e Pinguim 2.

6.1.2.2 Classe Estática

O *Controller* tem uma classe estática: *Program* (Programa). Ela é usada como "uma unidade de organização para métodos não associados com objetos particulares" (MORTENSEN e RASMUSSEN, 2008) Isso significa que, usando essa classe, criar objetos para usar os métodos relacionados não é necessário porque pode-se usar os métodos diretamente

com o nome da classe. Portanto, classes estáticas não são instanciadas, o que pode torná-las úteis para operar com parâmetros sem haver necessidade de nenhum campo de instância interno (MICROSOFT, 2015e).

6.1.2.2.1 Program

O *Program* está relacionado à definição do que acontece quando o programa inicia. Após o diálogo de conexão inicial aparecer, ele redireciona para a ação correspondente requerida.

6.1.2.3 Classe Selada

O modificador *sealed* em uma classe implica que aquela classe não pode ser usada como base para outras. O termo "sealed" impede que outras classes possam ter qualquer herança daquela classe. Um exemplo dessa classe é *Settings* no software do computador.

6.1.3 Structs

Structs são convenientes para aglomerar grupos pequenos de variáveis inter-relacionadas porque podem armazenar os valores de um grupo customizado de tipos. Uma *struct* pode ser usada em vez de criar-se uma classe inteira apenas para um pequeno número de aplicações. Como descrito em (DONETPERLS, "Struct", n.d.), "uma *struct* armazena seus dados em seus tipos. Não há alocação em separado. *Structs* comumente ficam na pilha de avaliação. Todo programa usa *structs* simples. Todos os tipos (int, bool, char) são naturalmente *structs*".

O *Controller* tem três *structs*: *MapCell*, *Message* e *Telemetry*.

6.1.3.1 MapCell

A struct *MapCell* é definida em Map.cs (declarada como *public struct* MapCell – uma struct pública) e seus membros (do tipo MapCell) são: Constructors e Fields.

6.1.3.2 Telemetry

Os membros da struct *Telemetry* são listados na Tabela 22.

Tabela 22 - Membros de Telemetry

Nome do Membro		
ClimateStatus	NavigationStatus	Temperature0
CurrentTime	PathDataLength	Temperature1
Heading	Position	Temperature2
HeaterState0	PrimaryAxis	Temperature3
HeaterState1	ScanDataLength	Temperature4
HeaterState2	StartTime	Temperature5
HeaterState3	TargetTemperature0	Temperature6
HeaterState4	TargetTemperature1	Temperature7
HeaterState5	TargetTemperature2	TrackSpeed0
HeaterState6	TargetTemperature3	TrackSpeed1
HeaterState7	TargetTemperature4	TrackTicks0
ImageDataLength	TargetTemperature5	TrackTicks1
MapDataLength	TargetTemperature6	VisionStatus
MotionStatus	TargetTemperature7	-

Fonte: Elaborado pelo autor.

Essa struct é declarada por:

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct Telemetry
```

6.1.3.3 Message

A struct *Message* é declarada por:

```
struct Message
```

e seus membros são: MessageId e Length.

6.1.4 Enum

O termo *Enum* declara uma enumeração (MICROSOFT, 2015b) que é uma lista de um conjunto de constantes nomeadas. O Controller tem duas enumerações: *TelemetryStatus* e *WaypointScanMode*.

TelemetryStatus tem os seguintes membros: *Driving*, *Idle*, *MapUpdated*, *Navigating*, *PathUpdated* e *Scanning*.

WaypointScanMode tem os membros *Full* e *None*.

7.1.5 Documentação em HTML

Complementando a compreensão do código, uma documentação foi desenvolvida. A maior parte do código foi coberta e documentada em arquivos HTML (usando comentários XML), conforme está resumido na Tabela 23 a seguir.

O único namespace no software analisado é o Controller. O controller tem, como já descrito, dezoito classes, duas *structs* e três enumerações (*enum*).

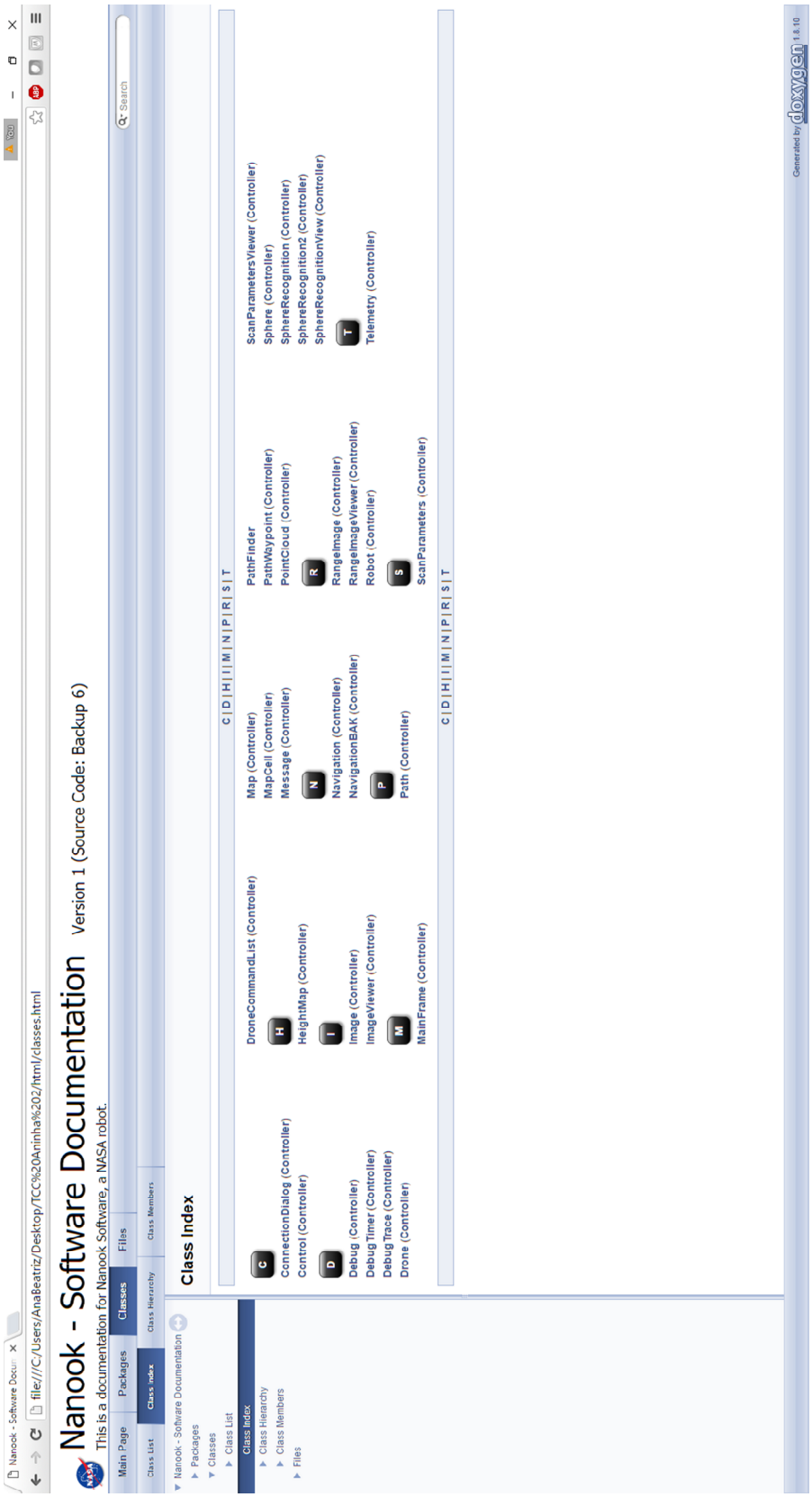
Tabela 23 - Elementos compilados

Tipo	**Elementos Compilados**	
	Com comentário XML	**Total**
Namespaces	0	1
Classes	7	18
Structs	2	2
Enumerações	3	4
Métodos	76	88
Propriedades	41	44
Variáveis (*fields*)	10	97
Constantes	0	16

Fonte: Elaborado pelo autor.

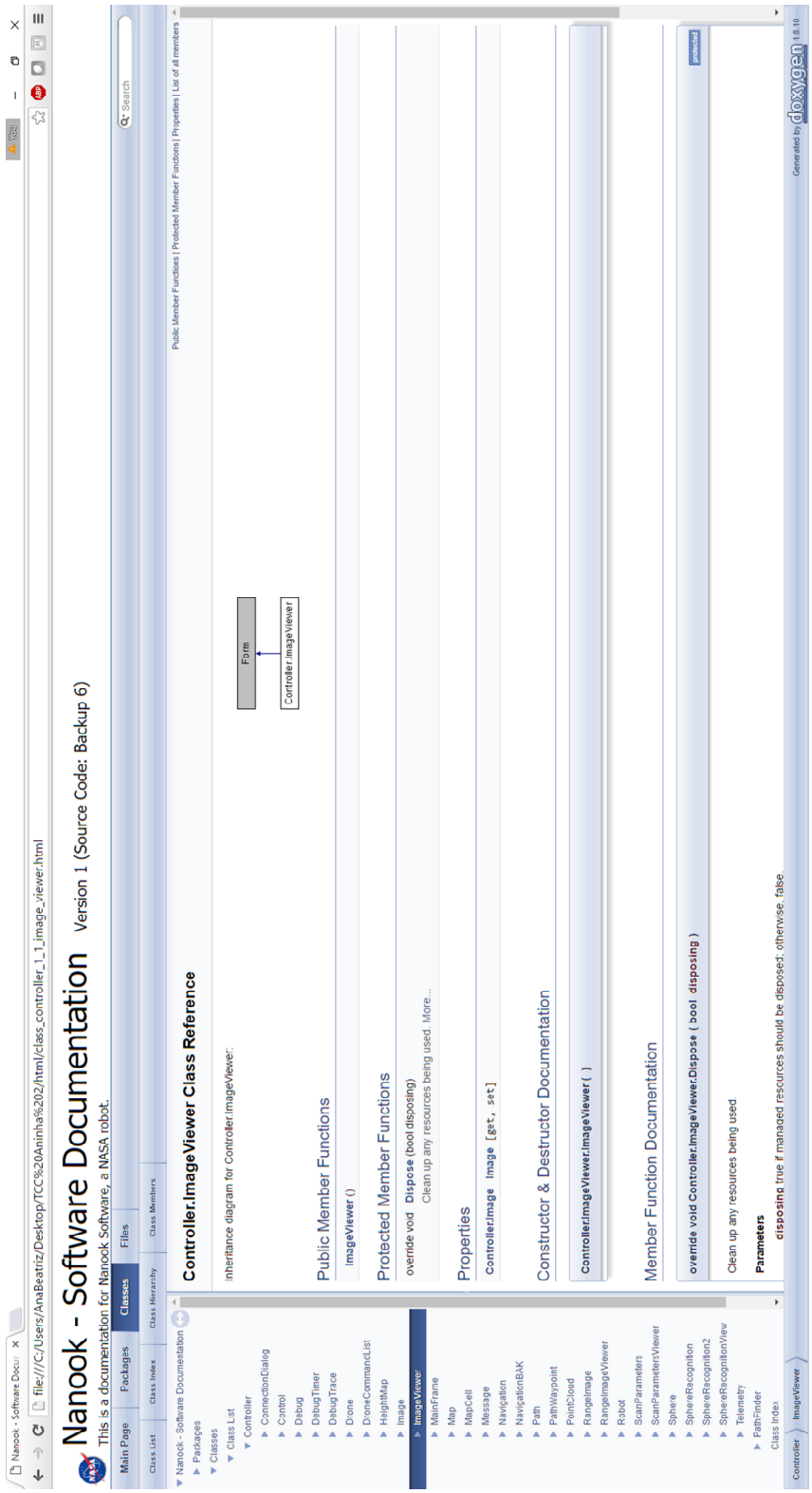
Abaixo, na Figura 104, na Figura 105 e na Figura 106, seguem algumas das visualizações em HTML possíveis, que foram geradas por essa documentação descrita.

Figura 104 - Visualização das classes como índice.



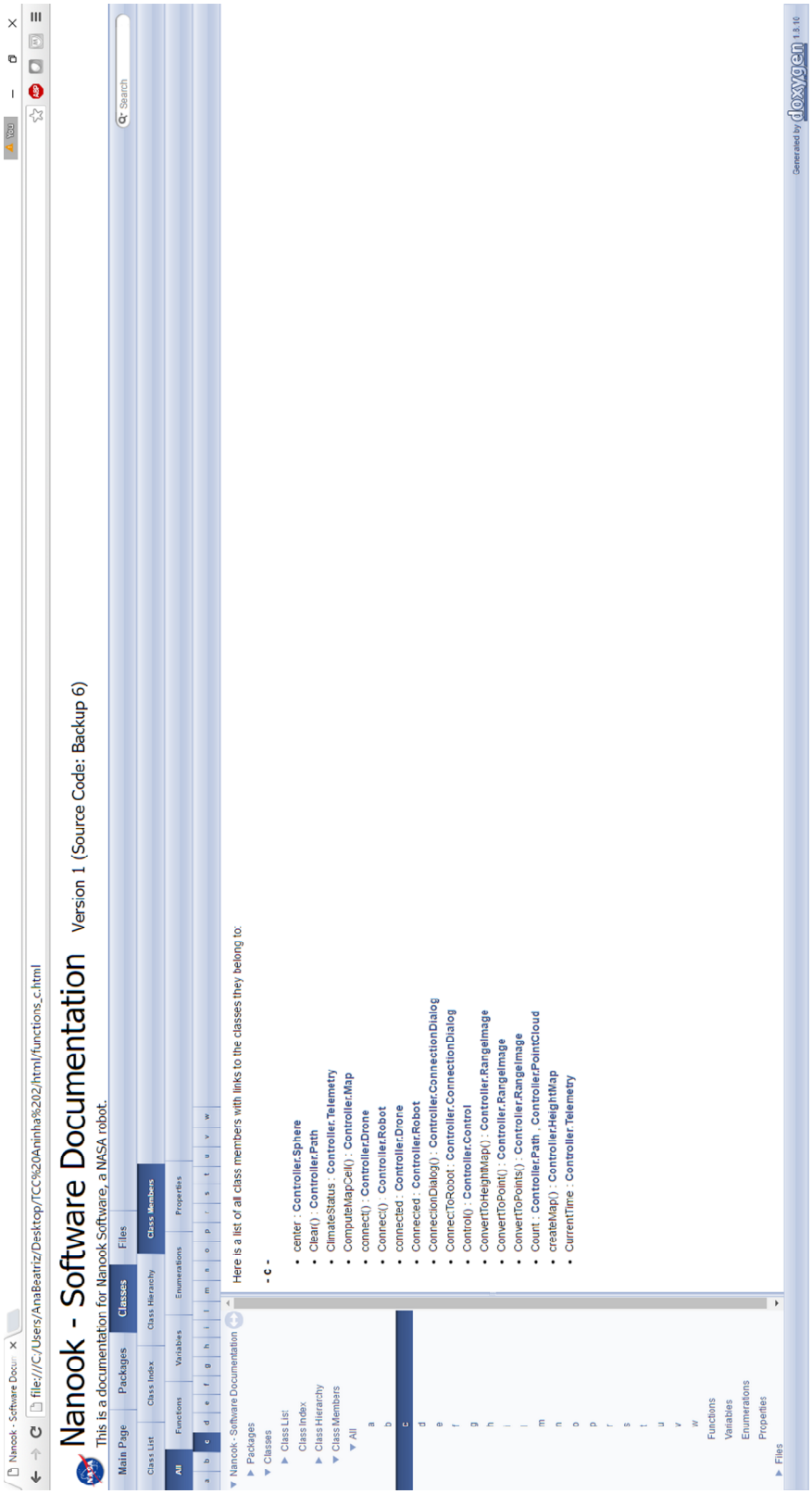
Fonte: Acervo do autor.

Figura 105 - Visualização das informações relativas a uma classe específica.



Fonte: Acervo do autor.

Figura 106 - Visualização dos membros das classes (em índice alfabético) com a indicação da classe a que pertencem.



Fonte: Acervo do autor.

6.2 Software Embarcado

O software embarcado foi escrito em C++, que também é uma linguagem orientada a objetos. O software é composto de arquivos *headers* (.hpp and .h), arquivos fontes (*source*) (.cpp), arquivos de projeto (.vcproj) e arquivos de solução (*solution*). O software embarcado que gerencia o sistema é o Windows XP Professional.

6.2.1 Estrutura do Software Embarcado

Como mencionado, o software embarcado foi escrito em C++, portanto, sua estruturação segue a de um projeto dessa linguagem.

6.2.1.1 Arquivos Fonte e Headers

Em C++, os arquivos *header* (.hpp and .h) são muito importantes para conectar os arquivos fonte (.cpp), que são aqueles que realmente são compilados. Primeiro, os arquivos fonte são compilados de modo independente, então, em um segundo passo, eles são conectados pelo compilador para criar um projeto. Quando um arquivo fonte precisa importar algo de outro arquivo fonte, o que ele precisa é importar o arquivo *header* equivalente.

Os arquivos *header* estabelecem a interface, enquanto os arquivos fonte contêm o código em si.

6.2.1.2 Arquivos de Projeto e de Solução

Em uma organização de alto nível, o software é composto de arquivos de projeto (.vcproj) e um arquivo de solução (.snl). O arquivo de projeto do Visual C++ contém a informação requerida para construir o projeto em si. (MICROSOFT, 2015d). O arquivo solução reúne um ou mais projetos que trabalham juntos para criar uma aplicação.

Imaginando como uma árvore, tem-se a seguinte estrutura (STACK OVERFLOW, 2011):

.sln

.vcproj

.hpp

.h

.cpp

6.2.1.3 Classes

Classes em C++ são definidas usando o termo *class* ou *struct*. Uma classe é composta de membros (declaração de dados e funções) e, opcionalmente, especificadores de acesso. Esses especificadores definem o direito de acesso para membros que seguem outros membros. Esses especificadores podem ser de três tipos: privado, público ou protegido. A Figura 107 mostra a estrutura da definição de uma classe.

Figura 107 - Definição de classe em C++.

```
class class_name {
  access_specifier_1:
    member1;
  access_specifier_2:
    member2;
  ...
} object_names;
```

Fonte: Wbuttutorials (n.d.).

6.2.1.4 Include e Include Guard

Para importar arquivos em um programa, a diretiva #include é usada. Especificamente em grandes projetos, como é o caso do software embarcado do Nanook, vários arquivos *header* podem ser incluídos mais de uma vez. Isso se torna um problema porque pode diminuir consideravelmente a velocidade do processo de compilação e mesmo gerar erros. Por essa razão, usar *Include Guard* é uma prática importante em C++. O Include Guard é uma técnica que usa um identificador único definido no topo do arquivo (usando #define). Por exemplo, supondo um arquivo x.hpp que é incluído em outros dois *headers*, j.hpp e k.hpp. Imaginando agora que outro arquivo, l.hpp, também inclua j.hpp e k.hpp. Isso poderia ser um

problema, porque internamente x.hpp estaria sendo importado duas vezes em l.hpp. Usa-se então o *include guard*, para evitar problemas desse tipo, conforme exemplificado ilustrativamente na Figura 108.

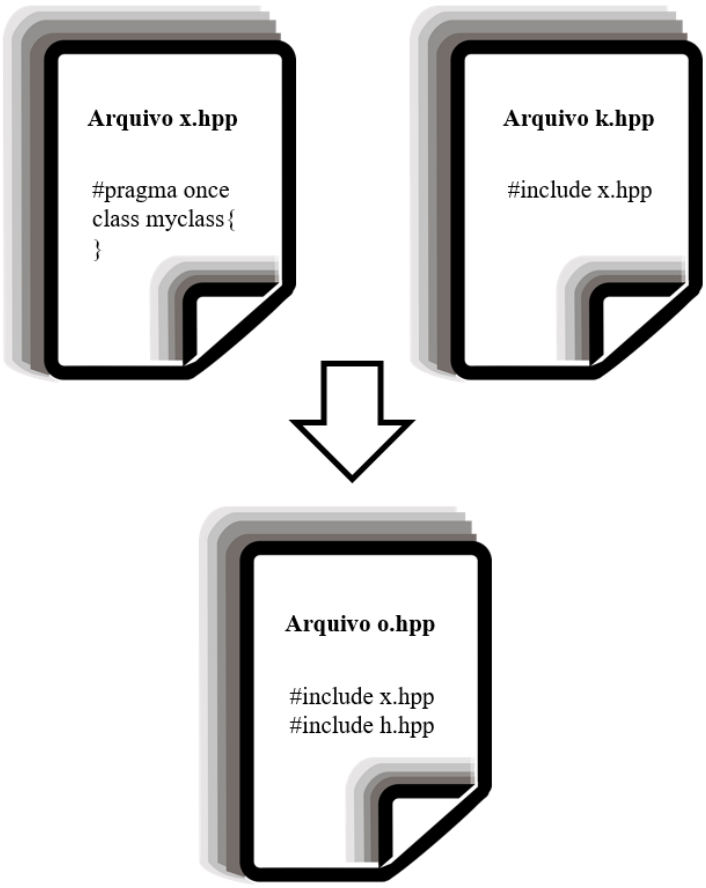
Figura 108 - Exemplo de aplicação da técnica do Include Guard.

```
1 //x.h
2
3 #ifndef __X_H_INCLUDED__   // if x.h hasn't been included yet...
4 #define __X_H_INCLUDED__   //   #define this so the compiler knows it has been included
5
6 class X { };
7
8 #endif
```

Fonte: CPLUSPLUS (2009).

Outra maneira para evitar essas situações seria usar "*#pragma once*", conforme ilustrado no exemplo abaixo:

Figura 109 – Utilização de "#pragma once" para evitar dupla importação.



Fonte: Elaborado pelo autor.

Dessa maneira, x.hpp só seria incluído uma vez em o.hpp.

6.2.1.5 Startup Project

Por padrão, *startup projects* (projetos de inicialização) são aqueles que rodam automaticamente quando o depurador do Visual Studio é iniciado (MICROSOFT, 2015b). Portanto, os projetos que pertenceram ao *Startup Project* serão aqueles que serão rodados na depuração.

6.2.2 Descrição do Software Embarcado do Nanook

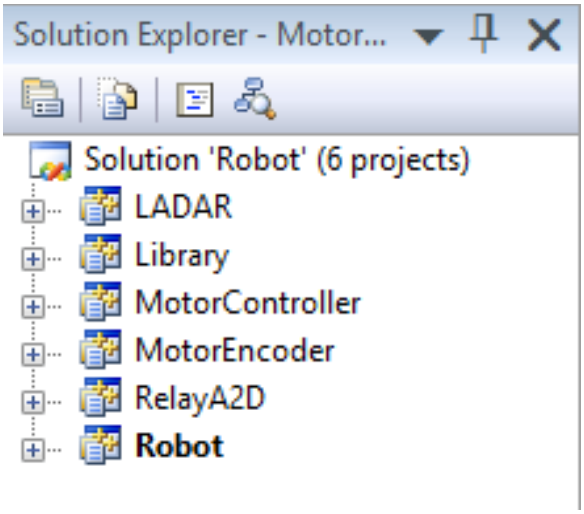
O arquivo de solução do Nanook foi denominado *Robot* e tem 6 arquivos de projeto associados a ele: LADAR, *Library* (Biblioteca), *MotorController* (Controlador do Motor), *MotorEncoder* (*Encoder* do Motor), *Robot* (Robô) e RelayA2D.

Os principais arquivos e suas funcionalidades são descritos abaixo.

6.2.2.1 Robot.vcproj

No arquivo de solução Robot.sln, o *startup project* (Robot.vcproj) pode ser identificado na janela *Solution Explorer* do Visual Studio, destacado em negrito, conforme Figura 110.

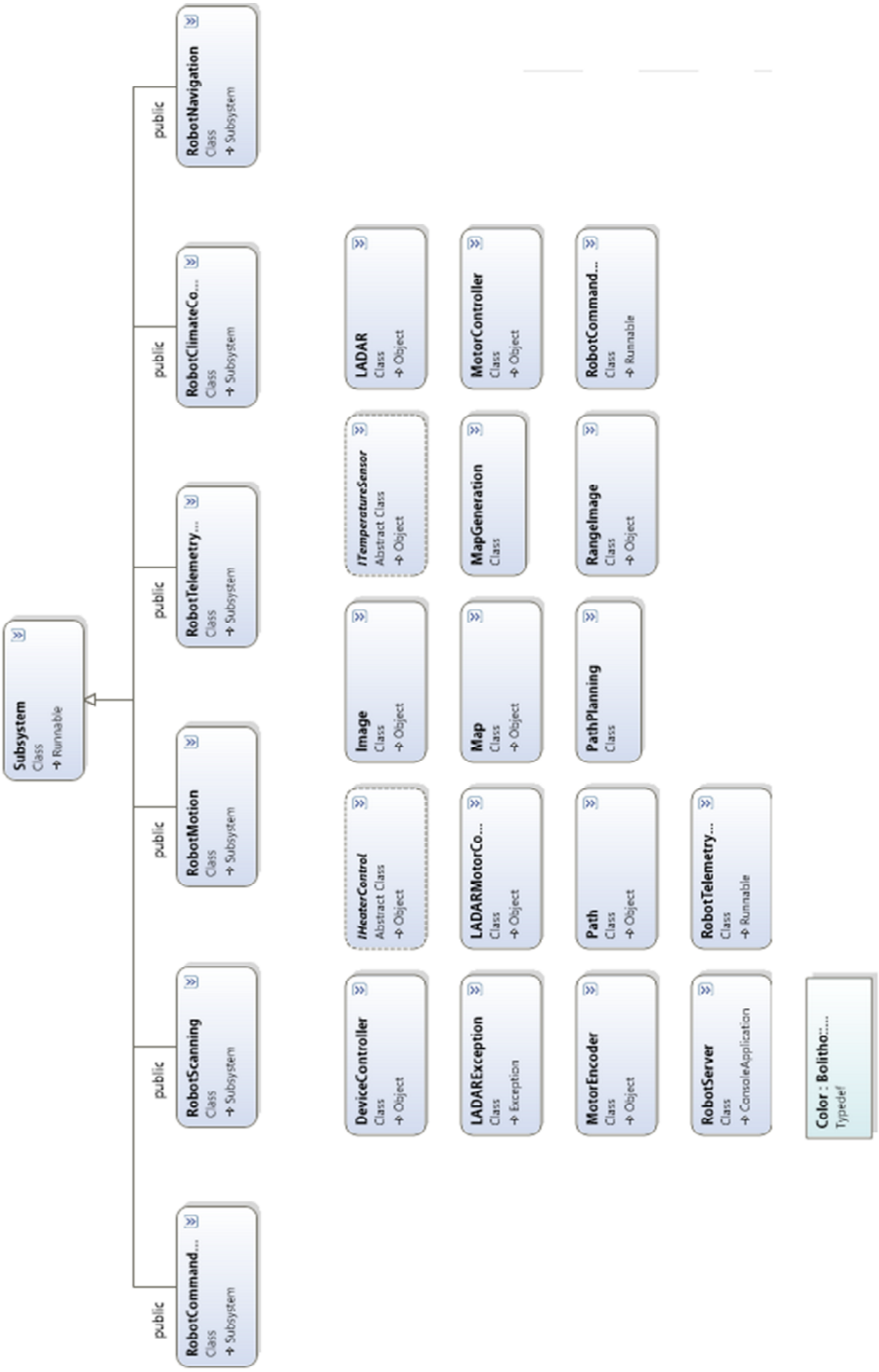
Figura 110 - Janela do Solution Explorer.



Fonte: Acervo do autor.

Esse é o arquivo de projeto mais importante do software embarcado e é composto pelos arquivos fonte principais (*source files*, com extensão .cpp) de outros arquivos de projeto e também por outros arquivos especificados para o *Startup Project*. É importante notar que ele não contém arquivos de teste de outros projetos. O diagrama de classes para esse projeto é dado pela Figura 111.

Figura 111 - Diagrama de classes.



Fonte: Acervo do autor.

Como pode ser observado no diagrama de classes, o sistema embarcado é composto de seis subsistemas principais. Os subsistemas *RobotCommand* e *RobotTelemetry* estão relacionados à conexão entre o robô e o cliente, ao recebimento de comandos do robô e também ao envio de dados. *RobotMotion* é responsável pelo movimento do robô, que pode ser dividido em três etapas: mover (*drive*), girar (*turn*) e escanear (*scan*).

O subsistema RobotMotion controla e gerencia a posição do robô, enquanto *RobotNavigation* controla e gerencia o sistema de navegação do robô. Por fim, o subsistema *TemperatureControl* seria responsável por controlar a temperatura de alguns alvos e subsistemas do robô. Entretanto, essa funcionalidade não foi completamente implementada, visto que não havia sensores de temperatura no hardware do robô.

6.2.2.2 LADAR.vcproj

Esse projeto é composto dos seguintes arquivos fonte: *LADARControl*, *LADARMotorControl*, *LADARTest*, *Map* e *RangeImage*. Os arquivos fonte desse projeto são responsáveis pelo controle e teste do motor de passo do LIDAR e da obtenção de imagem pelo LIDAR.

6.2.2.3 Library.vcproj

Esse projeto é responsável por fornecer importantes classes e métodos que suportam o software como um todo. Por exemplo, ferramentas para lidar com cores, data e tempo e operações matemáticas.

6.2.2.4 MotorController.vcproj

Nesse projeto, são escolhidos os motores a serem controlados (motores do lado direito ou motores do lado esquerdo). Para os motores escolhidos, pode-se inicializar seu movimento, pará-los e definir sua velocidade. Há também um arquivo de nome MotorControllerTest.cpp nesse projeto, que deve ter sido criado em uma fase de testes sem alterar o arquivo de projeto em funcionamento.

6.2.2.5 MotorEncoder.vcproj

Esse projeto é composto por dois arquivos fonte (MotorEncoder.cpp e MotorEnconderTest.cpp), além de três arquivos header (Config.hpp, MotorEncoder.hpp e Robot.hpp). Está relacionado ao recebimento de dados do *encoder*, seu teste e também a comparação de dados recebidos com os esperados. Ele é muito importante para o sistema de controle dos motores, uma vez que a CPU recebe informação referente ao deslocamento dos motores pelo *feedback* dos *encoders*.

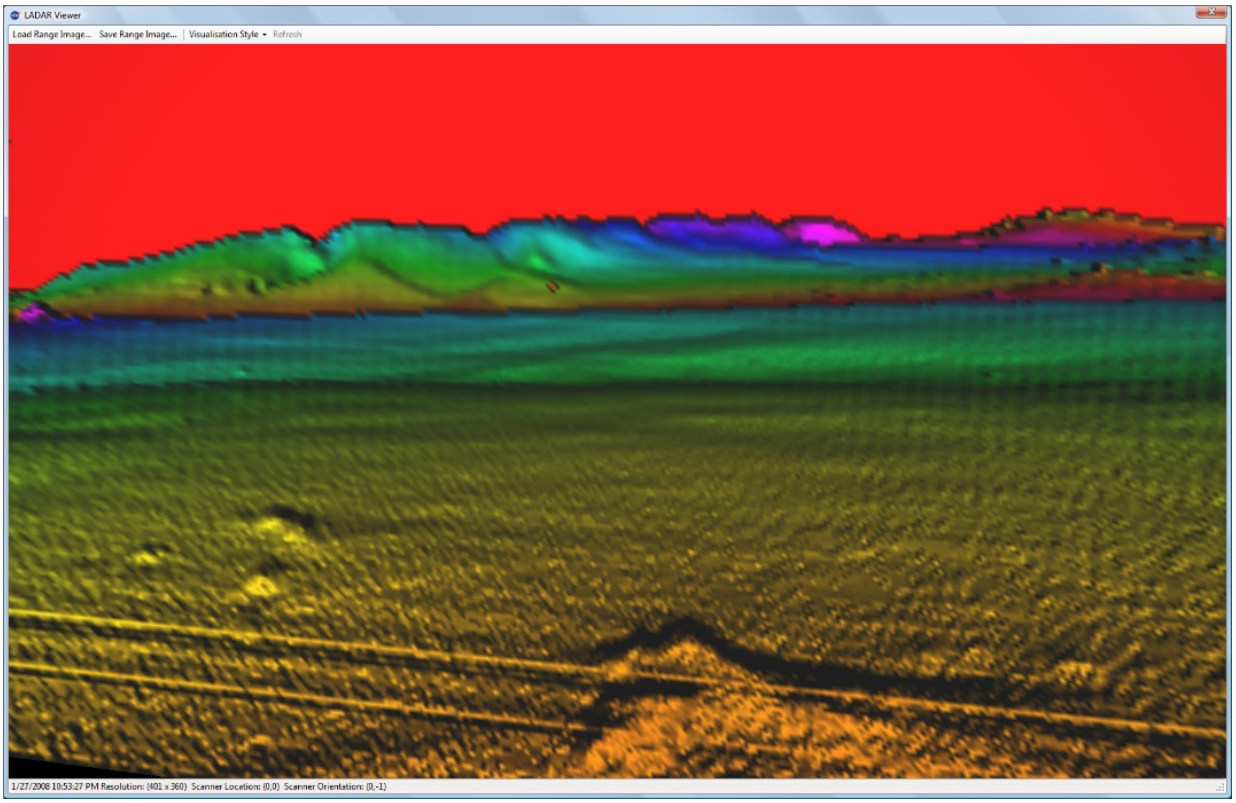
6.3 Funcionalidades do Software

A análise conjunta do programa de interface com o usuário e do *software* embarcado permitiram o entendimento das funcionalidades de alto nível do Nanook. Sua grande aplicabilidade é a obtenção de imagens do ambiente explorado, sendo capaz de comandar outros robôs para auxiliá-lo neste reconhecimento.

Uma rede de conexão sem fio entre os robôs do grupo permite que eles verifiquem quais robôs estão operacionais. Para operações simples do robô, em que o centro de comando está próximo, é possível usar essa mesma rede interna para controlá-lo. Em situações em que os operadores humanos estão em uma outra localidade, o Nanook, considerado o robô mãe, é capaz de usar como interface um satélite de comunicações em órbita e seus controladores na Terra.

O Nanook usa o escaneamento do LIDAR para desenvolver mapas do perfil 3D de um terreno com acurácia milimétrica, enviando os dados diretamente para o centro de controle. Um exemplo desse mapa pode ser visualizado na Figura 112, onde tons de amarelo e verde representam menores distâncias, enquanto tons de vermelho implicam em uma maior distância.

Figura 112 - Imagem do Polo Norte obtida pelo LIDAR.

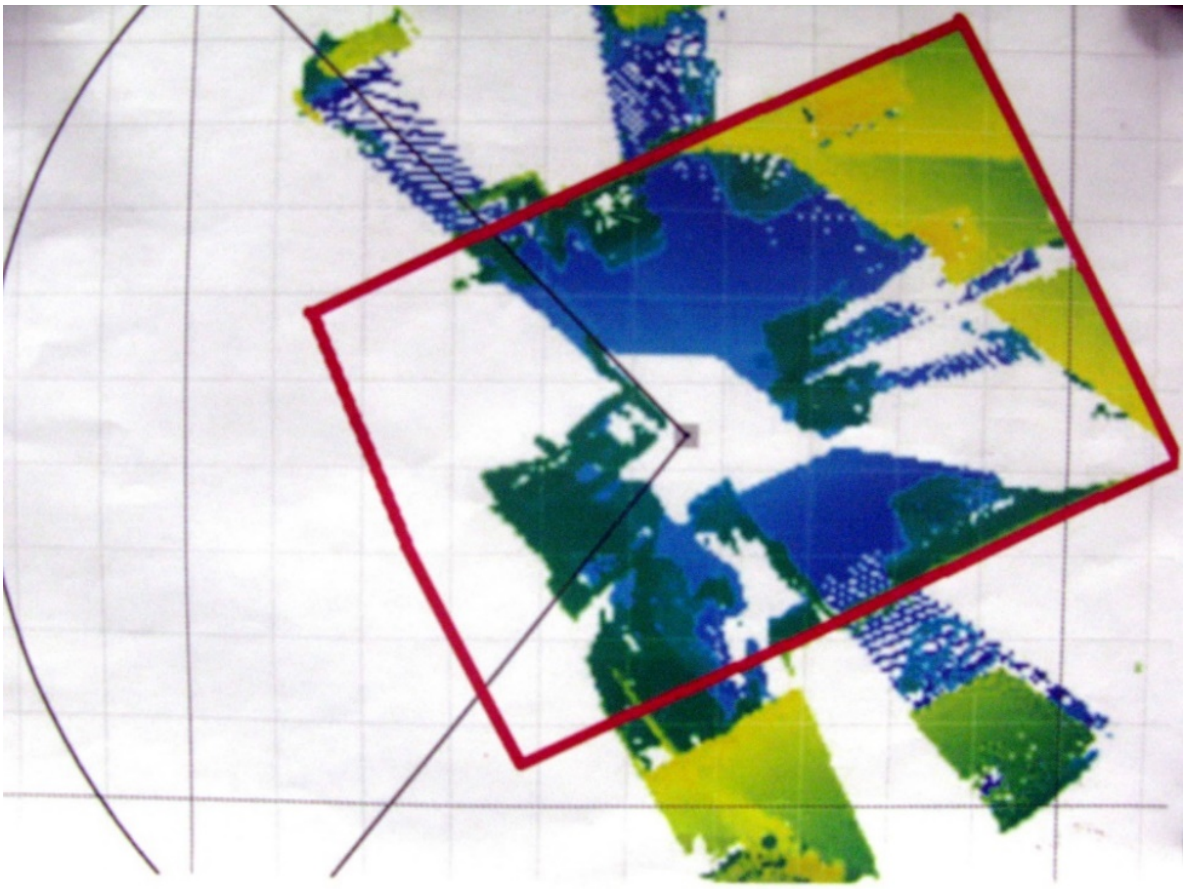


Fonte: Fornecido por Michael Comberiate.

O robô pode ainda rotacionar as imagens para uma visualização em mapa de alturas plano e colá-las sucessivamente, usando pontos de referência móveis conhecidos, obtendo um mapa conforme a Figura 113.

O Nanook transmite esses mapas compostos para operadores na Terra, quando requisitado, sempre que os *links* de comunicação estiverem disponíveis. Os operadores usam os mapas de perfil e de altura composto para desenvolver um plano de investigação, que será enviado para que o robô-mãe o execute.

Figura 113 - Mapa de altura composto.

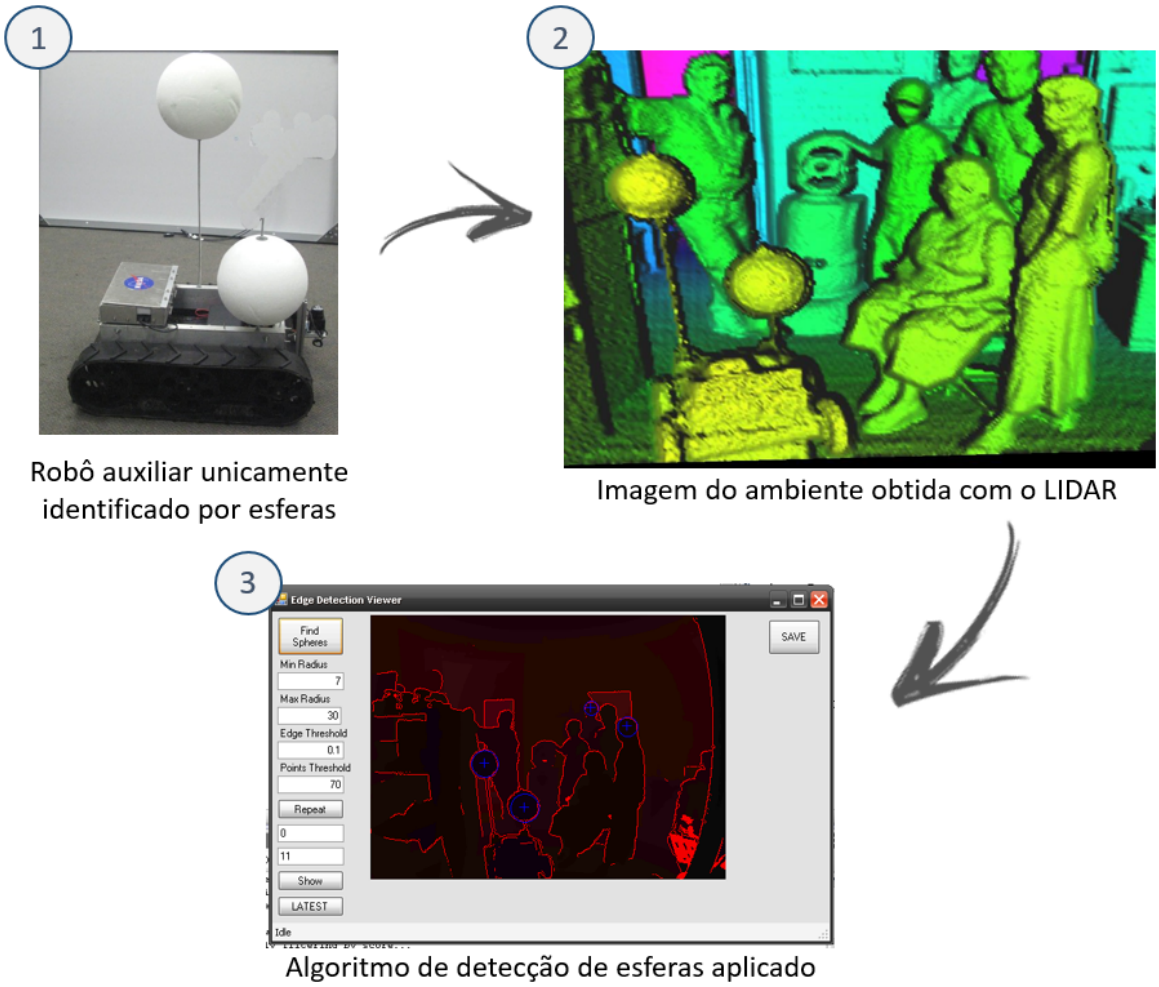


Fonte: Fornecido por Michael Comberiate.

Conforme comandado, o Nanook poderia agir ou enviar um robô auxiliar, que tenha as ferramentas adequadas, para uma localização específica. Uma aplicação poderia ser, por exemplo, enviar um robô auxiliar para retirar uma rocha (ferramentas adicionais, se disponíveis, poderiam ser usadas até mesmo para riscá-la e recolher suas amostras).

A Figura 114 apresenta uma ilustração desse processo de detecção de esferas para que o Nanook identifique os outros robôs.

Figura 114 – Detecção de esferas para identificação dos robôs auxiliares.

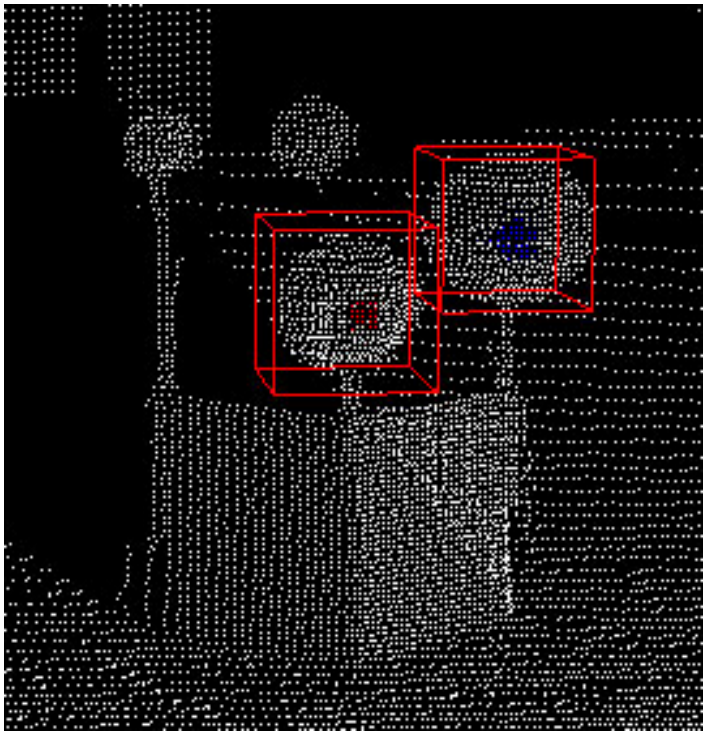


Fonte: Elaborado pelo autor a partir de imagens fornecidas por Michael Comberiate.

A identificação e diferenciação dos robôs auxiliares é possível pela utilização de uma técnica de reconhecimento de esferas, capaz de reconhecer os outros robôs pela disposição única de esferas a eles associada. As funcionalidades do software associadas ao reconhecimento de esferas são, portanto, necessárias para que o Nanook identifique marcas de referência esféricas, garantindo a identificação da localização e orientação dos robôs auxiliares. A tolerância para o tamanho das esferas identificáveis é configurada na própria janela de interface do programa.

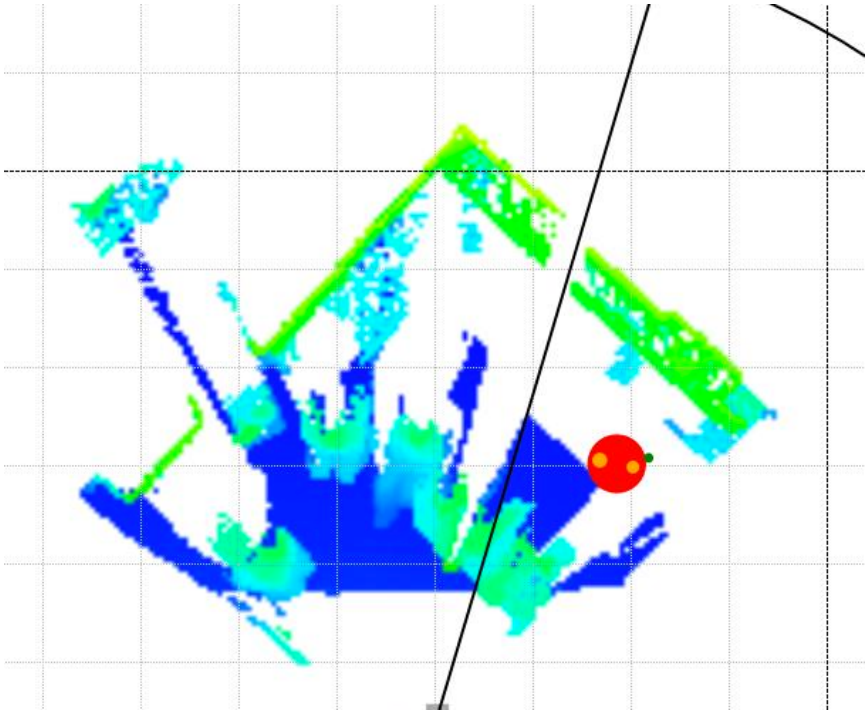
A detecção de esferas para identificação dos robôs auxiliares pode ainda ser aplicada às imagens em 2D ou em nuvem de pontos, conforme evidenciado na Figura 115 e na Figura 116.

Figura 115 - Detecção esférica em 3D pela nuvem de pontos.



Fonte: Fornecido por Michael Comberiate.

Figura 116 - Detecção de um robô auxiliar em 2D.



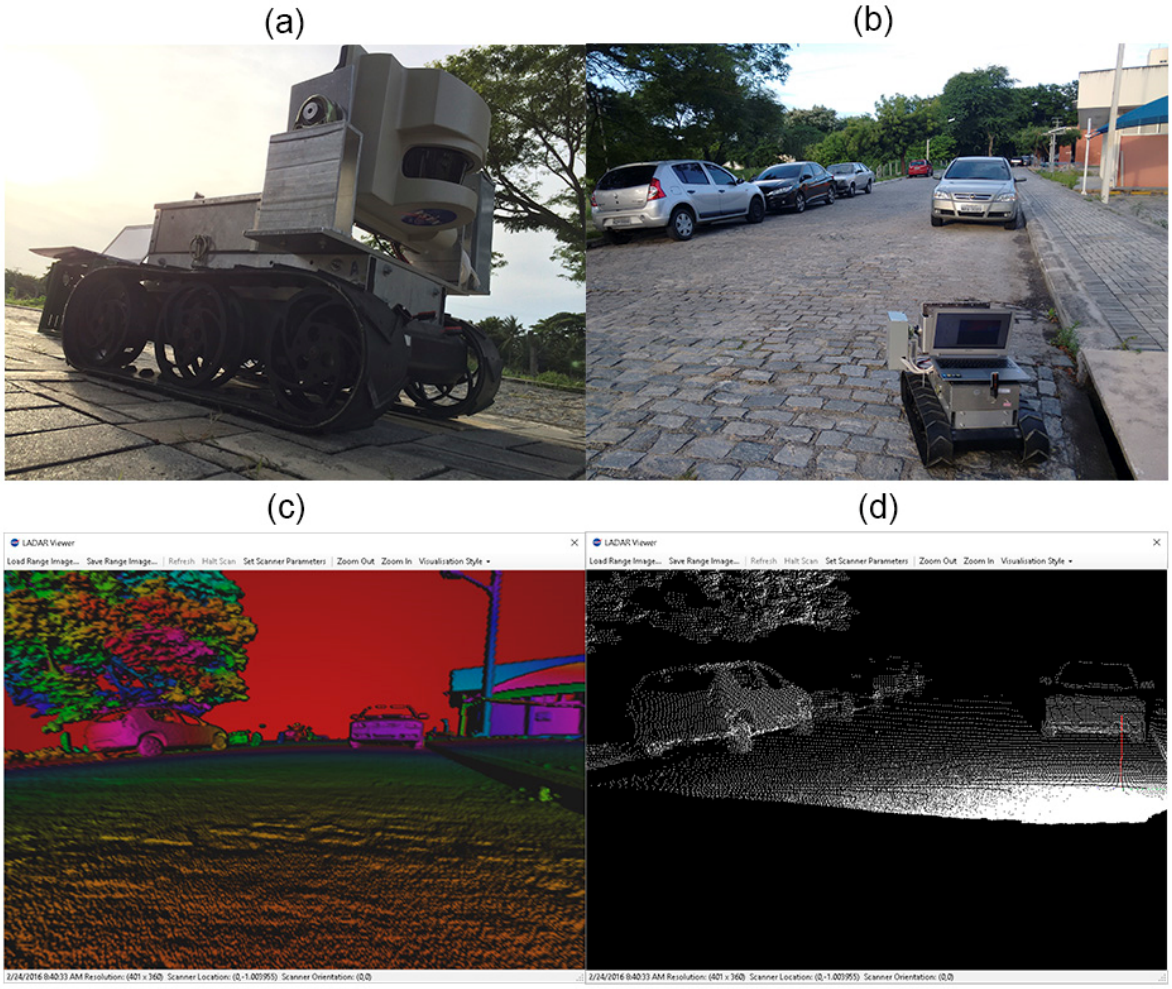
Fonte: Fornecido por Michael Comberiate.

O software do Nanook (embarcado e programa de interface com o usuário) permite a execução do reconhecimento de esferas, movimentos para frente e para trás, giros em um ângulo especificado, obtenção de imagens em um ângulo definido e paradas de emergência. Contudo, a ordem desses comandos não é uma decisão do Nanook. Ela é dada por um operador que esteja controlando o Nanook através do programa de interface com o usuário.

6.3.1 Teste de Mapeamento no Campus da UFC

Para corroborar o esclarecimento das funcionalidades do Nanook, foi realizado um teste de mapeamento no campus da UFC, ilustrado na Figura 117.

Figura 117 – Mapeamento da UFC: (a) Nanook, (b) Realização do mapeamento, (c) Mapa de distâncias e (d) Nuvem de pontos.



Fonte: Acervo do autor.

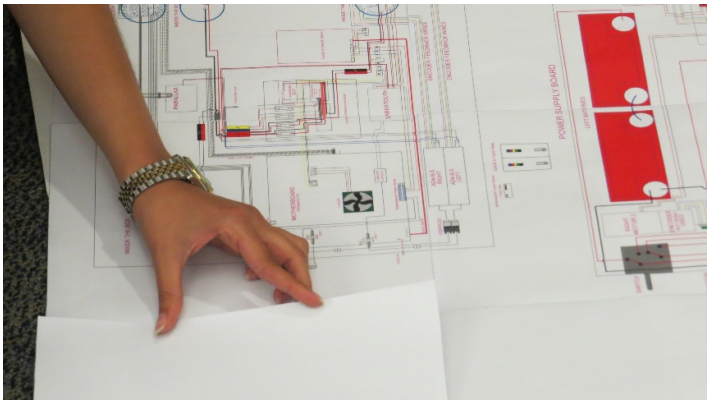
O robô, em destaque na Figura 117a, realizou o mapeamento do estacionamento em frente ao bloco de Engenharia Elétrica do Campus do Pici, da Universidade Federal do Ceará, conforme Figura 117b. O resultado mostrado na Figura 117c foi a obtenção de um mapa de distâncias em que é possível detectar visualmente o espaço livre à frente dos robôs, alguns carros com a distinção dos mais próximos e dos mais afastados e outros elementos do ambiente. Na Figura 117d, a mesma imagem é mostrada como uma nuvem de pontos.

7 DISCUSSÃO DOS RESULTADOS

O principal resultado da metodologia de engenharia reversa apresentada foi a elaboração de uma documentação para registrar as observações e conclusões obtidas. Foram desenvolvidas documentações para o hardware e para o software, conforme exposto, importantes para tornar acessível todo o conhecimento e entendimento adquirido durante a realização desse projeto. Assim, novas pessoas podem ser capazes de entender o Nanook e continuar o trabalho aqui desenvolvido. Além da facilidade e da não necessidade de despender um tempo considerável em algo que já foi realizado, os investimentos e esforços podem ser focados em gerar novos resultados, contribuindo para o avanço de novos projetos.

Além da documentação, pela inspeção completa do hardware, foi possível rastrear todo o sistema eletrônico do robô. Como resultado, foi desenvolvido um esquemático, conforme Figura 118, usando o AutoCad® para complementar a documentação do Nanook, de modo que pessoas interessadas consigam compreendê-lo sem a necessidade de uma investigação mais detalhada que requereria a desmontagem parcial do robô. Por ser não-invasiva, tem-se uma maneira mais segura para que, no futuro, as pessoas possam entender o sistema do robô e propor a implementação de mudanças no Nanook.

Figura 118 – Elaboração do esquemático resultante do projeto de engenharia reversa no Nanook.



Fonte: Acervo do autor.

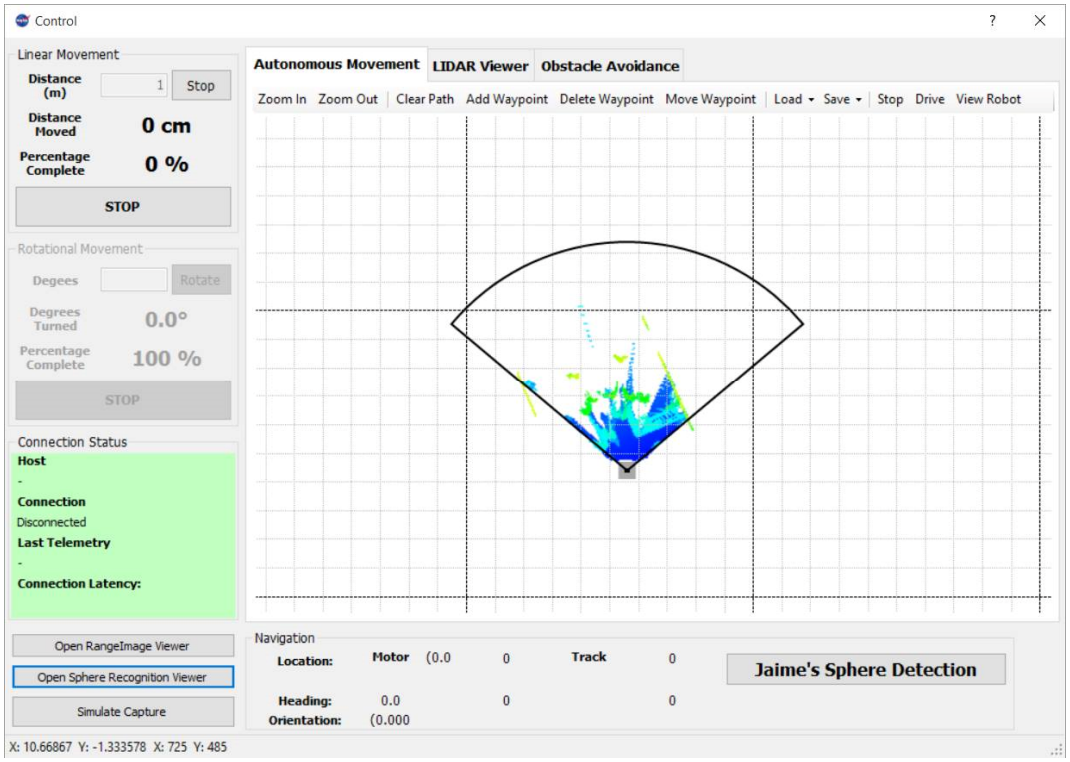
Adicionalmente, como resultado da fixação de etiquetas nas principais partes do robô, tem-se que, em uma situação que exija a investigação física do hardware, é possível identificá-lo mais facilmente e seguir as principais conexões de um modo coerente com a

documentação e os esquemáticos também resultantes desse projeto de engenharia reversa.

A documentação em HTML do software gerou uma interface amigável que relaciona as principais classes, métodos e funções de modo bastante acessível e simples.

O resultado da compreensão conjunta de hardware e software permitindo o entendimento das funcionalidades do Nanook foi a utilização do robô para mapeamento de ambientes. A Figura 119 ilustra uma visão *top-down* (vista superior do ambiente) de umas das salas do GPAR, obtida utilizando-se a interface de comando do robô. Nessa figura, o azul escuro representa alturas mais baixas. À medida que a altura correspondente aumenta, as cores indicadas no mapa mudam progressivamente para: verde, amarelo, laranja e, por fim, vermelho.

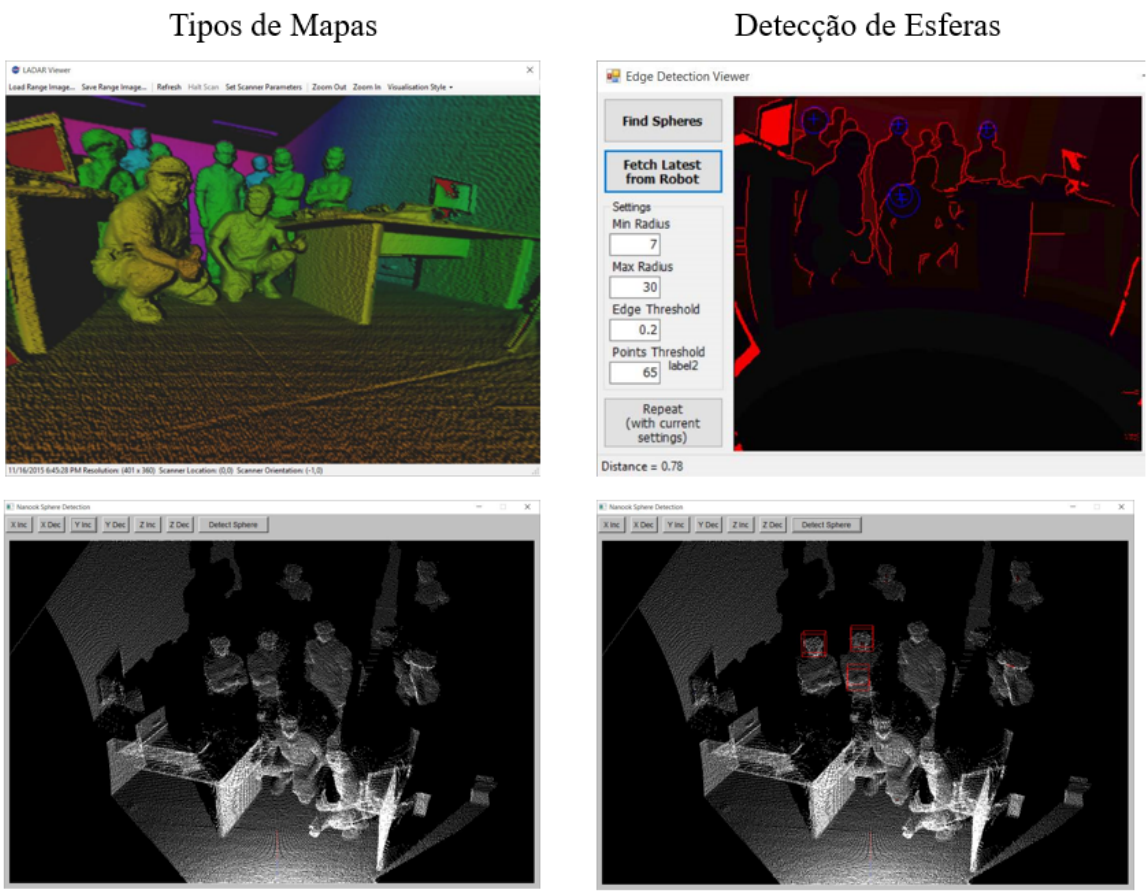
Figura 119 – Mapa com vista *top-down*.



Fonte: Acervo do autor.

A Figura 120 ilustra os tipos de visualização possíveis de serem obtidos com o Nanook (além da vista *top-down*): mapa de distância e nuvem de pontos, além de suas respectivas vistas com detecção de esferas.

Figura 120 – Estilos de visualização disponíveis.



Fonte: Acervo do autor.

7.1 Limitações do Nanook

Após o entendimentto das funcionalidades do Nanook e a possibilidade de sua completa operacionalização, percebeu-se que intrinsencamente existem algumas limitações que podem afetar a interpretação de seus resultados e trabalhos futuros com o robô.

Essas limitações são associadas à utilização do LIDAR para obtenção das imagens do ambiente. Como o LIDAR utiliza um sistema a laser para identificar o espaço à sua frente, se houver alguma superfície de coloração preta, ela será identificada como algo muito longe, independente da distância real. Isso acontece porque a cor preta absorve todos os componentes de luz, de modo que, como não há reflexão do laser nessa superfície, o sistema do robô entende erroneamente que o objeto está distante.

Outra restrição evidenciada refere-se à velocidade de escaneamento do LIDAR.

O escaneamento horizontal, que pode abranger uma visão de 180º e resolução variável de 0,25º, 0,5º e 1º, requer um tempo considerável para ser executado. Considerando que o escanemanto realiza-se por sucessivas linhas horizontais até abranger o alcance vertical pré-determinado, o tempo total de escaneamento deve ser avaliado quando se pretende desenvolver maior autonomia para o robô.

O LIDAR tem um espelho interno que gira promovendo a varredura de ondas da esquerda para a direita, medindo o tempo que é necessário para um pulso de luz infravermelha retornar após encontrar um objeto. O espelho gira quatro vezes a cada linha horizontal e então o motor de passo eleva a altura do LIDAR em 0,25º no eixo vertical. (STAKEM, 2015). O procedimento se repete a cada nova linha horizontal, construindo a imagem total após a varredura de todas as linhas. Desse modo, um escaneamento em tempo real, à medida que o robô se desloca, por exemplo, não é possível na máxima resolução de operação. Apesar de o tempo de escaneamento ser variável com a definição de alguns parâmetros de entrada (por exemplo, resolução, elevação vertical e amplitude do escaneamento), algumas especificações facilitam o dimensionamento desse tempo. O tempo de resposta mínimo do LIDAR (modelo SICK LMS221-30206 do Nanook) é de 13 ms, dependendo da resolução angular conforme Tabela 24, e a frequência máxima de escaneamento é de 75 Hz.

Tabela 24 – Tempo de resposta do LIDAR para diferentes valores da resolução angular

Resolução Angular	Tempo de Resposta Mínimo (ms)
0,25º	53,33
0,5º	26,66
1º	13,33

Fonte: SICK AG Waldkirch (2006)

O entendimento dessas limitações do Nanook não implicam necessariamente em restrições, mas evidenciam uma atenção a certos parâmetros que precisam ser considerados para garantir uma utilização adequada, ainda que diante desse cenário.

8 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho foi desenvolvido com o intuito de expor a engenharia reversa no ambiente acadêmico. As etapas de desenvolvimento deste trabalho passaram pela definição de engenharia reversa, com uma contextualização geral abordando sua história, a legislação atual e o entendimento da sua necessidade, pela proposição de uma metodologia aplicável a um robô móvel real e pela aplicação dessa metodologia em um estudo de caso. O caso prático aqui analisado foi o de um robô móvel que empregava um sistema de laser (LIDAR) para obter imagens do ambiente ao seu redor, o Nanook.

8.1 Conclusões

A engenharia reversa, quando licitamente aplicada, é uma ferramenta de promoção do conhecimento. Seu uso, inerente à natureza curiosa e dominadora do ser humano, foi largamente utilizado ao longo dos anos e tem ganhado repercussão recente devido as vantagens de ser aplicada em um cenário tão complexo como o atual. A visão de engenharia reversa aqui empregada, mais que a vantagem de tempo, de esforço e de dinheiro, pretendia fornecer uma ferramenta adequada para a difusão de conhecimentos no meio acadêmico.

Conforme a conceituação apresentada por Stefanelli et al (2010), “a engenharia reversa visa apropriar-se de conceitos estratégicos novos a partir da desconstrução de modelos ou soluções prontas”. A análise de um robô móvel real permitiu não só a familiarização com a engenharia reversa e suas metodologias, como também permitiu a apropriação de novos conceitos e sistemas aplicáveis na robótica.

O entendimento de que cada fim necessita de uma análise metodológica específica para o desenvolvimento da engenharia reversa permitiu diferenciar as peculiaridades do processo quando aplicado à robótica. A metodologia proposta, baseada em Ingle (1994) e Sharples (2010), foi satisfatoriamente empregada no Nanook, o robô analisado. Embora de simples estruturação, sua aplicação permitiu que os objetivos a que este trabalha almejava fossem alcançados com êxito. A análise de dados, o que incluiu o reconhecimento do projeto e a análise do robô em si, forneceram informações completas para o entendimento do Nanook. A geração de informação utilizou esse conhecimento adquirido para formalizar as conclusões e gerar uma documentação acessível ao público em geral para posteriores consultas e aprimoramentos. Apesar das particularidades de cada sistema, esse processo permitiu o

reconhecimento do sistema completo (hardware e software), que, embora apresentados separadamente, podem ser entendidos conjuntamente.

A engenharia reversa do hardware do robô identificou componentes do sistema eletrônico que não mais estavam sendo utilizados, como o *Parallax*. Também foi capaz de especificar componentes obsoletos (conversores CC/CC de 24 V para 5 V e 12 V) e reconhecer as interconexões do sistema eletrônico, garantindo sua reprodutibilidade. A organização do hardware efetuada pela etiquetação dos principais cabos garantiu um entendimento acessível por inspeção visual não-invasiva rápida (sem a necessidade de desmontagem do robô).

A engenharia reversa do software do robô, dividida no programa de interface com o usuário e no *software* embarcado, permitiu o entendimento da estruturação do código e de suas funcionalidades.

Quando considerados conjuntamente, hardware e software, não só são capazes de explicar como o Nanook pode explorar ambientes, obter imagens destes (como foi feito no teste de mapeamento no campus da UFC) e reconhecer seus robôs auxiliares, como também são suficientes para que uma cópia do robô seja desenvolvida.

A finalização deste trabalho com a exibição da documentação desenvolvida é, portanto, mais que uma divulgação dos resultados obtidos a partir da aplicação da metodologia de engenharia reversa proposta. Trata-se da disseminação de uma base de conhecimentos que permite a cópia do robô, seu aprimoramento e mesmo o emprego em outras aplicações cabíveis.

8.2 Sugestões para Trabalhos Futuros

Como continuação desse trabalho, sugere-se o prosseguimento da metodologia apresentada de modo a resultar em um processo de reengenharia. O Nanook apresenta oportunidades de aprimoramentos além dos aqui efetuados, podendo-se destacar a atualização de seu hardware. Em relação ao software, existe a potencial implementação de funcionalidades como desvio de obstáculo, que implicariam em uma maior autonomia do robô.

Ser capaz de replicar um robô tão complexo e aperfeiçoá-lo é uma excelente fonte de saber. Esse processo, além de apresentar oportunidades de inovação tecnológica, pode instigar novos projetos no Departamento de Engenharia Elétrica da UFC. Com mais de um robô similar ao Nanook, pode-se explorar suas funcionalidades associadas à sua utilização como robô mestre. A integração com robôs auxiliares permite aumentar a complexidade do mapeamento de ambientes, fornecendo informações mais precisas e mais eficazes.

REFERÊNCIAS

AMPHENOL Corporation. **Amphenol® MIL-DTL-5015 and MIL-5015 Type Standard Cylindrical Connectors**. Disponível em <http://www.farnell.com/datasheets/64131.pdf> Acesso em: 23 jul. 2015

BALLADE SPORTS. Disponível em <**https://www.balladesports.com/products/hondas2000/oddyssey-extreme-series-drycell-pc680-battery**> Acesso em: 26 ago. 2016

BRAGA, Rosana T. Vaccare. **Engenharia Reversa e Reengenharia**. Engenharia de Software – SCE 186. Disponível em <http://www.cin.ufpe.br/~pjs/IC/Aula19_ReengenhariaEngReversa.pdf> Acesso em: 20 jun. 2016

BRAGA, Rosana. T. V. **Engenharia Reversa e Reengenharia**. Notas de Aula, Engenharia de Software, Universidade Federal do Paraná, Curitiba, n.d. Disponível em: <www.inf.ufpr.br/silvia/ES/reengenharia/reengenharia.pdf> Acesso em: 09 ago. 2016.

BRASIL. Lei nº 12.853 , de 14 de agosto de 2013. Altera os arts. 5º, 68, 97, 98, 99 e 100, acrescenta arts. 98-A, 98-B, 98-C, 99-A, 99-B, 100-A, 100-B e 109-A e revoga o art. 94 da Lei nº 9.610, de 19 de fevereiro de 1998, para dispor sobre a gestão coletiva de direitos autorais, e dá outras providências. Disponível em <http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2013/Lei/L12853.htm> Acesso em: 10 ago. 2016

BRASIL. Lei nº 9.609 , de 19 de fevereiro de 1998. Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências. 1998a. Disponível em <http://www.planalto.gov.br/ccivil_03/leis/L9609.htm> Acesso em: 10 ago. 2016

BRASIL. Lei nº 9.610 , de 19 de fevereiro de 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências. 1998b. Disponível em <http://www.planalto.gov.br/ccivil_03/LEIS/L9610.htm> Acesso em: 10 ago. 2016

CANHOTA JUNIOR, Antonio Jorge Sapage da; SOUZA, Diego Alves de; MOUTINHO, Diogo dos Santos; LOHNEFINK, Felipe Paixão. **Engenharia Reversa**. Rio de Janeiro: Universidade Federal Fluminense, 2005. Disponível em <http://www2.ic.uff.br/~otton/graduacao/informaticaI/apresentacoes/eng_reversa.pdf> Acesso em: 20 jun. 2016

CPLUSPLUS. **Headers and Includes: Why and How**. 2009. Disponível em <http://www.cplusplus.com/forum/articles/10627> Acesso em: 23 jul. 2016

DALRYMPLE, Odesma. Engineering Education PhD Program. **Researching Reverse Engineering.** 2009. Disponível em <https://engineering.purdue.edu/ENE/HomepageFeatures/Spotlights/Researchingreverseengineering2> Acesso em: 10 ago. 2016

DIGSHIP. **AD4-B: Quadrature to RS232 Adapter**. DigShip.com. Disponível em <http://www.digchip.com/datasheets/parts/datasheet/502/AD4-B-pdf.php> Acesso em: 23 jul. 2015

DIMENSION ENGINEERING. **Sabertooth 2x25 User's Guide**. 2007. Disponível em <https://www.dimensionengineering.com/datasheets/Sabertooth2x25.pdf> Acesso em: 28 ago. 2016

DONETPERLS. **Class**. n.d. Disponível em <http://www.dotnetperls.com/class> Acesso em: 23 jul. 2015

DONETPERLS. **Partial**. n.d. Disponível em <http://www.dotnetperls.com/partial> Acesso em: 23 jul. 2015

DONETPERLS. **Struct**. n.d. Disponível em <http://www.dotnetperls.com/struct> Acesso em: 23 jul. 2015

DRIZIN, Ricardo; BUOANANNI, Ulisses. **A cultura Cracker e a Engenharia Reversa**. Disponível em <http://drizin.io/A-cultura-Cracker-e-a-Engenharia-Reversa/> Acesso em: 20 jun. 2016

DYNAPAR. **Encoder - Glossary of Terms**. Disponível em <https://www.dynapar.com/uploadedFiles/Downloads/Encoder_Glossary_of_Terminology.pdf> Acesso: 26 ago. 2016

EILAM, Eldad. **Reversing: Secrets of Reverse Engineering**. Wiley Publishing, Indiana, Estados Unidos, 2005.

ESTADOS UNIDOS. Lei Pública nº 105–304 – de 28 de outubro de 1998. Digital Millennium Copyright Act. To amend title 17, United States Code, to implement the World Intellectual Property Organization Copyright Treaty and Performances and Phonograms Treaty, and for other purposes.

FARIAS, Guilherme. **Engenharia Reversa**. 2009. Disponível em <http://www.guiky.com.br/2009/09/engenharia-reversa.html> Acesso em: 12 ago. 2016.

FURTADO, Lucia B.; ASSAD, Marta M. N. **Engenharia Reversa como Ferramenta de Melhoria em Processos de Montagem de Novos Produtos**. XXXII Encontro Nacional de Engenharia de Produção, Bento Gonçalves, RS, 2012. Disponível em <http://www.abepro.org.br/biblioteca/enegep2012_TN_STP_157_919_20049.pdf> Acesso em: 6 ago. 2016

GENGHINI, Paulo Rogério. **O processo de engenharia reversa, com a utilização de sistemas de digitalização óptico sem contato de superfície e sua aplicabilidade na indústria automotiva**. 2013. 39 f. Monografia (Graduação em Engenharia Automotiva) – Centro Universitário do Instituto Mauá de Tecnologia, São Caetano do Sul, 2013. Disponível

em <http://maua.br/files/monografias/completo-processo-engenharia-reversa-280725.pdf> Acesso em: 18 jun. 2016

GHODKE, Sumeet; AWATI, Mahesh. **Rapid Product Development**. Disponível em <https://pt.scribd.com/doc/117472329/Reverse-Engineering> Acesso em: 8 ago. 2016.

HAFEMANN, Lodemar José. **Protótipo de Gerador de Código Fonte baseado em Diagramas de Seqüências**. 2000. 64 f. Monografia (Bacharel em Ciências da Computação) – Universidade Regional de Blumenau, Blumenau, 2000.

IACRB. Information Assurance Certification Review Board. **Certified Reverse Engineering Analyst (CREA).** Disponível em <http://www.iacertification.org/crea_certified_reverse_engineering_analyst.html> Acesso em: 10 ago. 2016

INGLE, Kathryn A. **Reverse Engineering**. McGraw-Hill, Inc. 1994

INTEL. **INTEL DESKTOP BOARD D945GCLF2**. Technical Product Specification. 2008. Disponível em <http://downloadmirror.intel.com/16960/eng/D945GCLF2_TechProdSpec02.pdf> Acesso em: 17 jun. 2015.

JAPÃO. Ato nº 54, de 10 de julho de 2015. Emenda do Ato nº 47, de 19 de maio de 1993, de Prevenção da Concorrência Desleal. Disponível em <http://www.wipo.int/edocs/lexdocs/laws/en/jp/jp204en.pdf> Acesso em: 23 ago. 2016

LANDTEC. **Engenharia**. Disponível em <http://www.landtecengenharia.com.br/engenharia/> Acesso em: 8 ago. 2016

LAZENBY, John Francis. **The First Punic War: A Military History**. 1996. Stanford, California. Stanford University Press.

MACORATTI, José Carlos. **UML – Conceitos Básicos II**. Disponível em <http://www.macoratti.net/vb_uml2.htm> Acesso: 28 ago. 2016

MAYO, WAYLAND. **Russian B-29 Clone — The TU-4 Story**. Disponível em <http://www.rb-29.net/html/03relatedstories/03.03shortstories/03.03.10contss.htm> Acesso: 16 ago. 2016

MERRIAM-WEBSTER. **Reverse Engineer.** n.d. Disponível em: <http://www.merriam-webster.com/dictionary/reverse%20engineer> Acesso em: 17 jun. 2016

MICROSOFT. **Classes (C# Programming Guide)**. Visual Studio, 2015a. Disponível em <https://msdn.microsoft.com/en-us/library/x9afc042.aspx> Acesso em: 23 jul. 2015

MICROSOFT. **Enum (C# Reference)**. Visual Studio, 2015b. Disponível em <https://msdn.microsoft.com/en-us/library/sbbt4032.aspx> Acesso em: 23 jul. 2015

MICROSOFT. **Partial Classes and Methods (C# Programming Guide)**. Visual Studio, 2015c. Disponível em <https://msdn.microsoft.com/en-us/library/wa80x488.aspx> Acesso em: 23 jul. 2015

MICROSOFT. **Project Files**. Visual Studio, 2015d. Disponível em <https://msdn.microsoft.com/en-us/library/2208a1f2.aspx> Acesso em: 23 jul. 2015

MICROSOFT. **Solution Explorer**. Visual Studio, 2008a. Disponível em <https://msdn.microsoft.com/en-us/library/26k97dbc(v=vs.90).aspx> Acesso em: 23 jul. 2015

MICROSOFT. **Startup Project, Common Properties, Solution Property Pages Dialog Box**. Visual Studio, 2008b. Disponível em <https://msdn.microsoft.com/en-us/library/09138bex(v=vs.90).aspx> Acesso em: 17 ago. 2016

MICROSOFT. **Static Classes and Static Class Members (C# Programming Guide)**. Visual Studio, 2015e. Disponível em <https://msdn.microsoft.com/en-us/library/79b3xss3.aspx> Acesso em: 23 jul. 2015

MORTENSEN, Peter; RASMUSSEN, Mark S. **When to use static classes in C#**. Stack Overflow. 2008. Disponível em <http://stackoverflow.com/questions/241339/when-to-use-static-classes-in-c-sharp> Acesso em: 23 jul. 2015

MURY, Luiz Gilberto Monclaro. **Uma Metodologia para Adaptação e Melhoria de Produtos a partir da Engenharia Reversa**. 2000. 100f. Dissertação (Mestrado em Engenharia de Produção). Universidade Federal do Rio Grande do Sul, Porto Alegre.

PATEL, Nikunj. **Using Reverse Engineering Techniques for Product Innovation: The Legitimate Way**. 2014. Disponível em <http://www.roboticstomorrow.com/article/2014/12/using-reverse-engineering-techniques-for-product-innovation-the-legitimate-way/5217> Acesso em: 18 jun. 2016

PAULINO, Jorge. **A Engenharia Reversa**. 2009. Disponível em <http://www.artigonal.com/ensino-superior-artigos/a-engenharia-reversa-1308116.html> Acesso em: 19 jun. 2016

PAVIM, Alberto Xavier. **Documentação de Código-fonte com a Ferramenta Doxygen**. Laboratório de Metrologia e Automatização – LABMETRO/EMC. Universidade Federal de Santa Catarina. Florianópolis, 2006.

PIEKARSKI, Ana Elisa Tozetto; QUINÁLIA, Marcos Antonio. **Reengenharia de software: o que, por quê e como**. Departamento de Informática – UNICENTRO. Guarapuava, PR, 2000.

PITTMAN EXPRESS. **GM9236S027**. Disponível em <https://www.servo2go.com/support/downloads/GM9236S027.pdf> Acesso: 28 ago. 2016

PITTMAN. **DC Gearmotor GM9236S027-R1-SP**. Disponível em <http://www.servocomponents.com/_literature_115324/GM9236S027-R1_PDF> Acesso: 28 ago. 2016

PORTELLA, Monica de Carvalho. **Engenharia Reversa e a Importância para as Empresas**. 2015. Disponível em <http://www.webartigos.com/artigos/engenharia-reversa-e-a-importancias-para-as-empresas/131721/#ixzz4C9uiPDRS> Acesso em: 20 jun. 2016

POWER SONIC. **PSH-12180FR 12 Volt 21.0 AH.** Rechargeable Sealed Lead Acid Battery. Disponível em <http://goo.gl/6cYcxn> Acesso: 26 ago. 2016

POZZEBON, Rafaela. **O que GitHub?** Tableless. 2015. Disponível em <http://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/> Acesso em: 24 ago. 2016

REIS, Thiago. **Gestão de Projeto – O que é e para que serve?** 2014. Disponível em <http://www.projectbuilder.com.br/blog-pb/entry/conhecimentos/o-que-e-gestao-de-projetos-e-para-que-serve> Acesso em: 1 ago. 2016

RODRIGUES, Rivaldo. **Doxygen**. Prática de Algoritmos e Estrutura de Dados I, Universidade Federal do Rio Grande do Norte. Rio Grande do Norte, n.d. Disponível em <https://moodle.ufsc.br/pluginfile.php/1393663/mod_resource/content/0/DoxygenTutorial.pdf> Acesso em: 24 ago. 2016

ROUSE, Margaret; CRAWFORD, Arleigh. **Reverse Engineering**. Search Software Quality. 2007. Disponível em <http://searchsoftwarequality.techtarget.com/definition/reverse-engineering> Acesso em: 20 jul. 2015

SAMUELSON, Pamela; SCOTCHMER, Suzanne. **The Law & Economics of Reverse Engineering**. University of California, Berkeley, 2002. Disponível em <http://people.ischool.berkeley.edu/~pam/papers/l&e%20reveng3.pdf> Acesso em: 9 ago.

2016.

SCHWARTZ, Mathew. **How to Reverse-Engineering**. 2001. Disponível em <http://www.computerworld.com/article/2585652/app-development/reverse-engineering.html> Acesso: 17 jun. 2016.

SHARPLES, Rachel E. **The Efficiency of Reverse Engineering on the Design of the ORCA XI Autonomous Underwater Vehicle**. 2010. 43 f. Dissertação (Bacharelado em Engenharia Mecânica) – Massachusetts Institute of Technology, USA.

SICK AG Waldkirch. **LMS200/211/221/291 Laser Measurement Systems – Technical Description**. Alemanha, 2006. Disponível em <http://sicktoolbox.sourceforge.net/docs/sick-lms-technical-description.pdf> Acesso em: 23 jul. 2015

SILICON LABS. **CP2102**. Disponível em <https://cdn.sparkfun.com/datasheets/BreakoutBoards/CP2102_v1.2.pdf> Acesso em: 23 jul. 2015

SILVA, João Manoel Gomes da. **Controle em Malha Fechada**. 2000. Disponível em <http://www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/node6.html> Acesso em: 17 ago. 2016.

SRINIVAS, Pooja. **Russian B29 Clone – The Tu-4 History**. World News, Reverse Engineering Tupolev TU-4, 2010. Disponível em <https://wn.com/reverse_engineering_tupolev_tu-4> Acesso em: 20 ago. 2016

STACK OVERFLOW. **What are .sln and .vcproj files, and what do they contain?** 2011. Disponível em <http://stackoverflow.com/questions/7133796/what-are-sln-and-vcproj-files-and-what-do-they-contain> Acesso em: 23 jul. 2015

STACK OVERFLOW. **What is the difference between reverse and forward engineering in SQL**. 2015. Disponível em <http://stackoverflow.com/questions/32318183/what-is-the-difference-between-reverse-and-forward-engineering-in-sql> Acesso em: 20 jun. 2016

STAKEM, Patrick H. **Robots in the Classroom, Research and Space**. 2015. Disponível em <http://www.theoldrobots.com/Pat_Sc.html> Acesso em: 26 jun. 2016

STEFANELLI, Eduardo José; INFANTOZZI, Ricardo; VARGAS, Miltom Mansilla; POLITANO, Rodolfo. **Engenharia Reversa: Discussão sobre validade e legalidade desta prática**. Instituto Federal de Educação, Ciência e Tecnologia São Paulo. São Paulo, 2010.

TECHOPEDIA. **Forward Engineering**. Disponível em <https://www.techopedia.com/definition/19445/forward-engineering> Acesso em: 20 jun. 2016

TELEA, Alexandru C. **Reverse Engineering – Recent Advances and Applications**. Croácia, 2012, p. IX. ISBN 978-953-51-0158-1

UNIÃO EUROPEIA. Diretiva nº 2009/24/EC do Parlamento Europeu, de 23 de abril de 2009, relativa à proteção jurídica dos programas de computador. Disponível em<http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32009L0024&qid=1471356914905&from=en> Acesso em: 16 ago. 2016

US DIGITAL. **AD4B: Quadrature to RS-232 Adapter**. Disponível em <http://cdn.usdigital.com/assets/datasheets/AD4B_datasheet.pdf?k=634869575565388904> Acesso: 28 ago. 2016

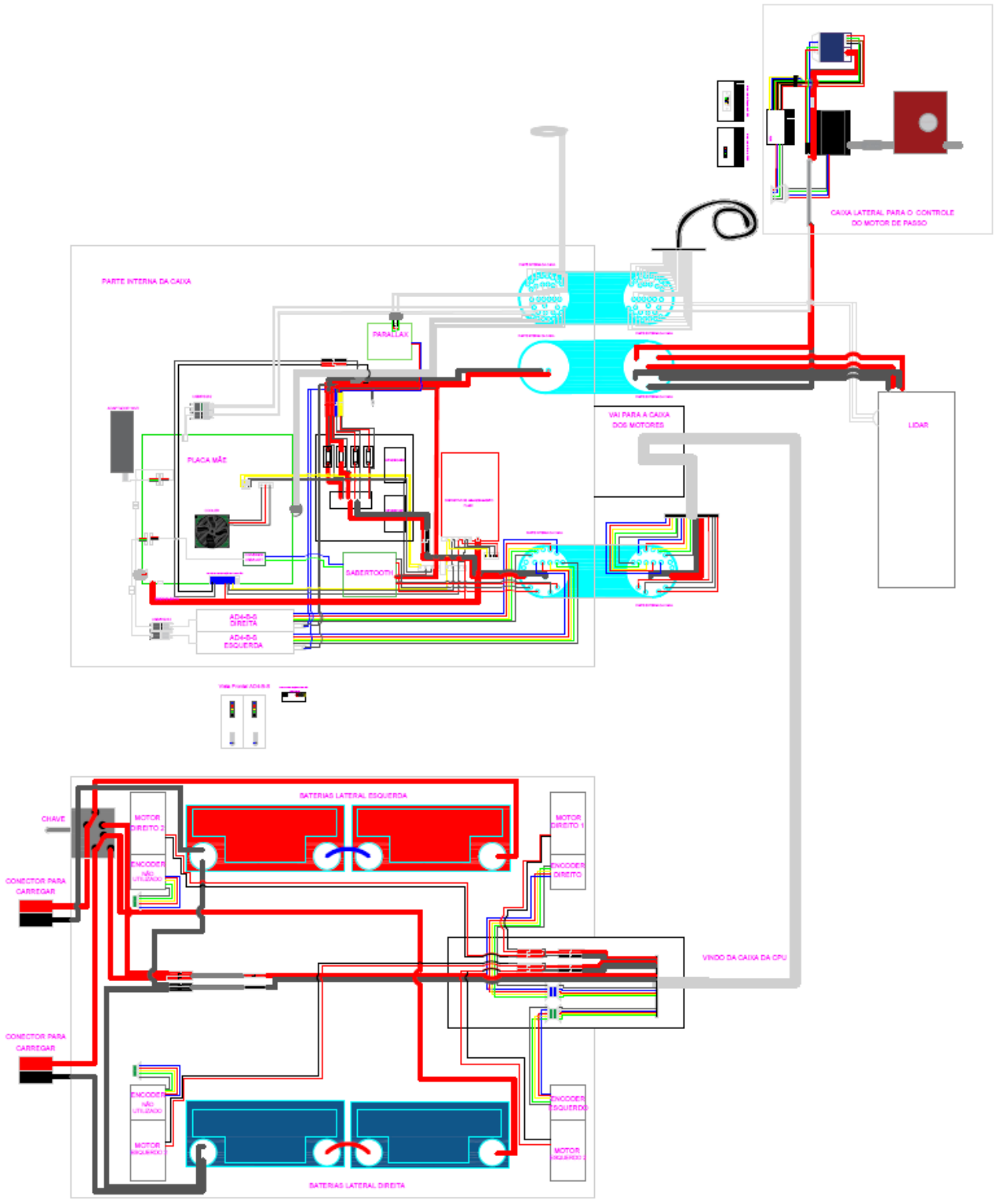
VÁRADY, Tamás; MARTIN, Ralph. R.; COX, Jordan. **Reverse Engineering of Geometric Models – An Introduction**. 1996. Disponível em: <http://ralph.cs.cf.ac.uk/papers/Geometry/RE.pdf> Acesso em: 9 ago. 2016.

WANG, Wego. **Reverse Engineering: Technology of Reinvention**. CRC Press, 2010.

WBUTTUTORIALS. **Objects Classes Methods**. Serampore, India. Disponível em <http://www.wbuttutorials.com/wbut-tutorials-objects-classes-methods.php> Acesso em: 11 set. 2016

WEST COAST BATTERIES. **PC680 Odyssey Drycell Battery**. Disponível em <http://www.odysseybatteries.com/batteries/pc680.htm> Acesso: 26 ago. 2016

APÊNDICE A – ESQUEMÁTICO GERAL DO NANOOK



APÊNDICE B –DIAGRAMA UML PARA AS CLASSES DO PROGRAMA DO NANOOK DE INTERFACE COM O USUÁRIO

ANEXO A – ILUSTRAÇÃO PARCIAL DA LEI Nº 9.609/98

Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 9.609 , DE 19 DE FEVEREIRO DE 1998.

Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

CAPÍTULO I

DISPOSIÇÕES PRELIMINARES

Art. 1º Programa de computador é a expressão de um conjunto organizado de instruções em linguagem natural ou codificada, contida em suporte físico de qualquer natureza, de emprego necessário em máquinas automáticas de tratamento da informação, dispositivos, instrumentos ou equipamentos periféricos, baseados em técnica digital ou análoga, para fazê-los funcionar de modo e para fins determinados.

CAPÍTULO II

DA PROTEÇÃO AOS DIREITOS DE AUTOR E DO REGISTRO

Art. 2º O regime de proteção à propriedade intelectual de programa de computador é o conferido às obras literárias pela legislação de direitos autorais e conexos vigentes no País, observado o disposto nesta Lei.

§ 1º Não se aplicam ao programa de computador as disposições relativas aos direitos morais, ressalvado, a qualquer tempo, o direito do autor de reivindicar a paternidade do programa de computador e o direito do autor de opor-se a alterações não-autorizadas, quando estas impliquem deformação, mutilação ou outra modificação do programa de computador, que prejudiquem a sua honra ou a sua reputação.

§ 2º Fica assegurada a tutela dos direitos relativos a programa de computador pelo prazo de cinqüenta anos, contados a partir de 1º de janeiro do ano subseqüente ao da sua publicação ou, na ausência desta, da sua criação.

§ 3º A proteção aos direitos de que trata esta Lei independe de registro.

§ 4º Os direitos atribuídos por esta Lei ficam assegurados aos estrangeiros domiciliados no exterior, desde que o país de origem do programa conceda, aos brasileiros e estrangeiros domiciliados no Brasil, direitos equivalentes.

§ 5º Inclui-se dentre os direitos assegurados por esta Lei e pela legislação de direitos autorais e conexos vigentes no País aquele direito exclusivo de autorizar ou proibir o aluguel comercial, não sendo esse direito exaurível pela venda, licença ou outra forma de transferência da cópia do programa.

§ 6º O disposto no parágrafo anterior não se aplica aos casos em que o programa em si não seja objeto essencial do aluguel.

Art. 3º Os programas de computador poderão, a critério do titular, ser registrados em órgão ou entidade a ser designado por ato do Poder Executivo, por iniciativa do Ministério responsável pela política de ciência e tecnologia. (Regulamento)

§ 1º O pedido de registro estabelecido neste artigo deverá conter, pelo menos, as seguintes informações:

I - os dados referentes ao autor do programa de computador e ao titular, se distinto do autor, sejam pessoas físicas ou jurídicas;

II - a identificação e descrição funcional do programa de computador; e

Fonte: Brasil (1998a)

ANEXO B – ILUSTRAÇÃO PARCIAL DA LEI Nº 12.853/2013

Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 12.853, DE 14 DE AGOSTO DE 2013.

Vigência

Regulamento

Altera os arts. 5º, 68, 97, 98, 99 e 100, acrescenta arts. 98-A, 98-B, 98-C, 99-A, 99-B, 100-A, 100-B e 109-A e revoga o art. 94 da Lei nº 9.610, de 19 de fevereiro de 1998, para dispor sobre a gestão coletiva de direitos autorais, e dá outras providências.

A PRESIDENTA DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Art. 1º Esta Lei dispõe sobre a gestão coletiva de direitos autorais, altera, revoga e acrescenta dispositivos à Lei nº 9.610, de 19 de fevereiro de 1998.

Art. 2º Os arts. 5º, 68, 97, 98, 99 e 100 da Lei nº 9.610, de 19 de fevereiro de 1998, passam a vigorar com as seguintes alterações:

"Art. 5º .....

.....

XIV - titular originário - o autor de obra intelectual, o intérprete, o executante, o produtor fonográfico e as empresas de radiodifusão." (NR)

"Art. 68. .....

.....

§ 6º O usuário entregará à entidade responsável pela arrecadação dos direitos relativos à execução ou exibição pública, imediatamente após o ato de comunicação ao público, relação completa das obras e fonogramas utilizados, e a tornará pública e de livre acesso, juntamente com os valores pagos, em seu sítio eletrônico ou, em não havendo este, no local da comunicação e em sua sede.

.....

§ 8º Para as empresas mencionadas no § 7º, o prazo para cumprimento do disposto no § 6o será até o décimo dia útil de cada mês, relativamente à relação completa das obras e fonogramas utilizados no mês anterior." (NR)

"Art. 97. .....

§ 1º As associações reguladas por este artigo exercem atividade de interesse público, por determinação desta Lei, devendo atender a sua função social.

§ 2º É vedado pertencer, simultaneamente, a mais de uma associação para a gestão coletiva de direitos da mesma natureza.

§ 3º Pode o titular transferir-se, a qualquer momento, para outra associação, devendo comunicar o fato, por escrito, à associação de origem.

§ 4º As associações com sede no exterior far-se-ão representar, no País, por associações nacionais constituídas na forma prevista nesta Lei.

§ 5º Apenas os titulares originários de direitos de autor ou de direitos conexos filiados diretamente às associações nacionais poderão votar ou ser votados nas associações reguladas por este artigo.

Fonte: Brasil (2013).

ANEXO C – INFORMAÇÕES PRÉVIAS DISPONÍVEIS SOBRE O NANOOK

Nanook System Description
6-29-15

Introduction

Nanook is a tracked robot platform, mounting a 2-d scanning lidar. It was developed at NASA-GSFC by a team of students under the lead of Mike Comberiate (“NASA-Mike”). The robot unit has been in both the Arctic and Antarctic. It is one of 3 similar tracked platforms. One of those is at Capitol Technology University.

Physical characteristics

Electronics

cpu - IBM-pc architecture motherboard, Intel model D945GCLF2. 32-bit dual core ATOM processor.

Intel 945GC chipset, GMA950 graphics

ATX power connector, using dc/dc, +12 to +5, +12.

memory – 240 pin DIMM SDRAM DDR2, 2 G max, 1.8 v

expansion – single ide and pci slot.

I/O expansion, 1000 Mbps lan RJ-45, dual SATA, Realtech Audio, line out/in, mic in.
PS/2 keyboard and mouse, serial and parallel port,
vga and s-video
8 usb ports.

Secondary storage, one pata, 2 sata.

Real time clock with CR-2032 battery

Motor Control

Control of the motors uses a 2-channel motor driver, a Sabretooth. It is provided with a direction bit and pwm information for the cpu. It is powered from the onboard batteries.

It communicates with the computer in the simplified serial mode. The output from the computer is via usb, which is level-shifted to ttl-level RS-232. There is a 2-wire interface to the Sabretooth module. The baud rate is selected by a dips switch on the Sabertooth. Commands are single byte.
Because Sabertooth controls two motors with one 8 byte character, when operating in Simplified Serial mode, each motor has 7 bits of resolution. Sending a character between 1 and 127 will control motor 1. 1 is full reverse, 64 is stop and 127 is full forward. Sending a character between 128 and 255 will control motor 2. 128 is full reverse, 192 is stop and 255 is full forward. Character 0 (hex 0x00) is a special case. Simplified serial is operated in option 1 (see user manual)

Motors

tbd, dc gearhead motors

Batteries

dual rechargeable lead acid, gell-cell, 12 volt, tbd A-H

Lidar

SICK LMS221-30206, Laser Measurement System, Outdoor Version, with fog correction, high range and 180 degree scanning angle, IP 67 (grey color)

LIDAR description

Features

Field of application:	Outdoor
Version:	Mid Range
Light source:	Infrared (905 nm)
MTBF:	50,000 h
Laser class:	1 (EN/IEC 60825-1), eye-safe
Field of view:	180 °
Scanning frequency:	75 Hz
	0.25 °
Angular resolution:	0.5 °
	1 °
Heating:	Yes
Operating range:	0 m ... 80 m
Max. range with 10 % reflectivity:	30 m
Fog correction:	Yes

Performance

Response time:	≥ 13 ms
Detectable object shape:	Almost any
Systematic error 1):	± 35 mm
Statistical error 2):	± 10 mm
Integrated application:	Field evaluation
Number of field sets:	2 field triples (6 fields)
Simultaneous processing cases:	1 (3 fields)

1) 2) Typical value; actual value depends on environmental conditions.

Interfaces

Serial (RS-232, RS-422):

Function (Serial (RS-232, RS-422)):	Host, AUX
Data transmission rate (Serial (RS-232, RS-422)):	RS-232: 9.6 / 19.2 kBd and RS-422: 9.6 / 19.2/ 38.4/ 500 kBd

Switching inputs:	1
Switching outputs:	3
Optical indicators:	0

Mechanics/electronics	
Electrical connection:	1 16-pin plug
Operating voltage:	≤ 24 V DC
Power consumption:	30 W, + 140 W heating
Housing:	Base plate aluminum diecast Covering PU
Housing color:	Gray (RAL 7032)
Enclosure rating:	IP 67
Protection class:	II (VDE 0106/IEC 1010-1) 1)
Weight:	9 kg
Dimensions:	241 mm x 351 mm x 265 mm
1) Insulated	
Ambient data	
Object remission:	1.8 % ... 10,000 %
Electromagnetic compatibility (EMC):	EN 61000-6-2 / EN 61000-6-3 / A11 (2004-07)
Vibration resistance:	IEC 68
Shock resistance:	IEC 68
Ambient operating temperature:	-30 °C ... +50 °C
Storage temperature:	-30 °C ... +70 °C
General notes	
Note on use:	Not suitable for personnel protection
Life cycle phase:	Phased out

Fonte: Fornecido por Patrick H. Stakem.