FACULDADE DE TECNOLOGIA SENAI CIMATEC

Rebeca Tourinho Lima

Método de monitoramento de servomotores para o subsistema de confiabilidade de um robô autônomo de inspeção

Brasil

Fevereiro, 2017

FACULDADE DE TECNOLOGIA SENAI CIMATEC

Rebeca Tourinho Lima

Método de monitoramento de servomotores para o subsistema de confiabilidade de um robô autônomo de inspeção

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Modelagem Computacional e Tecnologia Industrial do SENAI CIMATEC como requisito parcial para a obtenção do título de **Mestre**.

Orientador: Prof. Dr. Valter de Senna

Coorientador: Prof. Dr. Roberto Luiz Souza Monteiro

Faculdade de Tecnologia SENAI CIMATEC

Modelagem Computacional e Tecnologia Industrial

Programa de Pós-Graduação

Brasil

Fevereiro, 2017

Ficha catalográfica elaborada pela Biblioteca da Faculdade de Tecnologia SENAI CIMATEC

L732m Lima, Rebeca Tourinho

Método de monitoramento de servomotores para o subsistema de confiabilidade de um robô autônomo de inspeção / Rebeca Tourinho Lima. – Salvador, 2017.

69 f. : il. color.

Orientador: Prof. Dr. Valter de Senna.
Coorientador: Prof. Dr. Roberto Luiz Souza Monteiro.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Faculdade de Tecnologia SENAI CIMATEC, Salvador, 2017.
Inclui referências.

1. Instrumentação eletrônica. 2. Monitoramento de servomotores. 3. Confiabilidade. 4. Robótica móvel. I. Faculdade de Tecnologia SENAI CIMATEC. II. Senna, Valter. III. Monteiro, Roberto Luiz Souza. IV. Título.

CDD: 629.892

NDI - 01

Este trabalho é dedicado ao meu companheiro Lucas e à minha família por todo o suporte, incentivo e compreensão ao longo deste período, sem os quais eu não teria alcançado.

Agradecimentos

Meus agradecimentos são ao SENAI CIMATEC que me ofereceu esta oportunidade através de uma bolsa de estudos, aos professores Valter e Roberto que com suas experiências contribuíram para o desenvolvimento deste trabalho, ao Lucas, à minha família e ao Marco Reis pelo apoio na concepção deste projeto de pesquisa voltado para robótica.

Resumo

Um robô móvel autônomo de inspeção em linhas de alta tensão intitulado PI-Ro deve detectar anormalidades nas instalações das concessionárias de energia elétrica à medida que percorre e transpõe obstáculos nas linhas energizadas. Propõe-se que uma inteligência computacional o permita tomar decisões sobre as condições de sua própria operação, evitando a inoperância inesperada ou possíveis acidentes que impactem no fornecimento de energia. Para isso, o robô PI-Ro será dotado de um subsistema de confiabilidade continuamente atualizado com informações das condições reais dos seus atuadores. Tais informações serão providas por uma instrumentação eletrônica independente a ser desenvolvida neste trabalho. A este sistema de aquisição estará associado um método de monitoramento em tempo real das grandezas críticas ao funcionamento das juntas do PI-Ro. O resultado desse monitoramento será entrada para uma base de dados das falhas ocorridas durante a operação e portanto, permitirá ao subsistema de confiabilidade efetuar sua análise.

Palavras-chave: instrumentação. monitoramento. confiabilidade. robótica móvel.

Abstract

An autonomous powerline inspection mobile robot named PI-Ro must detect issues on transmission line installations of electrical energy concessionaries as long as the robot moves through cabling and obstacles belonging to high-voltage energized powerlines. It is proposed that a computational intelligence provides to this robot the ability to take decisions about its own condition operation, avoiding non operating condition or possible accidents, which impacts directly on energy supply. For this purpose, PI-Ro will have a continuously updated reliability subsystem with actuators' condition information of its robotic joints. These information will be deployed by an independent electronic instrumentation to be developed in this work. A real time monitoring method will be associated with this acquisition system gathering critical information about PI-Ro joints. The result of monitoring will be input to during operation detected failures database allowing reliability subsystem proceed with analysis.

Keywords: instrumentation. monitoring. reliability. mobile robotics.

Lista de ilustrações

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
AI	Artificial Intelligence (Inteligência Artificial)
BIR	Brazilian Institute of Robotics (Instituto Brasileiro de Robótica)
CEMIG	Companhia Energética de Minas Gerais
CI	Computational Intelligence (Inteligência Computacional)
CPU	Central Processing Unit (Unidade Central de Processamento)
EEPROM	Electrically-Erasable Programmable Read-Only Memory (Memória Somente-Leitura Programável Apagável Eletricamente)
ELEVA	Electric Power Line Exploration using Aerial Vehicle (Exploração de Linhas Elétricas usando Veículos Aéreos)
EIA	Electronic Industries Association (Associação das Indústrias de Eletrônica)
EMI	Electromagnetic Interference (Interferência Eletromagnética)
EUA	Estados Unidos da América
FHA	Fault Hazard Analysis (Análise de Perigo de Falhas)
FMEA	Failure Mode and Effects Analysis (Análise de Modos e Efeitos de Falhas)
FMECA	Failure Mode Effect and Criticality Analysis (Análise de Modos, Efeitos e Criticidade de Falhas)
FTA	Failure Tree Analysis (Análise de Árvore de Falhas)
I^2C	Inter-Integrated Circuit (Circuito Inter-Intergrado)
I/O	Input-Output (Entrada-Saída)
IDE	Integrated Development Enviroment (Ambiente de Desenvolvimento Integrado)
MTBF	Mean Time Between Failures (Tempo Médio entre Falhas)
MTTR	Mean Time To Repair (Tempo Médio para Reparo)

NBR	Norma Brasileira
P&D&I	Pesquisa, Desenvolvimento e Inovação
PCI	Placa de Circuito Impresso
PHA	Preliminary Hazards Analysis (Análise Preliminar de Perigos)
PID	Proporcional, Integral e Derivativo
RAM	Random Access Memory (Memória de Acesso Aleatório)
RPN	Risk Priority Number (Número de Prioridade de Risco)
SD	Secure Digital (Digital Protegido)
SO	Sistema Operacional
SPI	Serial Peripheral Interface (Interface de Periférico Serial)
TI	Texas Instruments
UART	Universal Asynchronous Receiver Transmitter (Transmissor Receptor Assíncrono Universal)
UAV	Unmanned Autonomous Vehicle (Veículo Autônomo Não-Tripulado)
USB	Universal Serial Bus (Barramento Serial Universal)
USD	United States Dollar (Dólar Americano)
VANT	Veículo Aéreo Não-Tripulado

Sumário

1 Introdução

A competitividade industrial vem evidenciando a necessidade de dotar os mais simples mecanismos de independência operacional. Essa é uma tendência evidenciada, por exemplo, por uma onda de robôs (e veículos) autônomos em desenvolvimento ou já disponíveis no mercado, que são sistemas robóticos dotados de uma autonomia em seus ambientes de atuação – no ar, água, terra ou espaço extraterrestre - isto é, sem a necessidade de operação e supervisão humana.

Para que um robô autônomo, por sua vez, seja capaz de se adaptar e lidar com situações adversas nestes ambientes, é necessário que o mesmo seja dotado não só de complexidade sensorial e construtiva, mas também de inteligência computacional capaz de lidar com eventualidades. Desta forma, uma vez em operação, ele se torna capaz de tomar decisões fundamentais e atuar, sem a interferência humana, a partir de condições parciais ou inteiramente novas. Uma tomada de decisão, por sua vez, exige autoconhecimento, obtido a partir de informações do seu próprio sistema que, por sua vez, são interpretadas a partir de uma base de dados, além da ininterrupta construção e renovação dessa base a partir de medições coletadas durante o processo de operação, indicando assim as mudanças sofridas ao longo da operação do equipamento.

Existe uma sutil diferença entre inteligência computacional (*computational intelligence, CI*) e inteligência artificial (*artificial intelligence, AI*). A primeira permeia os campos das redes neurais, computação evolucionária e lógica *fuzzy*. Juntamente com o *soft computing* (raciocínio probabilístico, inteligência de enxame, teoria do caos), a computação natural e a mineração de dados (*data mining*) compõe essencialmente um conjunto de estratégias matemáticas se que diferenciam pela origem e contexto de aplicação (KUHN, 2016). Inteligência artificial, por sua vez, é um termo mais abrangente, de forma que os métodos de inteligência computacional podem ser dedicados à parte *soft* da AI, enquanto a parte principal (*hard*) diz respeito a sistemas especialistas, lógica formal etc. Infelizmente, os vários termos e definições ainda não são consistentes na literatura, no entanto, é consenso que existe uma AI fraca e uma AI forte considerada verdadeiramente inteligente e que combina métodos *soft* e *hard*.

Dado que robôs utilizados em aplicações industriais devem atender a certos níveis de qualidade no desempenho de tarefas críticas, a importância e responsabilidade de efetuar tais atividades recai na capacidade de estas máquinas estarem disponíveis, serem confiáveis e completarem cada uma destas tarefas com sucesso. Uma das formas de avaliar os riscos envolvidos e a confiabilidade de um equipamento são as análises FMEA (*Failure Mode and Effects Analysis* - Análise de Modos de Falhas e Efeitos) e FTA (*Failure Tree Analysis*

- Análise de Árvore de Falhas). Esses estudos, no entanto, são escassos quando se tratam de robôs voltados a aplicações industriais específicas, ou seja, altamente especializados, pois a grande maioria destes ainda se encontra em baixos e médios níveis de maturidade tecnológica (*Technology Readiness Level*).

FMEA consiste em uma metodologia de avaliação de riscos capaz de classificá-los com base nas probabilidades de falhas, seus respectivos modos (formas) de ocorrência e efeitos, de acordo com três fatores quantitativos: Severidade, Ocorrência e Detecção. A partir destes índices obtém-se um Número de Prioridade de Risco (Risk Priority Number - RPN) que indica a potencialidade global de cada risco envolvido. Já a FTA consiste no mapeamento das causas para as falhas e das consequências em ocorrência das mesmas de forma a equacionar e estabelecer as relações lógicas existentes para o encadeamento de outras falhas e previsão de comportamento do sistema.

São ditos defeitos os processos ou definições incorretas de determinados dados que podem ocorrer na camada de hardware ou em uma linha de código. Defeitos causam erros quando estes passos específicos de algoritmo são executados (FERREIRA, 2007). Erros, portanto, só ocorrem durante a execução do programa e resultam em estados inconsistentes ou inesperados. Os erros são a causa das falhas, mas estas só ocorrerão se o erro for apresentado. Entende-se por falha (ou falta) a consequência física ou algorítmica do erro. Quando sistemas computacionais continuam operando corretamente mesmo após falhas em alguns de seus componentes, diz-se que estes são tolerantes a falha.

Prover dados de condições de operação à inteligência computacional de um robô, no que diz respeito a seus sensores e atuadores, por exemplo, permite aperfeiçoar o processo de tomada de decisão dentro de uma missão pré-estabelecida conforme são monitorados os estados destes sensores e atuadores ao longo de sua vida útil.

1.1 PI-Ro - *Powerline Inspection Robot*

Segundo o Balanço Energético Nacional de 2015: Ano base 2014 publicado pela Empresa de Pesquisa Energética - EPE, o consumo final de eletricidade no Brasil vem aumentando gradativamente nos últimos 10 anos e atingiu cerca de 17,2% do consumo final por fonte de energia da matriz energética, correspondente a 45.655 10^3 tep (toneladas equivalentes de petróleo). Destes, 63,7% atendem pelos consumos de energia elétrica dos setores industrial (38,8%) e residencial (24,9%) (EMPRESA DE PESQUISA ENERGéTICA - EPE, 2015).

No Sistema Interligado Nacional (SIN), formado por empresas das regiões Sul, Sudeste, Centro-Oeste, Nordeste e parte da região Norte, apenas 1,7% da energia requerida pelo país encontra-se em sistemas isolados localizados principalmente na região amazônica (ONS, 2016b). Dessa forma, de acordo com dados do relatório da Operação do SIN de

2014, existem 125.639,6 km de rede básica (instalações com tensão maior ou igual a 230 kV)(ONS, 2016a), o que sugere um grande potencial de demanda para a aplicação de robôs de inspeção em linhas de transmissão de alta tensão.

As linhas de transmissão de energia elétrica são formadas por cabos condutores, geralmente nus, que se interligam por torres de transmissão cuja função é erguer estes cabos a uma altura considerada segura e livre do contato com pessoas, veículos, animais ou vegetação local. Tais linhas de transmissão são mantidas em funcionamento de forma ininterrupta e são compostas de elementos como: espaçadores, que evitam o contato dos cabos - o que causaria um curto-circuito - e amortecem os esforços físicos causados pelos ventos; esferas de sinalização, que servem de alerta para pequenas aeronaves no espaço aéreo; e pelos isoladores que minimizam a energia dissipada no processo de transmissão e suportam o peso dos cabos. Em geral, os isoladores são fabricados com polímeros, cerâmica ou vidro (TECNOGERA, 2015a).

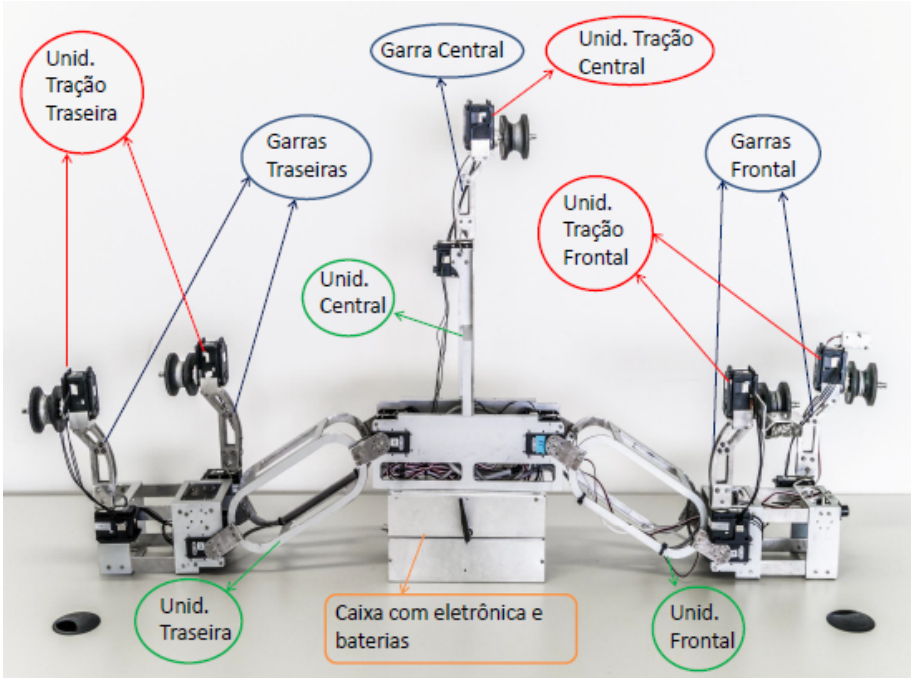
As concessionárias, tanto para atender às leis e normas vigentes quanto para buscar a melhoria contínua dos seus indicadores de qualidade de energia, mantém um programa de manutenção em que, algumas das atividades, são de altíssimo risco para seus funcionários. Particularmente, nas linhas de transmissão de alta tensão são realizadas inspeções através de veículos aéreos tripulados que se aproximam dos cabos energizados (i.e. linhas vivas) para que os especialistas realizem medições ou mesmo mantenham contato com as linhas. Embora sejam utilizados equipamentos de proteção, a exposição é muito grande e os riscos operacionais podem resultar em acidentes fatais (RANGEL; KIENITZ; BRANDãO, 2009). Eventualmente, más condições climáticas ou a própria geografia da região podem ser fatores que dificultam o sobrevoo. Formas alternativas de inspeção são realizadas através de veículos terrestres - um método muito limitado - e mais recentemente, por veículos aéreos não-tripulados (VANTs). As inspeções tem como objetivo verificar a integridade física das linhas de transmissão localizando fissuras, corrosões e danos que possam interferir na qualidade do fornecimento de energia.

Nesse contexto, a proposta do robô PI-Ro - *Powerline Inspection Robot*, projeto de P&D&I do SENAI CIMATEC, é realizar inspeções em linhas de alta tensão de forma autônoma, percorrendo as linhas como se estas fossem trilhos. O robô, por sua vez, possui a capacidade de ultrapassar obstáculos como emendas, grampos de suspensão e amortecedores e se propõe a realizar inspeções tanto visuais, quanto térmicas (ALCANTARA, 2015).

O PI-Ro possui uma estrutura em alumínio que combina características de manipuladores robóticos e robôs móveis, pesa cerca de 9,0 Kg incluindo sensores, eletrônica embarcada e bateria. Necessita de operadores apenas para a sua instalação e remoção do trecho de linha inspecionado. Mecanicamente, possui duas unidades manipuladoras e um unidade de apoio central (ALCANTARA, 2015). Se fixa à linha de transmissão através de garras e se desloca a partir das unidades de tração conforme ilustrado na figura 1, cujos

atuadores são 18 servomotores, 5 deles associados a garras com roldanas.

Figura 1 – PI-Ro - *Poweline Inspection Robot.*



Fonte: Alcantara (2015)

Possui uma unidade de processamento central, i.e. um computador ultra compacto com sistema operacional (SO) Linux e distribuição Ubuntu, que é de código livre e aberto, e portanto, permite modificações no código fonte do SO para atender a necessidades específicas dos pesquisadores durante o processo de desenvolvimento. Na unidade de processamento é desenvolvida a inteligência computacional do robô, o que combina aquisição e tratamento de dados, tomada de decisões e demais algoritmos necessários ao funcionamento do mesmo.

O protótipo original do robô acima descrito atualmente pertence à Companhia Energética de Minas Gerais - CEMIG, empresa parceira do SENAI CIMATEC na ocasião do desenvolvimento deste primeiro conceito através de um termo de cooperação. Sendo assim, uma vez concluído esse projeto, o SENAI CIMATEC iniciou uma nova fase de P&D&I, através do Brazilian Institute of Robotics - BIR, com o intuito de aperfeiçoar a tecnologia em questão, além de conceber o seu próprio protótipo. Portanto, embora o seu conceito mecânico tenha sido majoritariamente preservado, o projeto passa atualmente por uma reformulação da arquitetura eletroeletrônica e de software.

1.2 Definição do Tema

No âmbito do desenvolvimento acadêmico e tecnológico do novo protótipo do robô de inspeção em linhas de alta tensão (PI-Ro) se inserem alguns temas de pesquisa tais como: a criação da inteligência computacional que o tornará autônomo, o sistema de gerenciamento dos subsistemas, o sistema de controle das juntas que mantém a sustentação e coordenam a movimentação, o sistema de navegação que deve fazê-lo ser capaz de transpor obstáculos, o sistema de inspeção para detecção de anormalidades na linha, o sistema de confiabilidade para garantir a qualidade e a segurança da operação e ainda a instrumentação eletrônica que responde pela coleta de dados. Para desempenhar tais funções, cada um destes subsistemas do robô requer conceito e arquitetura próprios, além de algoritmos específicos.

Do ponto de vista da confiabilidade, a proposta do robô PI-Ro (Figura 2) é que ele seja capaz de monitorar o seu próprio funcionamento e condições de seus componentes para agir conforme esses estados. Medidas de tensão elétrica, temperatura, corrente elétrica, carga, entre outros parâmetros devem ser coletadas para que algoritmos internos ao robô possam calcular taxas de falhas e confrontá-las com dados de uma matriz de contradição através de uma análise ontológica. Essa análise proverá informações para uma inteligência artificial bayesiana tomar decisões e acionar os atuadores, que por sua vez serão constantemente monitorados, fechando o ciclo.

Diante destes desafios necessários à realização de um novo projeto do robô, um dos primeiros desenvolvimentos diz respeito à instrumentação eletrônica do PI-Ro, ainda que o protótipo não tenha sido construído. Sendo assim, este trabalho tem o objetivo geral de desenvolver o método de monitoramento contínuo das condições das juntas do robô PI-Ro, i.e. dos servomotores que proveem toda a sua mobilidade. Este monitoramento irá gerar um banco de dados com todo o histórico das condições de operação das juntas do robô em cada missão, i.e. um conjunto de medições dos parâmetros de engenharia relevantes, incluindo suas falhas. Esses dados alimentarão o sistema de confiabilidade e serão coletados através de um dispositivo embarcado independente do robô e que irá interagir diretamente com seus atuadores. A partir desses dados coletados pelo sistema de aquisição, o subsistema de confiabilidade poderá calcular iterativamente a taxa de falha dos atuadores do robô e tomar decisões. Como objetivos específicos deste trabalho, portanto, tem-se:

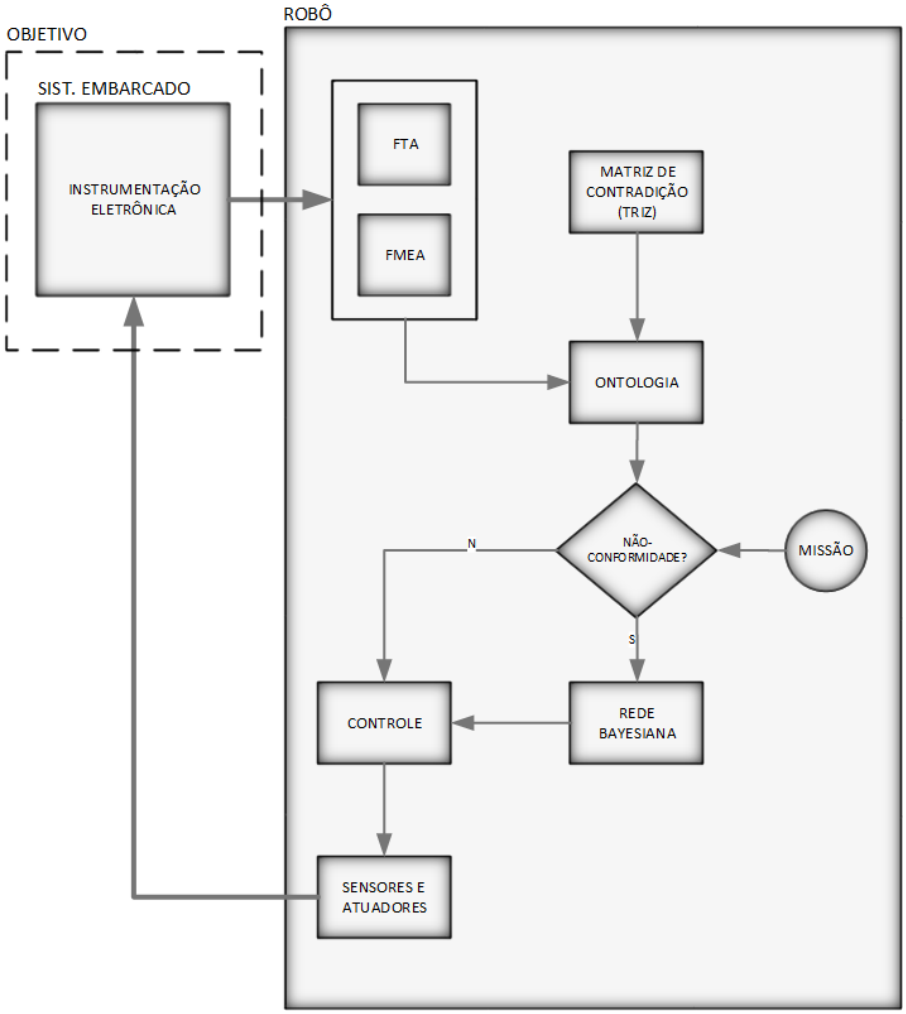
- Realizar a instrumentação eletrônica necessária, através do uso de um sistema embarcado, para extrair informações sobre as condições dos atuadores do robô de inspeção;
- Desenvolver algoritmo para tratamento e comunicação de dados oriundos da instrumentação indicada para atualização do banco de dados de referência para o

subsistema de confiabilidade do robô de inspeção.

Este trabalho contribuirá para o processo de tomada de decisões baseado na confiabilidade da operação do robô e o escopo, portanto, se refere ao monitoramento dos atuadores deste robô através de instrumentação eletrônica com a proposição de um método que favoreça a análise de confiabilidade.

Assim, a inteligência artificial do PI-Ro poderá controlar as ações do robô e, ter sua base de dados alimentada de forma complementar e não única, com informações providas por análises oriundas do subsistema de confiabilidade, que por sua vez as realizará de forma iterativa à medida que o subsistema de instrumentação a ser desenvolvido neste trabalho coletar dados acerca das condições das juntas.

Figura 2 – Subsistemas do robô de inspeção PI-Ro: Diagrama parcial.



Fonte: Própria.

1.3 Organização do Texto

No capítulo 2, a seguir, apresenta-se a fundamentação teórica requerida neste trabalho no que diz respeito aos conceitos acerca de robótica e como interagem com análises de confiabilidade e conceitos de eletrônica embarcada.

No capítulo 3 é apresentada a metodologia para implementação do sistema embarcado proposto e a descrição detalhada das etapas de desenvolvimento do mesmo, desde a definição das especificações-meta, passando pela definição de componentes a serem utilizados até a concepção do algoritmo e abordagens escolhidas.

No capítulo 4 são reunidos os resultados referentes aos ensaios de validação da metodologia de coleta de dados.

No capítulo 5 traz-se as considerações sobre a contribuição deste projeto para a comunidade técnica e acadêmica bem como as sugestões de trabalhos futuros.

2 Fundamentação Teórica

2.1 Introdução

Este capítulo apresentará os principais fundamentos e referências requisitadas na motivação, concepção e desenvolvimento deste trabalho. Discorre-se sobre a atualidade da robótica móvel voltada para inspeção de linhas de transmissão, sobre os trabalhos que envolvem monitoramento de parâmetros de engenharia direcionados a análises de confiabilidade e alguns fundamentos acerca das técnicas para análise de falhas, bem como sobre a eletrônica embarcada que viabilizará o implementação do método de monitoramento pretendido.

2.2 Robótica para Inspeção em Linhas de Transmissão

A robótica tradicional é extremamente popular e bem-sucedida na manufatura industrial. Braços robóticos, também conhecidos por manipuladores são caracterizados por serem instalados numa determinada posição em uma linha de montagem de forma que podem atuar com grande velocidade e precisão na execução de tarefas repetitivas tais como soldagem e pintura na indústria automotiva à montagem de equipamentos eletrônicos. Apesar de sua versatilidade, os robôs manipuladores possuem a desvantagem de não terem mobilidade.

Baseada em disciplinas das engenharias e ciências, o campo recente da robótica móvel se estende da mecânica, elétrica e eletrônica à computação e ciências social e cognitiva. Um robô móvel, em contrapartida, pode circular pela planta de manufatura, com flexibilidade tal que pode ser utilizado onde for mais eficiente. A preocupação com o fato de um robô ter mobilidade no mundo real sem supervisão faz com que a confiabilidade esperada seja ainda mais significativa.

Sistemas teleoperados ganharam popularidade em ambientes hostis, perigosos e inóspitos favorecendo o desenvolvimento dos mais incomuns mecanismos de locomoção. No entanto, a complexidade desses sistemas torna difícil o controle direto humano, de forma que o homem assume o desempenho das atividades de localização e cognição mas confia ao sistema de controle a movimentação dos robôs. Em outros casos, quando robôs dividem o espaço com o homem não é por sua mobilidade, mas sim por sua autonomia e, portanto, pela sua habilidade de manter um senso de posição e navegar sem a intervenção humana.

Embora robôs móveis tenham uma variedade de aplicações e mercados, seu design envolve a integração de muitas áreas do conhecimento. Isso torna a robótica móvel

extremamente interdisciplinar. Para resolver questões de locomoção é necessário entender de mecanismos e cinemática; dinâmica e teoria de controle. Para criar sistemas de percepção robustos é necessário adentrar na análise de sinais e visão computacional, assim como combinar diversas tecnologias de sensores. Localização e navegação requer conhecimentos em algoritmos computacionais, teoria da informação, inteligência artificial e teoria da probabilidade (SIEGWART; NOURBAKHSH, 2004).

No contexto da automação para inspeção em linhas de distribuição de energia elétrica, as primeiras tentativas foram no campo das inspeções através do uso de helicópteros, nos quais a tripulação coleta dados a partir de câmeras coloridas, térmicas e de detecção de efeito corona. Esse efeito pode causar queda na capacidade da linha, podendo resultar na perda de centenas de quilowatts por quilômetro de condutor elétrico (TECNOGERA, 2015b). Embora seja uma abordagem rápida em relação às patrulhas terrestres, as inspeções em helicópteros são uma alternativa cara, pouco precisa e que ainda expõem os trabalhadores a grandes riscos (PERGAM USA, 2015). Nos últimos anos, as pesquisas nessa área, portanto, tem se direcionado ao uso de VANTs (Veículos Aéreos Não Tripulados), também conhecidos por UAVs (Unmanned Autonomous Vehicles), que podem voar sobre as linhas de transmissão (Fig. 3) e de robôs trepadores - *climbing robots* - capazes de se locomover sobre os condutores (Fig. 4).

Figura 3 – Helicóptero versus VANT.

Fonte: Pergam USA.

VANTs precisam voar de forma autônoma sobre as linhas de transmissão e localizar defeitos além de requererem eficiência energética para cobrir longas distâncias. Com isso, os desafios de projeto são o controle de posição, o planejamento automático de trajetória, o desvio de obstáculos, a comunicação, aquisição de imagens, detecção de falhas, a manutenção da distância segura à linha de transmissão e até mesmo a alimentação direta da linha.

Já os robôs trepadores devem ser capazes de transpor obstáculos no mínimo de forma semi-autônoma, percorrer de forma autônoma os condutores e realizar inspeções

Figura 4 – Robô trepador LineScout.

Fonte: Montambault e Pouliot.

visuais. Outro desafio é a blindagem eletromagnética, uma vez que o campo elétrico na superfície dos condutores pode ser de cerca de 1,5 MV/m em condições normais ou ainda maiores na presença de defeitos na linha (KATRASNIK; PERNUS; LIKAR, 2010). Por outro lado, o campo magnético pode ser objeto de estudo para alimentação do próprio robô.

Os principais defeitos que ocorrem em linhas de transmissão são apresentados nos condutores, isoladores e nas torres de sustentação. Dois dos defeitos mais comuns em condutores de alumínio são os danos mecânicos (fissuras) que causam aumento do campo elétrico e, portanto, o efeito corona; e a corrosão, que provoca aumento de temperatura no condutor.

VANTs tem desafios muito similares aos das inspeções com helicópteros automatizados: estabilização de câmeras, rastreamento de alvos e detecção automática de defeitos. Os principais desenvolvimentos com VANTs para esta aplicação são do grupo de pesquisa da Universidade de Wales (JONES, 2005), do projeto ELEVA - Electric Power Line Exploration using Aerial Vehicle (CAMPOY et al., 2001) e mais recentemente da Universidade do Texas (ZHOU et al., 2016).

A principal vantagem dos robôs trepadores é a precisão da inspeção visto que atuam em grande proximidade da linha e com baixas vibrações aumentando a qualidade das aquisições de imagens. Por outro lado, o conceito eletromecânico para viabilizar a missão e torná-lo capaz de percorrer a linha e transpor obstáculos é muito mais complexo. Dados os desafios, um dos primeiros trabalhos remontam ao início da década de 90 (SAWADA et al., 1991) e já propõe o conceito de um robô autônomo capaz de transpor torres de

transmissão. Trabalhos mais recentes, se concentram no desenvolvimento de sistemas de controle (TANG; WANG; FANG, 2004), (LUDAN et al., 2006), simulação dinâmica (XIAO et al., 2005), novas técnicas de detecção de falhas (TAVARES; SEQUEIRA, 2007), transposição de obstáculos (MONTAMBAULT; POULIOT, 2006) e na proposição de robôs compactos com grande habilidade de locomoção (WEI et al., 2015).

O robô LineScout (Fig. 4) oriundo da pesquisa de MONTAMBAULT; POULIOT hoje é comercializado pela MIR Innovation, subsidiária da concessionária estatal canadense Hydro-Québec. É capaz de cruzar um certo número de obstáculos, tais como amortecedores, isoladores e anéis de proteção. Possui cerca de 115 Kg, dimensões $1,37m \times 0,85m \times 0,90m$, velocidade de 1 m/s, bateria com capacidade de 5 a 9 horas, temperatura de operação de $-10°$C a 35°C, lidando com inclinações de cabos de até 35° (MIR INNOVATION, 2016).

2.3 Revisão da Literatura

Considerando o propósito da geração de um banco de dados de monitoramento das condições das juntas do robô PI-Ro é preciso entender de que forma essa medições irão contribuir para o sistema de confiabilidade do robô que se apóia na sua inteligência computacional.

Dessa forma, em Dhillon e Fashandi (1997) são discutidas técnicas de análise que englobam segurança e confiabilidade em sistemas robóticos. Neste artigo afirma-se que a FTA é uma técnica mais quantitativa, que geralmente envolve a aplicação da teoria da probabilidade para quantificar os riscos de cada evento ou componente de um sistema, atribuindo-lhe uma taxa de falha. Considerando algumas técnicas citadas no artigo, os autores sugerem que a FTA e FMEA seriam as técnicas mais apropriadas para análise de segurança em robôs. Para análise de confiabilidade, uma vez considerados fatores como custo, simplicidade e eficácia, são sugeridos os seguintes métodos: FMEA, FTA, diagrama de blocos, modelos combinacionais, modelos Markovianos e não-Markovianos e a simulação de Monte-Carlo.

Segundo o autores, a vantagem da FMEA é levantar hipóteses da fonte de falha, reduzindo a probabilidade de falha ou a severidade através de ações de contenção ou *redesign* do sistema. Apesar disso, FMEA não é aplicável quando efeitos de duas ou mais falhas são combinados como ocorre em sistemas de maior complexidade, sendo esta a sua maior desvantagem. O método de Markov tem sido usado largamente para estudos de confiabilidade de sistemas complexos. O método é útil para situações em que as falhas não são independentes e considera que as taxas de transição do sistema variante no tempo são constantes. Não sendo possível assumir essa condição, recai-se em um modelo não-Markoviano onde são aplicadas técnicas específicas.

Outros trabalhos, mais específicos, mostram como diferentes formas de monitora-

mento de grandezas podem contribuir para análises de confiabilidade.

Em Peng et al. (2007) utiliza-se controle estatístico de processo para monitoramento online de um processo de manufatura. Detecta-se eventos relevantes em tempo real com base no modelo de produção digital do MES. As falhas que ocorrem são diagnosticadas usando tabelas de controle, o método FTA e uma base de conhecimento. Em Li et al. (2012) são utilizados de dados de múltiplos sensores para monitoramento de condições e diagnóstico de falhas através da combinação do algoritmo FastICA, short time Fourier Transform (STFT) e Fuzzy-neural network (FNN).

Em Papadopoulos (2003) é proposto um monitoramento de segurança genérico que pode operar com diagramas de estados e árvores de falhas para aprimorar a detecção online, o diagnóstico e controle de riscos. Neste artigo o estudo de caso é um sistema de combustível de uma aeronave e nesse âmbito, são referências o NPPC (UNDERWOOD, 1982) que se trata de uma planta de energia nuclear computacional experimental cujos procedimentos de inferência interpretam situações particulares do processo com base em um modelo do que são considerados eventos normais ou anormais e o REACTOR (NELSON, 1982) se trata de um sistema especialista capaz de assistir aos operadores no diagnóstico e tratamento de acidentes em reatores nucleares. Ambos monitoramentos ajudaram operadores de plantas a determinar as causas de eventos anormais através do registro das medições da instrumentação, observando os desvios das condições normais de operação.

Em Duan e Zhou (2012) apresenta-se uma nova abordagem para FTA e redes bayesianas para diagnóstico de falhas com base em um caso de estudo de um sistema de pressão de óleo de uma aeronave. Já em Jingwei et al. (2011) apresenta-se um novo modelo matemático, o modelo proporcional, para monitoramento de falhas devido a desgaste com base em análise espectrométrica de óleo, detectando a concentração dos principais elementos químicos. No caso de estudo, um motor foi avaliado por cerca de 850h (mais de 35 dias) com amostras tomadas de 5 em 5 horas.

Em Montani, Bobbio e Codetta-Raiteri (2008) apresenta-se uma ferramenta computacional que permite analisar uma árvore de falhas dinâmica (DFT) com base em sua conversão em uma rede Bayesiana, explorando algoritmos clássicos de inferência para calcular confiabilidade.

Em Shalev e Tiran (2007) é introduzido um novo método de análise de árvores de falhas, o método *Condition-Monitoring Fault Tree Analysis* (CBFTA).

Geralmente a FTA é realizada durante a fase de *design* de um sistema e utiliza taxas de falhas de componentes provenientes de *handbooks*, manuais de fabricantes, padrões industriais e padrões militares. O método CBFTA se origina de uma FTA conhecida, porém utiliza métodos de monitoramento das condições do sistema (CMs) tais como análises

de vibrações, análises de lubrificação, análises de corrente elétrica, entre outros, para determinar taxas de falhas atualizadas dos componentes sensíveis e críticos, trazendo uma análise mais acertada. Durante a operação de um sistema, sabe-se que a sua confiabilidade muda com o tempo devido à deterioração de componentes básicos. Portanto, em virtude de condições severas às quais estes componentes estão expostos e/ou processos de manutenção insuficientes e inadequados, as taxas de falha aumentam. Na maioria dos casos, portanto, o verdadeiro MTBF (*Mean Time Between Failures* ou Tempo Médio Entre Falhas) é menor do que o projetado, o que torna a FTA simples não muito aceita.

Dessa forma, os autores propõem que com o método CBFTA as taxas de falhas sejam atualizadas aplicadas à FTA conhecida. O CBFTA, então, recalcula periodicamente a taxa de falha do evento principal da árvore de falhas (*top event*) e determina tanto a probabilidade de uma falha no sistema, como a probabilidade de sucesso da operação, ou seja a confiabilidade do sistema. O nome CBFTA foi escolhido em distinção ao *Dynamic Fault Tree* (DFT) e ao *Real Time Fault Tree*, enfatizando que a análise é baseada em condições medidas no sistema. Como exemplo de aplicação do método proposto, são analisadas as falhas em rolamentos de um sistema duplo de bombas mostrando através de medições e CBFTA como a "curva da banheira", i.e. a curva da vida útil destes componentes se modifica apresentando um fim mais precoce do que aquele tradicionalmente esperado.

Essa abordagem e os resultados apresentados por Shalev e Tiran mostram o potencial do monitoramento contínuo para uma análise de falhas mais realista e mais precisa. Dessa forma, o robô de inspeção PI-Ro poderá por exemplo, determinar se uma de suas juntas tem condições de realizar determinado movimento e seguir ou não com a inspeção sem colocar em risco a operação.

2.4 Confiabilidade

Confiabilidade pode ser definida como a probabilidade de algo realizar a sua função adequadamente por um período de tempo desejado quando o mesmo está em operação de acordo com as especificações definidas para tal. A este conceito tem-se diretamente associado o conceito de segurança que pode ser definido como estar livre de condições que possam causar danos ou perdas de equipamentos ou ainda lesões ou morte de seres humanos (DHILLON; FASHANDI, 1997).

Em geral, as falhas nos robôs podem ser classificadas em três categorias: Falhas devido a mau funcionamento estrutural, o que advém das características dos materiais utilizados e suas condições de trabalho; Falhas devido a mau funcionamento da tecnologia, que são ocasionadas por falhas aleatórias de componentes, falhas sistemáticas de hardware e erros de software e Falhas comportamentais oriundas de decisões humanas durante a operação ou manutenção. Quando decisões são tomadas pelo próprio robô em operação,

como é o caso de robôs autônomos, a falha passa a ser da tecnologia e representa um erro de software.

Qualquer decisão que tomamos é baseada no nosso conhecimento presente sobre uma situação ao nosso alcance. Esse conhecimento vem, em parte, de nossas experiências diretas com situações relevantes ou de experiências relacionadas a situações semelhantes. Nosso conhecimento, portanto, pode ser aumentado por exemplo através de testes e análises apropriados, i.e. experimentação (U.S. NUCLEAR REGULATORY COMMISSION, 1981).

A existência da restrição de tempo nos leva a fazer uma distinção entre decisões boas e decisões corretas uma vez que para classificar decisões como boas ou ruins nos utilizados do retrospecto, i.e. dos fatos registrados. Se, a partir das informações de que o futuro de uma determinada empresa é promissor, investimos nas ações desta empresa, os futuros lucros recebidos ou os prejuízos dirão se a decisão original foi boa ou ruim, embora de acordo com a perspectiva da empresa naquele momento esta tivesse sido uma decisão correta. Portanto, nos concentramos em tomar as decisões corretas e para isso exige-se:

- A identificação das informações que serão pertinentes à antecipação da decisão;
- Um método para a aquisição das informações pertinentes;
- Uma consultoria racional ou análise dos dados adquiridos.

A análise é um processo direto de aquisição de dados ordenado e temporal e investigação de informações específicas de um sistema pertinentes a uma dada decisão. A partir disto, assume-se que a principal função da análise de sistemas é a aquisição de informações e não a geração de um modelo de um sistema. Portanto, há dois métodos analíticos genéricos através dos quais se toma decisões: os indutivos e os dedutivos.

A abordagem indutiva se apóia nos casos individuais para chegar a uma conclusão generalizada. Significa que, considerando um sistema, a partir de uma falha específica ou condição inicial, tentamos verificar o efeito dessa falha ou condição no funcionamento do mesmo e dessa forma faz-se uma análise indutiva. Muitas abordagens podem ser consideradas indutivas e alguns exemplos delas são: Análise Preliminar de Riscos (*Preliminary Hazards Analysis* - PHA), a anteriormente citada FMEA, a Análise de Modos de Falhas, Efeitos e Criticidade (*Failure Mode Effect and Criticality Analysis* - FMECA), Análise de Riscos de Falhas (*Fault Hazard Analysis* - FHA) e Análise de Árvore de Eventos (*Event Tree Analysis*) (U.S. NUCLEAR REGULATORY COMMISSION, 1981).

A abordagem dedutiva, por sua vez, se utiliza das generalizações para chegar a uma análise específica. Isto quer dizer que se um sistema apresenta uma falha qualquer, procura-se quais modos do comportamento do sistema ou de seus componentes contribuíram para aquela falha. A análise de árvores de falhas (FTA) é um exemplo de análise dedutiva.

Nesta técnica algum estado específico do sistema, que geralmente é um estado de falha, é estabelecido, e falhas mais básicas são encadeadas de forma sistemática levando a este evento indesejado.

Em resumo, os métodos indutivos são aplicados para determinar quais estados (geralmente de falhas) são possíveis e os métodos dedutivos são aplicados para determinar como um dado estado de um sistema (geralmente de falha) pode ocorrer.

2.4.1 Análise de Modos de Falhas e Efeitos (FMEA)

FMEA é uma abordagem qualitativa formal e sistemática para identificar potenciais modos de falhas de um sistema, suas causas e efeitos na ocorrência destas durante a operação de um sistema. Com isso, promove a base para identificar falhas potenciais e inadmissíveis, sendo inclusive fonte para o estabelecimento de requisitos de projeto e eventualmente *redesign*.

Uma grande desvantagem do FMEA é a sua análise de falha individualizada, o que significa que esta técnica não é adequada para combinar efeitos de duas ou mais falhas que venham a ocorrer ao mesmo tempo. Para tais casos, recomenda-se a técnica de Markov pois considera falhas de componentes não-independentes.

O grande perigo dos métodos indutivos é que se o estudo se tornar mero exercício de preenchimento de formulários ao invés de uma análise apropriada, o exercício será completamente fútil. Além disso, se um sistema é complexo, se torna uma tarefa difícil para um único analista imaginar sozinho e conduzir uma correta investigação de todas as falhas e seus efeitos num sistema. No FMEA e suas variações pode-se identificar, com certa razão, aquelas falhas em componentes cujo efeitos são "não-críticos", mas o número de possíveis modos de falhas pode ser considerado limitado. Sendo conservador, modos de falhas não especificados e efeitos questionáveis são classificados como críticos. Os objetivos desta análise são identificar modos de falhas individuais e quantificá-los.

2.4.2 Diagrama de Blocos

A análise por diagrama de blocos é um método cujo primeiro passo é identificar os modos de falhas e coletar as informações de confiabilidade de cada um dos componentes de um sistema. Dessa forma, blocos aos quais são atribuídas as probabilidades de sucesso ou falha são conectados para formar uma rede de confiabilidade que representa a dependência entre os componentes do sistema (DHILLON; FASHANDI, 1997). Estes componentes, portanto, podem estar em série (não-redundantes) ou em paralelo (redundantes). No primeiro caso, a falha de um único componente faz com que o sistema falhe, enquanto que no segundo a falha de um ainda permite que o sistema funcione através dos componentes do caminho redundante. Para que o sistema falhe quando há caminhos de blocos em

paralelo é preciso que todos os caminho sofram uma falha em pelo menos um componente.

2.4.3 Modelos Combinacionais

São combinações de árvores de falhas e diagramas de blocos. No entanto, sabe-se que às vezes se torna difícil ou até impossível estabelecer relações entre todas as falhas, além disso gerar modelos muito complexos. Se um robô possui muitas condições de falhas, FTAs separadas devem ser construídas para cada uma delas.

2.4.4 Análise de Árvore de Falhas (FTA)

FTA é uma análise de falhas dedutiva que se concentra em um evento indesejado específico e fornece um método para determinar as causas deste evento. O evento indesejado configura o evento de topo do diagrama de árvore de falhas construído para o sistema e geralmente é uma falha crítica/catastrófica. Trata-se um método sistemático para adquirir informações de um sistema que podem ser usadas para tomadas de decisões.

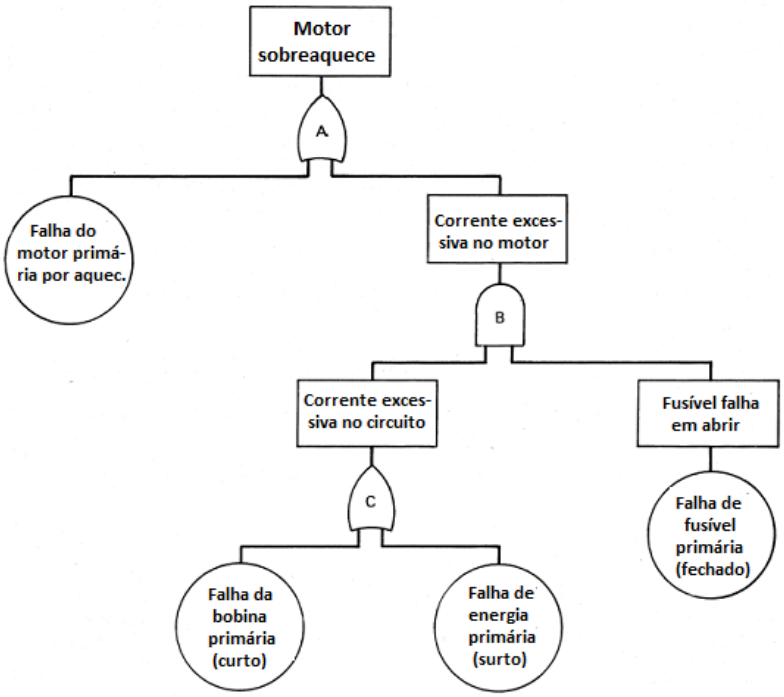
A árvore de falhas é um modelo gráfico de várias combinações paralelas ou sequenciais de falhas que resultarão no evento indesejado. As falhas podem ser associadas a falhas de componentes de hardware, falhas humanas ou qualquer outro evento pertinente que possa levar à falha principal, no topo do diagrama. A árvore de falhas então é composta de interrelações lógicas dos eventos básicos que levam ao evento indesejado através de *gates* que, tal como em um circuito lógico, permitem ou inibem a passagem de um sinal (neste caso, a falha) até a saída (o topo da árvore) (U.S. NUCLEAR REGULATORY COMMISSION, 1981). Um exemplo de FTA pode ser visto na figura 5.

É importante frisar que a FTA não é um modelo de todas as possíveis falhas do sistema ou todas as possíveis causas para a falha do sistema, é uma análise de um modo de falha específico e que inclui as falhas básicas que contribuem para esta falha principal. Além disso, não é exaustivo e está sempre sujeito às crenças consideradas mais plausíveis pelo analista. Há 4 tipos de eventos primários para construção de uma árvore de falhas (FTA), listados a seguir:

- Evento básico - um **círculo**;
- Evento condicional - uma **elipse** que representa uma condição/restrição à qual em geral se aplica um *gate* do tipo PRIORITY AND ou INHIBIT;
- Evento não-desenvolvido - **losango**;
- Evento externo - **pentágono/casa**.

Há também 5 tipos de *gates* (portas lógicas) que são utilizados para interligar os citados eventos básicos.

Figura 5 – Exemplo de FTA (Adaptado).



Fonte: Berman (2013)

- AND, OR, XOR;
- PRIORITY AND - a falha ocorre se todas as entradas ocorrerem numa sequência específica sendo esta representada por um evento condicional desenhado à direito do *gate*;
- INHIBIT - a falha ocorre se a entrada (única) ocorrer na presença de um **evento condicional** que a habilita;
- TRANSFER IN - indica que a árvore continua em um TF OUT correspondente;
- TRANSFER OUT - indica que essa porção da árvore deve ser anexada ao TF IN correspondente.

Os dois últimos são tipos de símbolos de transferência e, embora a simbologia não limite aos itens mencionados, estes são os símbolos mais utilizados.

2.5 Eletrônica Embarcada

Entende-se por sistema embarcado um conjunto de periféricos e microcontroladores integrados em uma mesma plataforma a fim de oferecer uma determinada solução de

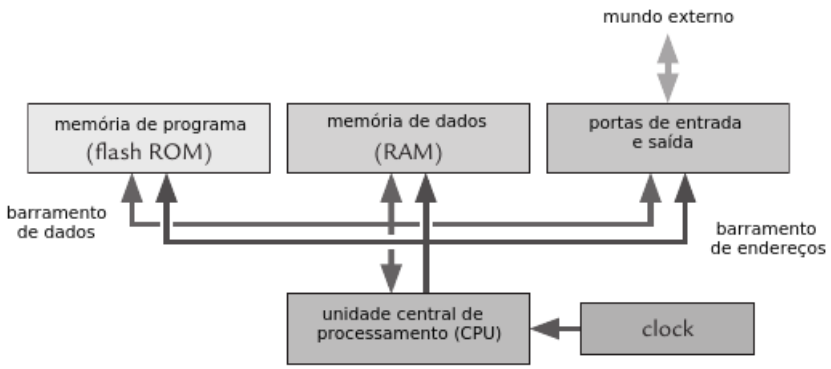
engenharia. A medida que os microprocessadores tem se tornado reduzidos e baratos, cada vez mais produtos tem se utilizado destes componentes embarcados para torná-los um tanto inteligentes, i.e. "*smart*". Diz-se que um sistema embarcado é todo e qualquer sistema computacional embutido em produtos como relógios digitais, elevadores, reprodutores de mídia, termostatos, equipamentos de controle industrial, instrumentos científicos e médicos, entre outros. Os *softwares* desenvolvidos para sistemas embarcados precisam lidar com questões que transcendem as aplicações de *software* voltadas para computadores pessoais e *mainframes*. Esses sistemas embarcados muitas vezes precisam realizar inúmeras tarefas de uma única vez. Além disso, precisam responder a eventos externos como, por exemplo, quando alguém aperta o botão de um elevador. Precisam estar preparados para todas as situações inesperadas sem que haja necessidade de intervenção humana e, ainda, estão sujeitos à interrupção de seu funcionamento caso seja previamente definido (SIMON, 2005).

O principal componente de um sistema embarcado é o microcontrolador (μC). Ainda que outros dispositivos sejam de fundamental importância para os objetivos de determinada aplicação, é no microcontrolador que se concentram todas as ações de controle e fluxo de dados do sistema, configurações e inicializações. Além disso, havendo mais de um microcontrolador no sistema, ao menos um deles atuará como mestre (*master*) coordenando os demais que serão seus escravos (*slaves*), formando uma rede, e que desempenharão funções específicas como processamentos dedicados a certos periféricos ou mesmo o papel de mestre em caráter temporário numa possível falha do principal em sistemas de alta criticidade.

Os microcontroladores são dispositivos eletrônicos compostos por uma unidade central de processamento (CPU), osciladores (*clock*), memória de dados e de programa, barramentos de dados e de endereço, portas de entrada e de saída conforme a figura 6 (DAVIES, 2008). Tais componentes são associados aos mais diversos módulos funcionais que variam conforme a família e modelo, mas que vão desde a simples portas digitais (I/Os), temporizadores, comparadores, conversores analógico-digitais, a padrões mais sofisticados de comunicação serial tais como UART (*Universal Asynchronous Receiver Transmitter*), I^2C (*inter-integrated circuit*), SPI (*serial peripheral interface*) e o popular USB (*Universal Serial Bus*). Em geral essas funcionalidades não são todas utilizadas de uma só vez e envolve uma questão de flexibilidade do *hardware* oferecida pelos fabricantes ao desenvolvedores para dar-lhes a chance de realizar mudanças de projeto ou simplesmente a possibilidade de reutilização do circuito integrado em uma aplicação completamente diferente. Com isso, os módulos sempre precisam ser habilitados via *software* pois fisicamente compartilham os terminais de um mesmo encapsulamento, o que permite, por exemplo, a redução do custo por unidade e poder de miniaturização do dispositivo.

A utilização de microcontroladores requer basicamente o conhecimento e manipula-

Figura 6 – Diagrama de blocos de um microcontrolador (Adaptado).



Fonte: Davies (2008).

ção de seus registradores e *flags* (indicadores de estado), o que deve ser obtido através das folhas de dados (*datasheets*) e manuais do dispositivo oferecidos pelo fabricante, além da exigência de habilidade em lógica e programação. Embora inicialmente eles só pudessem ser programados em linguagem de baixo nível (i.e. *Assembly*) hoje já é possível fazer uso de linguagem orientada a objeto como o C++, além de existirem *softwares* denominados *Integrated Development Environment* (IDE) que facilitam o trabalho de desenvolvimento principalmente no que diz respeito à depuração (*debugging*) do algoritmo que pode ser feita *online*.

O desenvolvimento do software, que neste caso consiste de um *firmware* (i.e. um programa gravado em memória não-volátil que não pode ser modificado pelo usuário, só pelo desenvolvedor), compreende as mesmas características básicas de um programa comum. Dessa forma, funções são implementadas e bibliotecas adicionadas; porém a forte interação com o hardware permite o uso de recursos como o controle de consumo de energia ou uso de interrupções, que segundo Davies são como funções, mas com a distinção crítica de serem requisitadas pelo hardware com certa prioridade em momentos imprevistos, em detrimento das funções acionadas via software. Ou seja, quando uma interrupção ocorre, o programa em execução para no ponto em que se encontra, salva o endereço de memória que indica o ponto do algoritmo em que foi interrompido, executa a rotina associada à interrupção acionada, o que ao final inclui a limpeza do *flag* (indicador) da interrupção em questão e retorna para o ponto do algoritmo em que estava para dar continuidade à sua execução normal.

2.5.1 Comunicação Serial

Quando se aumenta a complexidade dos dados que entram e saem de um dispositivo, portas I/O deixam de ser soluções práticas e não justificam sua utilização. Os módulos

de comunicação são, em essência, portas I/O modificadas para atender a padrões de comunicação criados para serem capazes de lidar com tráfego de dados em grandes quantidades de forma ordenada, controlada e com baixa incidência de erros.

Pequenos microcontroladores podem precisar armazenar dados em memórias externas, o que requer uma comunicação através de milímetros de trilhas de circuito impresso, por exemplo. Da mesma forma, informações também podem precisar ser descarregadas de um instrumento registrador conectando-o a um computador. No passado, o mais comum era que essa comunicação fosse por UART, padrão RS-232, mas recentemente esse tipo de comunicação tem sido substituída gradativamente pelo padrão USB, mais fácil para o usuário final porém mais complexo em termos de hardware e software, já que requer o desenvolvimento de *drivers* para comunicação com sistemas operacionais, por exemplo. Comunicação *wireless* também tem ganhado importância com o crescimento das redes *ZigBee* (TECHNAVIO, 2016) e outros sistemas de baixo consumo de energia. Esses tipos de comunicação são ditos seriais por consistirem na transferência de bits um a um de forma sequencial ao longo do tempo, embora também exista a comunicação paralela, pouquíssimo utilizada nos dias atuais depois dos avanços e diversidade de padrões de comunicação serial.

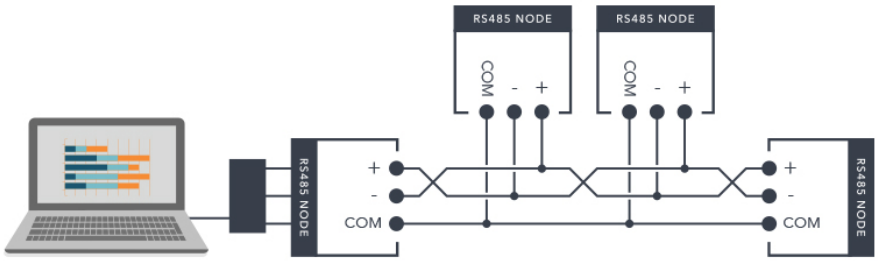
Os diferentes padrões de interface digital levam em consideração os níveis de sinal, o tipo de lógica (positiva ou negativa), a orientação por bit ou byte, o tipo de linha de transmissão a ser usada; a forma de transmissão, i.e. se balanceada ou não, com terminação ou não, unidirecional ou bidirecional, multiplex ou simplex; a imunidade a ruído; os tipos de conectores usados e a velocidade de transmissão (BRAGA, 2016). No caso dos padrões RS que são padrões de aplicação industrial publicados pela EIA - *Electronic Industries Association*, os mais conhecidos são RS-232, RS-422 e RS-485.

O padrão RS-232 é o mais antigo e mais conhecido padrão. É indicado para linhas curtas não balanceadas, ponto a ponto. As principais especificações deste padrão são operação lógica positiva com tensões de $\pm 5V$ à $\pm 15V$; comprimento máximo de cabo recomendado de 15 m e taxa de transmissão de até 160 kbps. Os outros dois padrões vieram mais tarde como melhorias do RS-232 (BRAGA, 2016).

O padrão RS-422 é indicado para linhas balanceadas (diferenciais) e se caracteriza por transmissão de dados unidirecionais em transmissões com ou sem terminação. A velocidade de transmissão chega aos 10 Mbps com linhas de até pouco mais de 1200 m. O envio dos sinais é feito geralmente em *half-duplex*, com 2 fios além da referência. É útil para aplicações multiponto onde apenas um transmissor é conectado a um barramento de até 10 receptores (NATIONAL INSTRUMENTS, 2016). *Half-duplex* é um tipo de comunicação serial onde o transmissor (TX) e o receptor (RX) não podem ser usados ao mesmo tempo. Normalmente este método é utilizado quando muitos dispositivos estão conectados a um único barramento, como é o caso de muitas redes RS-485. O padrão

RS-485 é uma melhoria do RS-422 que permite até 32 transmissores e até 32 receptores (fig. 7). Qualquer dos dispositivos escravos em um barramento RS-485 pode se comunicar com quaisquer outros escravos sem atravessar um dispositivo mestre. Além disso, todo dispositivo RS-422 pode ser controlado pelo RS-485.

Figura 7 – Rede serial cabeada RS-485.



Fonte: Accuenergy Inc. (2016).

Enquanto as comunicações SPI e I^2C costumam ser utilizadas entre microcontroladores e outros dispositivos na mesma PCI, a comunicação assíncrona é comumente utilizada para troca de dados entre equipamentos como o sistema embarcado e um computador pessoal. SPI e I^2C são síncronas, o que significa que o sinal de *clock* é enviado com os dados existindo pelo menos um mestre e um escravo na comunicação. Embora transmissão e recepção sejam processos separados, eles podem ocorrer no mesmo meio, num único fio além da referência (*ground* - GND) - como ocorre no I^2C - e ao mesmo tempo, ao que se dá o nome de comunicação *full-duplex*. Já na *half-duplex*, os dados podem transitar em qualquer direção, porém sendo uma de cada vez (DAVIES, 2008). No SPI dois fios são usados permitindo que as informações sejam enviadas simultaneamente nas duas direções e um terceiro fio (além da referência) pode ser utilizado quando se quer selecionar um dispositivo escravo. Por isso a comunicação SPI pode ser encontrada a 3 ou 4 fios. A necessidade de uma ligação extra advém da própria característica da comunicação SPI não possuir controle de transmissão via software como ocorre na I^2C (i.e. envio de endereços e aceites/*acknowledgments*), o que a torna menos sofisticada mas mais simples, mais rápida e mais apropriada à transmissão de grandes volumes de dados.

2.5.2 Plataformas de Desenvolvimento

Apesar do grande destaque de plataformas (*kits*) de desenvolvimento da Intel como a Galileo Board, que é compatível com Arduino, e a Edison Board (figura 8), que é a

aposta da empresa para as tecnologias *wearables* devido a suas dimensões comparáveis as de um cartão de memória SD, tais plataformas se enquadram em um patamar acima dos sistemas embarcados microprocessados já que são dotadas de processadores poderosos (arquitetura x86, Pentium) e são capazes de executar sistemas operacionais Linux (INTEL CORP., 2016). Portanto, há de se considerar que essas tecnologias devem ser direcionadas a aplicações mais complexas, em geral desenvolvidas em linguagens de alto nível e que exigem maior poder de processamento. Consequentemente, possuem maior custo e consomem mais energia, embora sejam dispositivos de dimensões compactas e performance otimizada se comparados a outros computadores.

Figura 8 – Intel Edison.

Fonte: Intel Corp. (2016).

No âmbito dos sistemas microprocessados é inegável a importância das plataformas Arduino (atualmente vendida como Genuino, fora dos EUA) e suas variações que se popularizaram pela proposta inovadora de *open source hardware* (ARDUINO, 2016), ganhando adeptos rapidamente e fortalecendo uma comunidade de desenvolvedores por todo o mundo. A proposta do Arduino é favorecer o acesso à prototipagem eletrônica àqueles que não estão familiarizados com desenvolvimento de hardware ou mesmo de software. Para isso, uma IDE própria é oferecida para que os algoritmos sejam desenvolvidos na linguagem intuitiva e própria do Arduino, considerada de alto nível e baseada em C/C++. Em virtude do vasto número de bibliotecas, que incluem funções de comunicação, controle, armazenamento, conectividade, processamento de sinais, entre outras oferecidas pela plataforma no intuito de atender a um maior número de usuários bem como às suas placas de expansão (*shields*), os programas desenvolvidos pelo usuário final acabam sendo menos otimizados já que essas bibliotecas são construídas e mantidas para serem o mais

genéricas possível. Outro ponto é o acesso à arquitetura do microprocessador do Arduino que não é direto e que pode exigir um esforço de codificação e conhecimentos extras para alterar a programação padrão.

Portanto, em aplicações de baixo custo e baixo consumo de energia, que exigem a execução de tarefas de pouca a média complexidade, com criticidades variadas, pode-se recorrer aos sistemas embarcados microprocessados como os oferecidos pelas empresas fabricantes de semicondutores. Nestes casos o desenvolvedor compreende as características da arquitetura do microprocessador a ponto de trabalhar diretamente com os registradores, podendo criar algoritmos em linguagem C/C++ ou Assembly. Essas aplicações podem ser complementares, mas não menos importantes, ao sistema de um robô, como por exemplo um subsistema de emergência ou um subsistema de roteamento de informações.

A Texas Instruments (TI) é um fabricante de semicondutores que produz chips analógicos e embarcados para dispositivos móveis, eletrônicos, carros, equipamentos industriais e infraestrutura *wireless*, dentre outros dispositivos semicondutores. Possui kits de desenvolvimento conhecidos por *LaunchPads* que estão disponíveis em uma variedade de especificações e capacidades de *hardware* para atender às necessidades de cada projetista a preços acessíveis que variam atualmente de 10 a 25 dólares. São plataformas embarcadas que utilizam microcontroladores das quatro famílias da Texas: MSP, C2000™, Tiva™ C Series ARM® e Hercules (TEXAS INSTRUMENTS, 2016b). Merece destaque também a Microchip Technology, empresa fabricante de semicondutores responsável pelas populares famílias de microcontroladores PIC® e dsPIC® cujas versões variam entre 8-bit, 16-bit e 32-bit (MICROCHIP TECHNOLOGY INC., 2016).

Um kit *LauchPad* pode ter suas funcionalidades expandidas através dos chamados *BoosterPacks*, módulos fabricados pela TI, outras empresas especializadas ou pela própria comunidade de desenvolvedores com compatibilidade física e eletrônica que podem ser montados sobre a plataforma principal adicionando recursos para determinadas aplicações antes insuficientes ou inexistentes na plataforma original. Proposta muito semelhante aos *shields* da plataforma Arduino.

3 Implementação do Sistema

3.1 Introdução

Neste capítulo serão detalhadas as etapas de desenvolvimento do trabalho de acordo com os objetivos apontados na seção 1.2.

3.2 Especificação de Requisitos e Componentes

Tendo em vista que toda a motorização dos sistemas de tração, apoio e busca do robô de inspeção em linhas de alta tensão - PI-Ro - é composta por servomotores *dynamixel* da ROBOTIS, série MX, o dispositivo-alvo de monitoramento escolhido neste trabalho foi o modelo MX-28R com torque máximo de 2,5 N.m e velocidade sem carga de até 55rpm, utilizando alimentação de 12V (ROBOTIS INC., 2010a). Este modelo possui dimensões 35,6 x 50,6 x 35,5mm, pesa 72g e pode ser visto na figura 9. No robô para inspeção em linhas de alta tensão, este modelo de servomotor é utilizado em conjunto com o modelo MX-106R e os mesmos são distribuídos ao longo das juntas conforme a exigência de maior ou menor torque. Dessa forma, promovem toda a movimentação do sistema.

Figura 9 – Dynamixel MX-28R.

Fonte: ROBOTIS Inc. (2010a)

A série MX compreende servomotores com capacidades de torque máximo que variam de 2,5 a 8,4 N.m, com velocidades máximas sem carga que variam de 45 a 63 rpm sob a tensão nominal, i.e. 12V. Podem efetuar comunicação TTL (UART) ou RS-485, o que lhes atribui o sufixo T ou R, respectivamente. Os dispositivos desta série são equipados com um processador 32-bit de 72MHz Cortex M3, *encoder* magnético sem contato e taxas de transmissão de até 4,5Mbps usando barramento RS-485 (ROBOTIS INC., 2010a). Um

algoritmo de controle PID é usado para assegurar a posição do eixo que, por sua vez, pode ser ajustada individualmente para cada servo, permitindo controlar a velocidade e a força com que o motor responde. Além das informações de posição, velocidade e ângulo, é possível obter informações de temperatura, ângulo, posição, velocidade, torque, carga e tensão elétrica através dos sensores internos dos servomotores.

A fim de realizar o monitoramento dos parâmetros de engenharia de interesse e atualizar o banco de dados de registro de falhas destinado ao subsistema de confiabilidade do robô, é necessário definir uma plataforma eletrônica de interfaceamento e tratamento de dados para estabelecer a comunicação com os atuadores - neste caso, os servomotores *Dynamixel* - e o sistema de gerenciamento de dados pertencente ao robô PI-Ro.

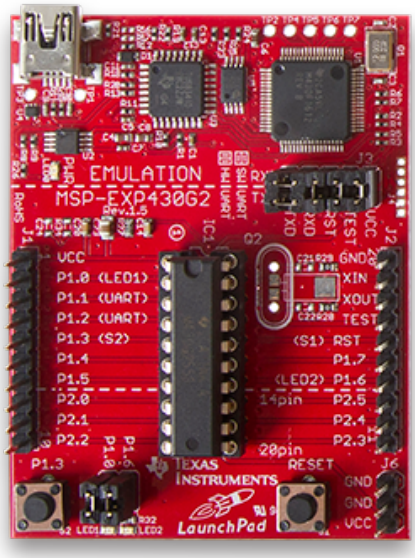
Portanto, com base no exposto na seção 2.5.2, para realizar a interface serial com o servomotor e interface com outros dispositivos, desenvolvendo o *firmware* para a instrumentação, propõe-se o uso de uma plataforma de desenvolvimento de sistema embarcado de teste, conforme figura 10, baseada em um microcontrolador da Texas Instruments da família MSP430, a MSP-EXP430G2 *LaunchPad*.

Trata-se da versão de menor custo das plataformas *LaunchPad* (USD 9,99), ressaltando-se que devido a sua capacidade de emulação *on-board* não requer a utilização de gravadores externos para programação e depuração, o que representa um custo financeiro de desenvolvimento eliminado. A placa LaunchPad escolhida para este trabalho possui o modelo de microcontrolador **MSP430G2553** e as seguintes características:

- Emulação *on-board* - o que dispensa o uso de gravadores externos;
- Soquete DIP de 20 pinos para prototipagem;
- Baixo consumo de energia;
- Processador de 16MHz;
- Memórias flash 16KB e RAM de 512B;
- 2 temporizadores (*timers*) 16-bit;
- Comparador e ADC 10-bit;
- Conexões UART, SPI e I^2C.

A proposta de uso desta plataforma de testes, cujo diagrama pode ser visto na figura 11, tem a finalidade de validar o método de monitoramento para, somente depois disto, conceber-se um projeto eletrônico de sistema embarcado personalizado para o robô PI-Ro, contemplando um microcontrolador da mesma família testada e incluindo montante de memória suficiente para a abrangência do monitoramento de todas as juntas, pelo

Figura 10 – MSP-EXP430G2 *LaunchPad*.



Fonte: Texas Instruments (2016a)

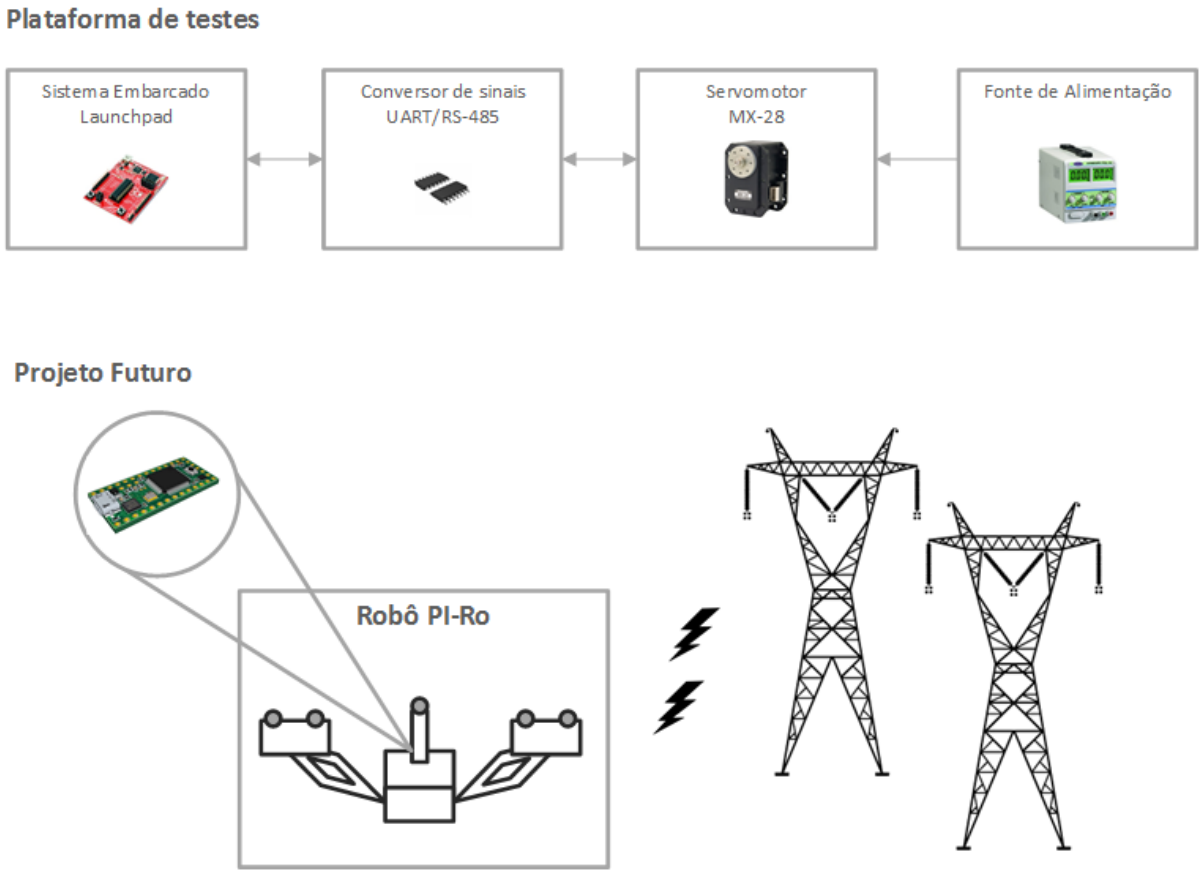
tempo máximo possível das operações pretendidas. Ressalta-se que plataforma definitiva será instalada no robô em seu compartimento blindado eletromagneticamente, assegurando um ambiente de operação sem interferências ao sistema de aquisição.

3.3 Montagem do *hardware*

Considerando a inexistência de conectividade direta entre a *LaunchPad* e o servomotor MX-28R, foi projetada e montada uma placa eletrônica de expansão (sob a filosofia dos *BoosterPacks*) para compatibilizar a conexão UART pertencente ao microcontrolador MSP430 e o padrão RS-485 existente no servomotor *dynamixel*. Para isso foi utilizado um circuito integrado conversor **ADM3491**. Trata-se de um CI compatível com os padrões RS-485/RS-422 com capacidade de transmissão *half-duplex* e *full-duplex* e alimentação 3,3V, interoperável com 5,0V. Possui baixo consumo de energia elétrica e alcança taxas de transferência de até 10Mbps. Em conformidade com o padrão RS-485, permite que até 32 dispositivos estejam conectados ao barramento.

Na rede RS-485, os dados são transmitidos de forma diferencial (i.e. independente da referência) em um par de fios fisicamente trançados (*twisted pair*) para minimizar as interferências eletromagnéticas (EMI). Na figura 12 o ruído é gerado pelos campos magnéticos (azul) do ambiente que resultam nas correntes induzidas (preto). Nos fios em ligação direta todo o ruído flui na mesma direção criando uma corrente em loop como se fosse um transformador comum. Quando o par de fios é trançado, as correntes induzidas se

Figura 11 – Diagrama da plataforma de testes e proposta futura.

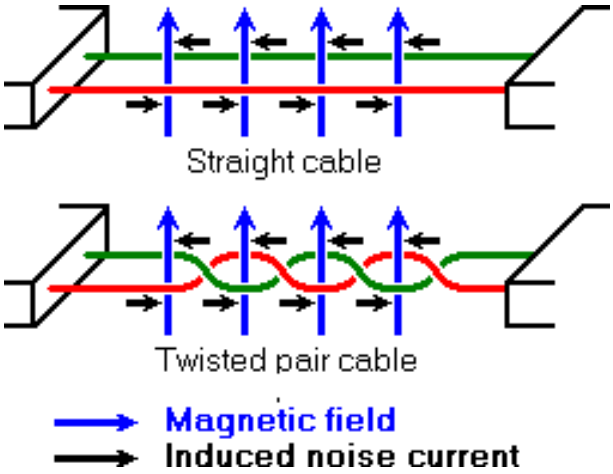


Fonte: Própria.

encontram em direções opostas umas às outras, reduzindo drasticamente o ruído na rede (BIES, 2015). Essa condição confere à transmissão RS-485 grande imunidade a ruídos e alcance de grandes distâncias (geralmente até 1200m). Pode ser feita também a 4 fios. No modo mais usual, a 2 fios, o transmissor e o receptor de cada dispositivo são conectados a um par trançado. A 4 fios, a rede possui um mestre cujo transmissor é conectado a cada receptor dos dispositivos escravos através de um par trançado. Os transmissores dos escravos, por sua vez, são todos conectados ao receptor do dispositivo mestre também através de um par trançado (B&B ELECTRONICS, 2016).

Redes a 2 fios tem a vantagem do menor custo com cabeamento e a habilidade dos nós se comunicarem entre si. Por outro lado, esse modo é limitado à configuração *half-duplex* e requer atenção no que diz respeito às temporizações na troca de pacotes. A 4 fios, pode-se criar uma rede *full-duplex*, no entanto, a rede fica limitada a configuração mestre-escravo (i.e. um mestre solicita informações de cada escravo individualmente e os escravos não se comunicam entre si). Embora o padrão RS-485 possa operar tanto em *half-duplex* como *full-duplex*, o servomotor MX-28R possui apenas duas conexões de sinal,

Figura 12 – Imunidade à ruído em pares de fio trançados.

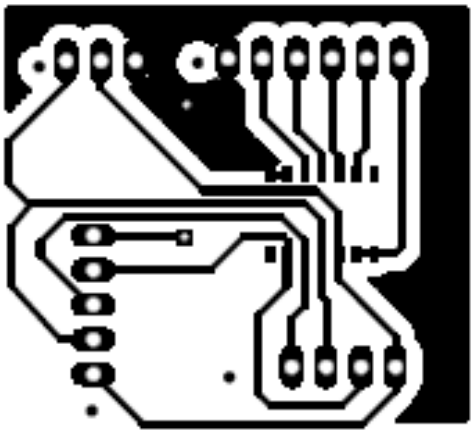


Fonte: Bies (2015).

sendo compatível apenas com a transmissão *half-duplex* que, portanto, é a utilizada neste trabalho.

Portanto, a placa de circuito impresso (PCI) do conversor ADM3491, que pode ser vista nas figuras 13 e 14, foi confeccionada e montada de forma genérica com flexibilidade entre os modos *half-duplex* ou *full-duplex*, permitindo utilizá-la tanto nesta quanto em outras aplicações, explorando ao máximo as capacidades do circuito integrado escolhido para conversão.

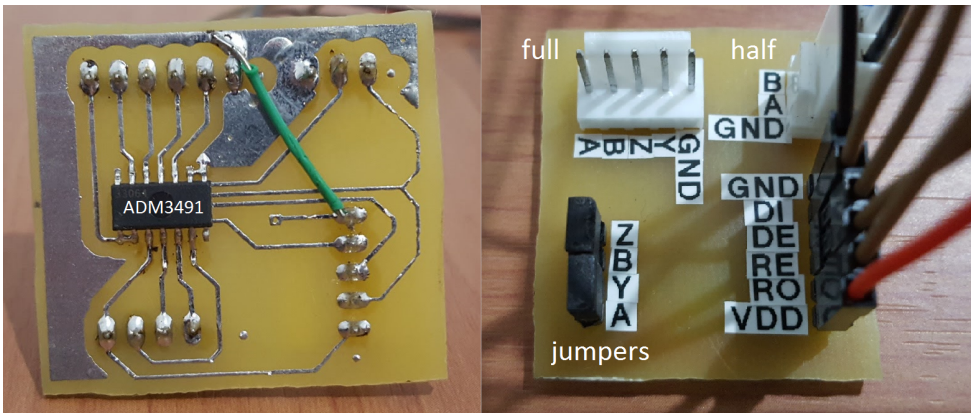
Figura 13 – *Layout* da PCI do conversor UART/RS-422/RS-485.



Fonte: Própria.

Como pode ser visto no diagrama de blocos funcional da figura 15, o CI ADM3491 possui as entradas A e $\overline{\text{B}}$ e as saídas Y e $\overline{\text{Z}}$ com as quais é possível realizar a comunicação *full-duplex*. Na aplicação proposta neste trabalho, portanto *half-duplex*, é necessário realizar um

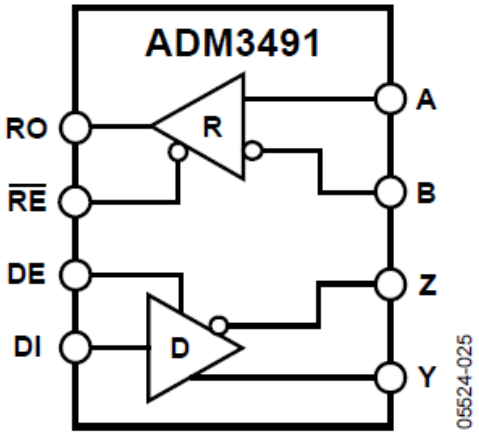
Figura 14 – Placa de expansão do conversor ADM3491.



Fonte: Própria.

curto-circuito através de condutores *jumper* entre os terminais A e Y / $\bar{\text{B}}$ e $\bar{\text{Z}}$ configurando o CI como unidirecional. O terminal DI (*Driver Input*) corresponde ao terminal ao qual é conectado o trasmissor (TX) da UART e o terminal RO (*Receiver Output*), onde é conectado o receptor (RX) da UART. Os terminais DE (*Driver Output Enable*) e $\bar{\text{R}}$E (*Receiver Output Enable*) respondem pela direção dos dados e existem de forma independente para atender ao modo de transmissão *full-duplex*. Para *half-duplex*, eles podem ser curto-circuitados já que transmissão e recepção ocorrem sempre em momentos distintos.

Figura 15 – Diagrama de blocos funcional do ADM3491.



Fonte: Analog Devices (2011).

3.4 O protocolo dxl 1.0

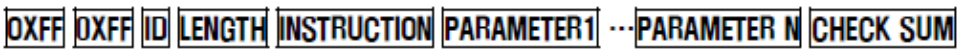
No protocolo dxl 1.0, cuja natureza dos dados é binária, existem dois tipos de pacotes: Instrução (*instruction packet*) e Status (*status packet*). O primeiro consiste de pacotes que podem ser enviados pelo controlador principal (neste caso, a Launchpad) a um servomotor *dynamixel*. O segundo é o tipo de pacote enviado pelo *dynamixel* em resposta ao controlador principal.

A comunicação estabelecida através desse protocolo é serial, assíncrona, 8-bit com 1 bit de parada (*stop bit*) e sem bit de paridade. Como dito anteriormente, é *half-duplex* e requer atenção nas temporizações tanto entre o envio de pacotes quanto no tempo de seleção da direção dos dados.

3.4.1 *Instruction Packet*

A estrutura do pacote do tipo **Instrução** (*Instruction Packet*) pode ser vista na figura 16. O pacote se inicia com um cabeçalho contendo dois envios do *byte* **0xFF**. A seguir vem o *byte* de ID que é o número de identificação (modificável) que cada servomotor possui. Com ele, o controlador principal pode identificar, comunicar-se e controlar um determinado servomotor mesmo que esteja conectado a outros num mesmo barramento. Este identificador requer atenção do programador pois, se dois ou mais *dynamixels* forem configurados com o mesmo ID e forem conectados à mesma rede, haverá colisão de pacotes e a rede enfrentará problemas. É possível atribuir IDs que variam de 0 a 253 (0x00 a 0xFD) (ROBOTIS INC., 2010b). Para a ação de *broadcasting*, i.e. o envio de uma instrução para todos os *dynamixels* no barramento ao mesmo tempo, utiliza-se o ID=254 (0xFE).

Figura 16 – Estrutura do pacote instrução.



Fonte: ROBOTIS Inc. (2010b).

O byte seguinte identificado como LENGTH corresponde ao tamanho do pacote. É calculado a partir do número de parâmetros da instrução (N) + 2. Já INSTRUCTION corresponde ao comando/instrução que se deseja enviar ao *dynamixel*. Pode ter parâmetros ou não, conforme a figura 17. Dentre as opções, pode-se solicitar o *status* do servomotor através do comando PING (0x01), a ser detalhado na seção 3.4.2 - Pacote *Status*; realizar uma leitura em memória através do comando READ_DATA (0x02), como por exemplo, a tensão elétrica atual ou realizar uma escrita em memória através do comando WRITE_DATA (0x03), como por exemplo, alterar o ID do servomotor.

Figura 17 – Valores correspondentes a cada instrução (Adaptado).

Valor	Nome	Função	N° de parâmetros
0x01	PING	Sem execução. É usado quando o controlador está pronto para receber um Status Packet.	0
0x02	READ_DATA	Este comando lê dados do dynamixel.	2
0x03	WRITE_DATA	Este comando escreve dados no dynamixel.	2 ou mais
0x04	REG WRITE	Similar ao WRITE_DATA mas fica em standby sem ser executado até que um comando de ação seja disparado.	2 ou mais
0x05	ACTION	Esse comando dispara ações registrada pelo REG WRITE.	0
0x06	RESET	Este comando restaura o dynamixel às configurações de fábrica.	0
0x83	SYNC WRITE	Comando utilizado para controlar vários dynamixels ao mesmo tempo.	4 ou mais

Fonte: ROBOTIS Inc. (2010b).

PARAMETER 1..N correspondem aos *bytes* de parâmetros que contém informações adicionais exigidas por uma determinada instrução. No caso da função PING, não são necessários parâmetros; Já na função READ_DATA são requeridos dois parâmetros: o endereço de memória inicial do dado a ser lido (PARAMETER 1) e o tamanho do dado a ser lido (PARAMETER 2) (ROBOTIS INC., 2010c). Todos os endereços de memória EEPROM e RAM podem ser consultados na Tabela de Controle (*Control Table*) do manual eletrônico do fabricante e indicam as alocações para todo tipo de informação fornecida pelo servomotor (ROBOTIS INC., 2010a).

Por fim, insere-se o *checksum*. Trata-se de um método utilizado para verificar a integridade do pacote, ou seja, atestar se os dados transmitidos foram danificados durante a comunicação. O *checksum* é calculado de acordo com a seguinte expressão:

$$checksum = \sim (ID + Length + Instruction + Parameter1 + \ldots ParameterN)$$

Onde $\sim$ é o operador lógico NOT. Além disso, quando o resultado entre parêntesis for maior que 255 (0xFF), o cálculo deve considerar apenas o byte menos significativo. Portanto, somente a esse valor aplica-se o operação de inversão (NOT).

Para o correto funcionamento do protocolo, o atraso de envio de cada *byte* de um mesmo pacote não deve ser superior a 100 ms, caso contrário, o *dynamixel* reconhece isso como um erro de comunicação e aguarda pelo cabeçalho (0xFF 0xFF) de um novo pacote.

3.4.2 *Status Packet*

Exceto em modo *broadcasting*, são pacotes enviados periodicamente pelos dynamixels ou em resposta a um pacote de instrução que aguarde retorno. Também são chamados de pacotes de resposta (*return packet*).

3.5 O *firmware* para instrumentação

Para que o objetivo de monitoramento dos parâmetros de engenharia de interesse do servomotor para a confiabilidade do sistema seja alcançado, um algoritmo de comunicação e armazenamento de dados foi desenvolvido sob a forma de *firmware* na plataforma *Launchpad*. Este deverá ser integrado sob uma plataforma de sistema embarcado projetado para esta finalidade que comunicará diretamente com o sistema de gerenciamento do robô PI-Ro a fim de que seu acionamento seja coordenado com o funcionamento do robô durante as inspeções.

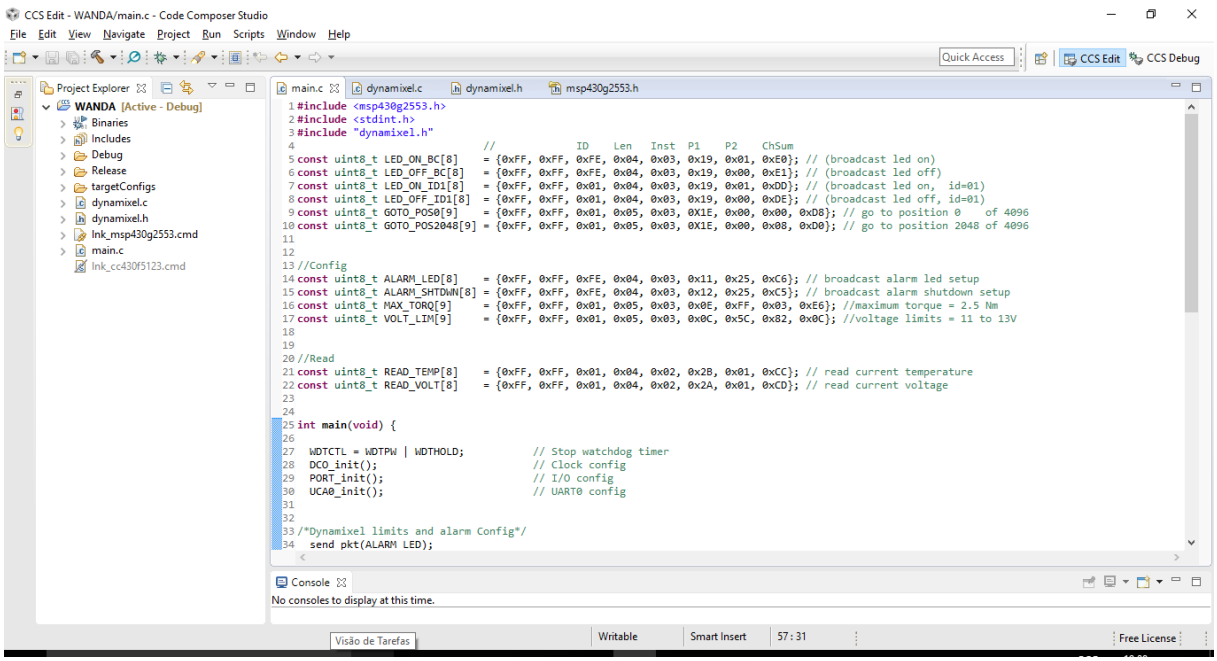
O *software* embarcado estabelece uma comunicação com o servomotor através de um barramento RS-485 e de um protocolo próprio e comum a algumas séries de servomotores do fabricante, o protocolo dxl 1.0 (ROBOTIS INC., 2010d). Com isso, adquire informações dos sensores internos tais como carga, tensão elétrica, temperatura e velocidade.

O *firmware* foi desenvolvido na IDE gratuita **Code Composer Studio** (CCS) v6.1.2.00015 oferecida pela Texas Instruments. Consiste de um projeto em linguagem C com um arquivo **main.c** e uma biblioteca desenvolvida para esta aplicação denominada **dynamixel**, composta de um arquivo de extensão C (*source file*) e um arquivo de extensão H (*header file*) (Figura 18) que podem ser vistos no **Apêndice A**.

Conforme ilustra a figura 19, no **main**, após serem inicializados os periféricos que dizem respeito à base de tempo do sistema (i.e. *clock* de 1MHz), configuração de portas de entradas e saídas (I/O) que serão conectadas à placa de expansão do conversor ADM3491 e configuração da comunicação serial (UART), são também configurados os alarmes disponíveis nos dynamixels assim como os limites de operação de torque e tensão elétrica.

Em seguida, é executado um loop infinito para que, enquanto o robô estiver em operação, o sistema de aquisição colete um *dataset* composto por carga, tensão e temperatura atuais. Esse conjunto de dados por fim é armazenado na memória flash para posterior transmissão ao subsistema de confiabilidade do robô. Embora sejam tratados apenas esses parâmetros de engenharia, o sistema de monitoramento pode ser estendido a medições de ângulo, posição angular, velocidade angular e aceleração angular.

Figura 18 – *Firmware* desenvolvido em CCS.



Fonte: Própria.

3.5.1 Configurações

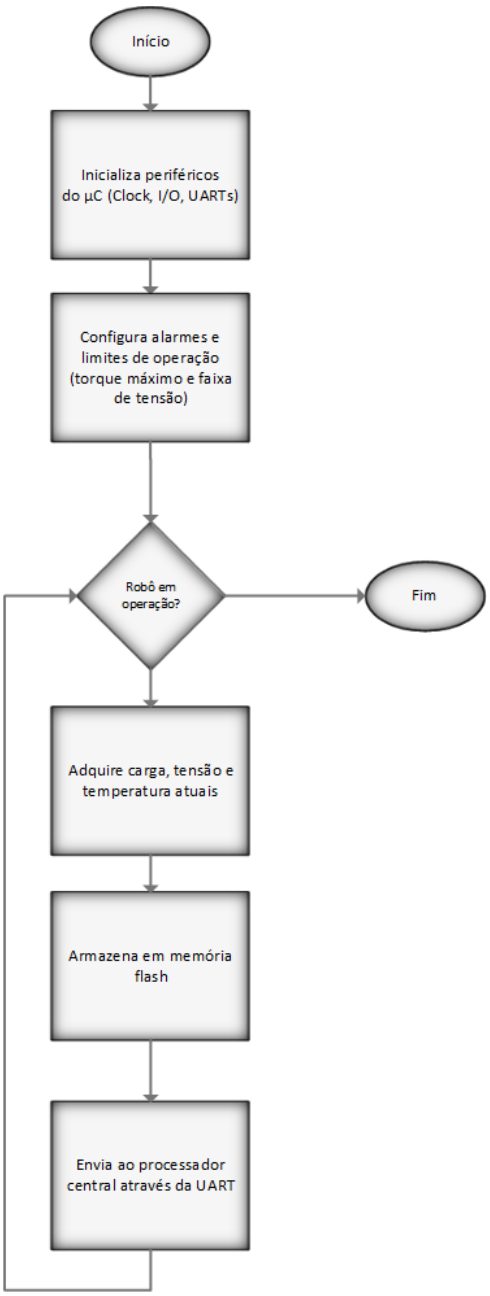
Os motores dynamixel tem a capacidade de gerar alarmes sob a forma de sinal luminoso piscante (*Alarm LED*) e sob a ação de desligamento do motor (*Alarm Shutdown*). Para configurá-los é necessário observar a tabela da figura 20. Por padrão, os alarmes do servomotor são configurado para atuar na ocorrência de sobrecarga (bit 5) e de sobreaquecimento (bit 2).

Neste trabalho, adotando uma abordagem mais conservadora e visando à confiabilidade, ambos os alarmes são reconfigurados pelo firmware da Launchpad para incluir o erro de tensão de entrada. Essa reconfiguração é feita através de uma operação de escrita por *instruction packet*, conforme o protocolo dxl 1.0, em modo broadcast para que futuramente todas as juntas do robô recebam a mesma instrução de alarme e permita o monitoramento.

Sobre a configuração de torque máximo admissível, para o dynamixel MX-28 tem-se 2,5Nm (12V) que corresponde ao valor 0x3FF conforme a Control Table (ROBOTIS INC., 2010a). Já a temperatura máxima vem de fábrica configurada para 80°C a qual não se recomenda alterar a não ser para valor inferior. Para fins de testes, portanto, o limite de temperatura será configurado em 50°C, temperatura que pode ser atingida em ambiente controlado a partir de uma fonte de calor externa.

Por fim, a faixa de tensão elétrica admissível para funcionamento padrão do servomotor é de 6 a 16V, mas, da mesma forma que os parâmetros anteriores, foi reconfigurada

Figura 19 – Fluxograma principal do *firmware*.



Fonte: Própria.

para um intervalo mais rigoroso de 11 a 13V para garantir o desempenho nominal de suas funções.

Todas essas decisões são preliminares e poderão ser alteradas conforme a evolução do desenvolvimento do sistema de confiabilidade do PI-Ro que levará em consideração ambiente e condições de operação do robô, vida útil dos componentes, regime de operação, entre outros fatores.

Figura 20 – Descrição do byte de erros do *status packet* (Adaptado).

Bit		
Bit 7	0	-
Bit 6	Erro de instrução	Quando uma instrução indefinida é enviada ou um comando de ação é disparado sem um comando REG_WRITE
Bit 5	Erro de sobrecarga	Quando a corrente de carga não pode ser controlada com máximo torque configurado
Bit 4	Erro de checksum	Quando o checksum do pacote de instrução transmitido é inválido
Bit 3	Erro de faixa de uso	Quando o comando é dado fora da faixa admíssivel do parâmetro
Bit 2	Erro de sobreaquecimento	Quando a temperatura interna está fora do intervalo de operação estabelecido na tabela de controle ou configurado
Bit 1	Erro de ângulo limite	Quando a posição desejada é escrita com um valor que não está entre os limites de ângulo horário e anti-horário
Bit 0	Erro de tensão	Quando uma tensão é aplicada fora do intervalo de operação estabelecido na tabela de controle ou configurado

Fonte: (ROBOTIS INC., 2010a).

3.5.2 Biblioteca *dynamixel*

A biblioteca é composta por três seções: a primeira responde pela inicialização dos periféricos necessários à comunicação com os servoatuadores dynamixel, a segunda pela implementação do protocolo de comunicação dxl 1.0 e a terceira pelas operações com a memória Flash.

3.5.2.1 Inicialização

Na primeira seção tem-se a função **PORT_init** que configura as portas I/O responsáveis pelos sinais de controle do conversor TTL-RS485, ou seja *Driver Input Enable* (DE) atribuído à porta P2.0 da LaunchPad e *Receiver Output Enable* (RE) atribuído à porta P2.1, ambas em modo *output* (P2DIR=0x03).

Configura também as portas referentes ao módulo de comunicação da UART0 (padrão TTL): receiver (UCA0RXD) e transmitter (UCA0TXD). Para que o módulo UART possa utilizar as portas P1.1 e P1.2 como RX e TX, respectivamente, é necessário fazer a seleção da função secundária através dos registradores de P1SEL e P1SEL2.

Ademais, a função **DCO_init** configura o clock interno do sistema, DCO (Digitally Controlled Oscillator) para a frequência de 1MHz enquanto a função **UCA0_init** configura os registradores necessários ao funcionamento da UART0 para comunicação com o dynamixel.

3.5.2.2 Funções Auxiliares

Já na segunda parte da biblioteca, antes de compreender a função principal do protocolo que é o envio de pacotes de instrução, é necessário definir algumas variáveis e funções auxiliares:

Para gerenciar o processo de transmissão e recepção dois tipos de vetores do tipo inteiro são utilizados: tx_data e rx_data. Este último tem tamanho fixo de 16 posições, tamanho suficiente para tratar qualquer pacote de status recebido em resposta a um comando enviado, especialmente os comandos de leitura de registros das grandezas de interesse tais como carga, tensão e temperatura.

Uma função de limpeza de vetor denominada **flush_rxbuffer** permite limpar quaisquer resíduos de informação que afetem a aquisição de novos dados a cada recebimento de pacote no vetor rx_data.

Já que a comunicação com o *dynamixel* é *half-duplex*, foi implementada uma função intitulada **bus** para multiplexar o barramento RS-485 conforme o envio (*instruction packets*, TX) e recebimento (*status packets*, RX) de pacotes.

3.5.2.3 Envio de pacotes

send_pkt é a função principal da biblioteca. Ao ser executada no main.c recebe um vetor cujo tamanho varia conforme o tipo de instrução a ser enviada ao servomotor (leitura ou escrita) e quantidade de parâmetros a serem executados. Inicia esse processo limpando o buffer de recepção através da já mencionada função flush_rxbuffer. Executa a seguir a função bus para o modo half-duplex de transmissão (TX) e finalmente copia o pacote para o buffer registrador de transmissão da UART (UCA0TXBUF). Respeitando as devidas temporizações, executa novamente a função bus no modo half-duplex de recepção (RX), recebe um pacote de status ou de retorno com o dado de interesse (e.g. temperatura atual) e o armazena no vetor rx_data.

3.5.2.4 Memória Flash

A memória Flash do MSP430G2553 da plataforma de testes é de 16KB (16384 bytes) (TEXAS INSTRUMENTS, 2013a) e é utilizada, inclusive, para gravação do próprio *firmware*. Embora possam ser gravados desde 1 bit a *words* inteiras (neste hardware, dados de 16-bit), um segmento é a menor porção de memória que pode ser apagada. Além disso, a Flash se divide em *main memory* e *information memory* os quais se diferenciam apenas pelo tamanho do segmento e pelos endereços de memória. A primeira tem 1 ou mais segmentos de 512 bytes (0 a n) e encerra a memória como um todo com o endereço 0x0FFFF. A outra possui 4 segmentos de 64 bytes nomeados de A a D, sendo que o segmento A é bloqueado por conter informações de calibração do sistema (TEXAS INSTRUMENTS, 2013b).

As memórias Flash possuem a peculiaridade de realizarem escritas apenas onde o dado possuir bits iguais a 0. Sendo assim, todo um espaço de memória inteiramente vazio armazena registros de bits iguais a 1 e à medida que são escritos os dados, estes vão sendo substituídos por 0s. Por este motivo, antes de qualquer escrita é necessário efetuar uma operação de limpeza geral ou por segmento.

Considerando o funcionamento das memórias flash e, portanto, a necessidade de efetuar operações de limpeza de segmentos de memória antes de qualquer operação de escrita, garantindo assim a validade dos dados gravados, desenvolveu-se as funções **erase_flash** e **write_flash**.

Em ambos os casos, as interrupções globais precisam ser desabilitadas para que a operação ocorra, caso contrário, os dados podem ser corrompidos em virtude de uma interrupção inesperada de outro periférico. Feito isso, para entrar no modo de acesso à memória Flash é necessário parametrizar três registradores de controle e, em todos os casos, utilizar a palavra chave **FWKEY**, uma espécie de senha para acesso.

A diferença entre as funções, portanto, reside nas parametrizações destes registradores e nas escritas realizadas em sequência a elas. Dessa forma, no caso da função erase_flash que apaga um setor de 512 bytes por vez, a partir do endereço de início informado, uma operação de escrita qualquer dispara a limpeza de todo o bloco. Já na função write_flash utiliza-se um laço para transferir os bytes do vetor de interesse - neste caso, bytes 4, 5 e 6 (este último, se houver) do rx_data - para a memória, também iniciando o armazenamento no endereço informado como entrada da função.

4 Resultados

4.1 Introdução

Neste capítulo serão apresentados os resultados obtidos em ensaios de monitoramento com o servomotor dynamixel. Também será apresentada a especificação técnica para o design eletrônico do sistema personalizado a ser projetado para o robô PI-Ro.

4.2 O Método

O método de avaliação do protótipo e algoritmo implementados consistirá na realização de ensaios em condições a serem descritas a seguir.

Primeiramente, será verificado o funcionamento operacional da plataforma, i.e. comunicação e registro de dados. Somente assim será possível prosseguir com a validação da instrumentação. Dessa forma, para atestar a validade do método e dos dados coletados, ensaios em ambiente controlado foram conduzidos em 2 condições:

- Operação normal do servomotor;
- Operação com falhas.

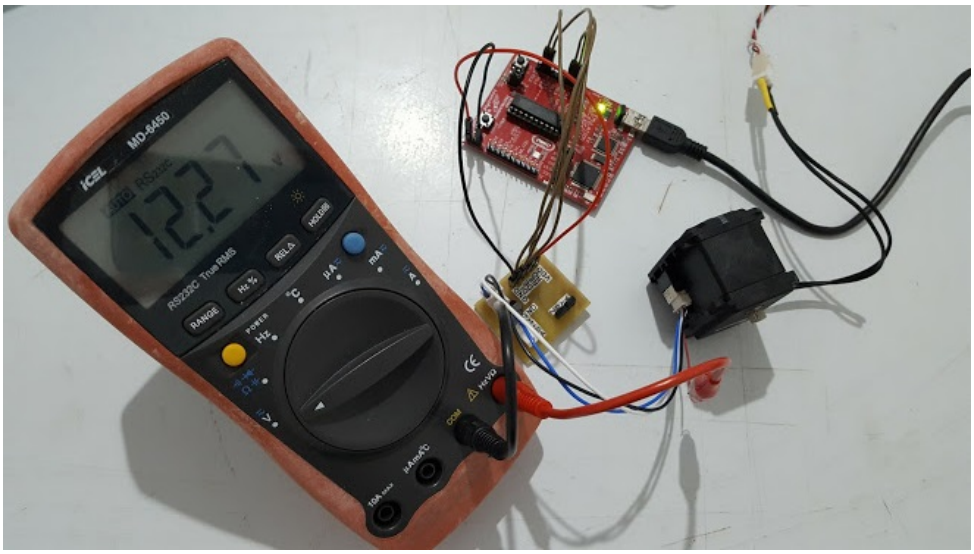
No primeiro caso, espera-se que as medições correspondam a valores dentro da faixa de operação de cada parâmetro coletado conforme configuração prévia realizada na inicialização do sistema.

No segundo caso, falhas conhecidas foram provocadas considerando que as faixas de operação pré-estabelecidas foram escolhidas de forma restritiva para não exceder as capacidades físicas reais do componente.

Em ambos os casos compara-se as medidas tomadas a partir do sistema de monitoramento às medidas feitas por instrumentos de medição de referência: um multímetro ICEL MD-6450, cuja incerteza é de 0,01 V e um termômetro IR MESCO TL-100, cuja incerteza é de $0,1°C$ para atestar a validade dos dados armazenados conforme as figuras 21 e 22.

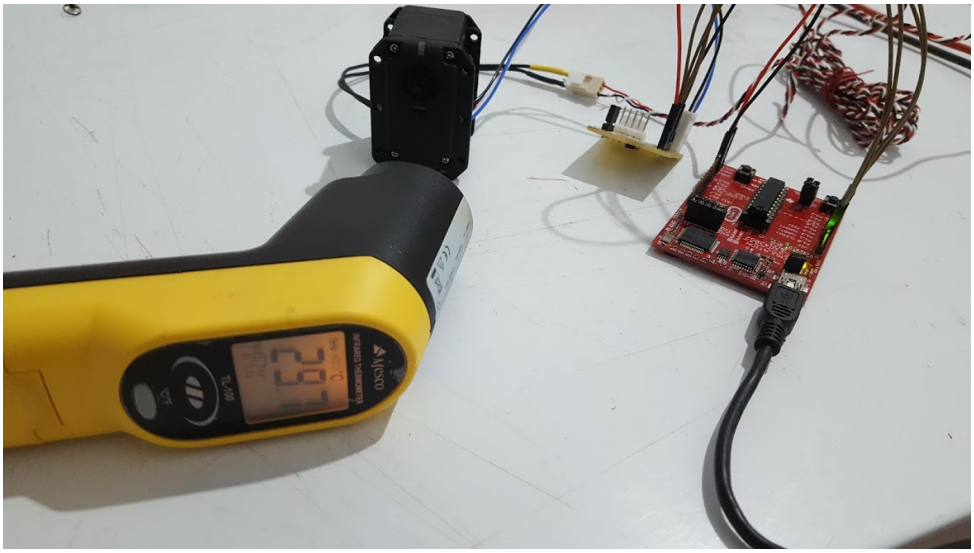
Para alimentação do servomotor foram utilizadas, conforme o ensaio, uma fonte fixa de 12V e a uma fonte de bancada variável 0-30V. Já para provocar as desejadas variações de temperatura foi utilizada uma fonte externa de calor, um soprador térmico de 1200W cujo jato de ar foi direcionado ao servomotor durante os experimentos.

Figura 21 – Multímetro de referência ICEL MD-6450.



Fonte: Própria.

Figura 22 – Termômetro IR de referência MESCO TL-100.



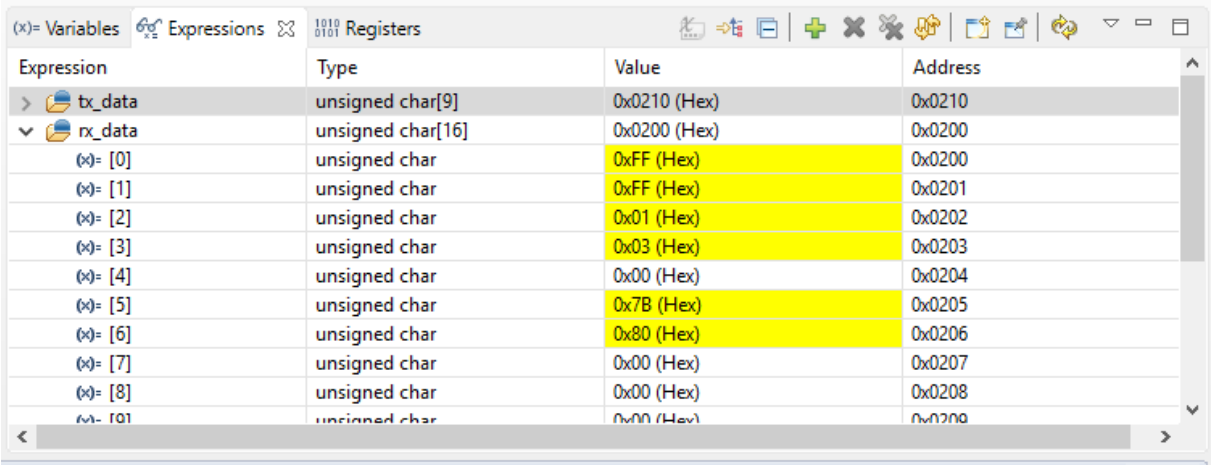
Fonte: Própria.

Os testes foram realizados através da IDE - Code Composer Studio v6.1.2.00015 - em modo **debug** de forma que fosse possível pausar e retomar o programa de acordo com as intervenções necessárias para registro das evidências, tomada e comparação de medidas.

4.3 Verificação Operacional

A fim de verificar o correto funcionamento das funções de comunicação e armazenamento em memória foram tomadas amostras de tensão utilizando uma fonte de alimentação fixa para o servomotor. Na figura 23 é exibido o vetor de recepção *rx_data* recebido em resposta a uma requisição de leitura de tensão onde é possível identificar no byte 4 a inexistência de falha (0x00) e no byte 5 a tensão medida pelo protótipo, 12,3V (0x7B). Na figura 24 tem-se o mapa da memória Flash para uma sequência de leituras da tensão nominal fixada, todas com bytes de erros nulos e concatenados com suas respectivas tensões elétricas na memória.

Figura 23 – Exemplo de pacote de status de leitura de tensão.



Fonte: Própria.

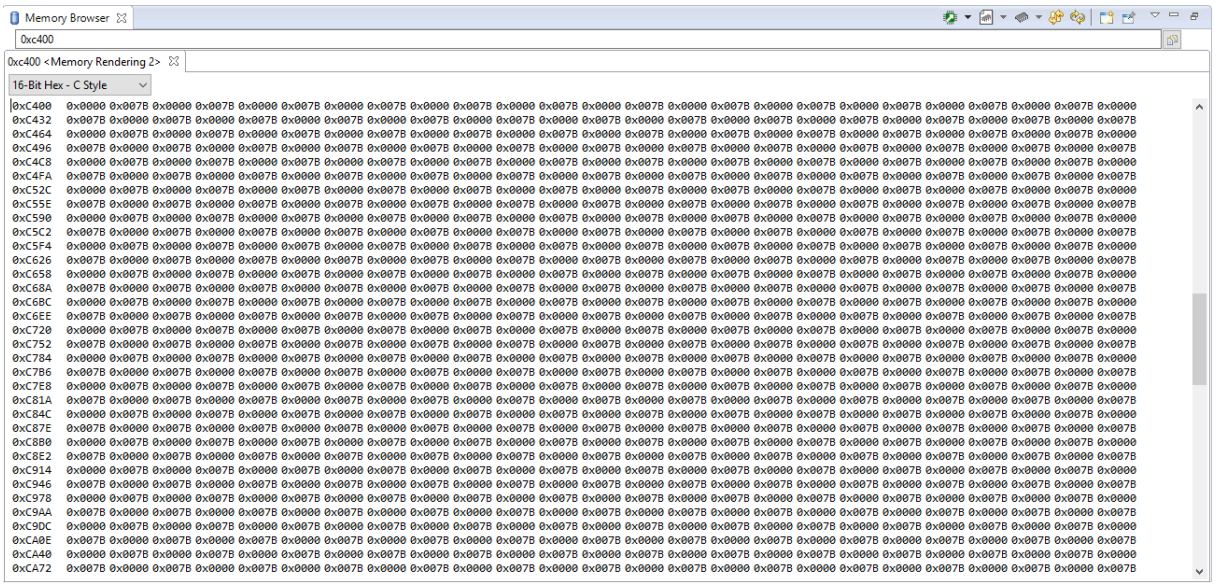
4.4 Ensaios em operação normal

Considera-se aqui operação normal aquela cujos parâmetros de engenharia monitorados permanecem nas faixas de operação esperadas e não comprometem o desempenho do componente, neste caso o servomotor.

Para o parâmetro de tensão elétrica foi estabelecido como intervalo admissível a faixa de 11 a 13V, visto que 12V é a tensão nominal de alimentação do servomotor que garante o seu desempenho da forma esperada. Para estes testes, portanto, foi utilizada uma fonte de alimentação fixa de 12V.

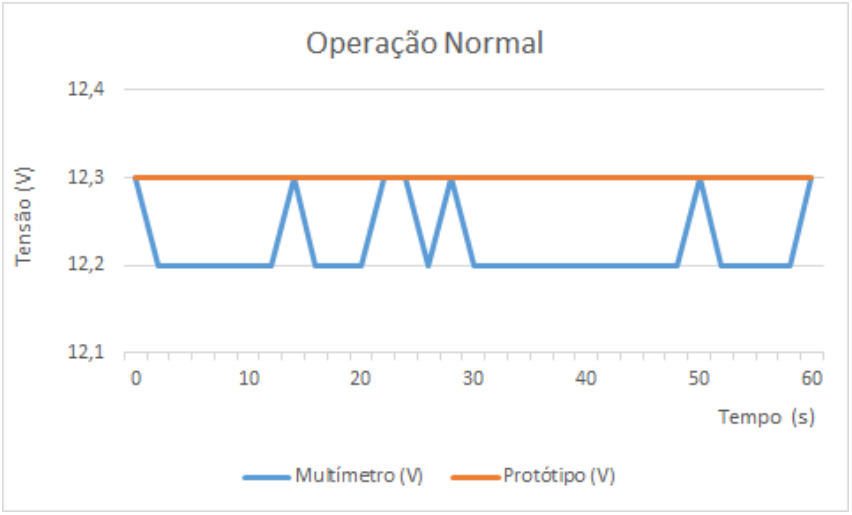
Na figura 25 é possível verificar que a tensão medida pelo protótipo e a tensão medida pelo multímetro se mantém próximas com mínimos desvios, considerando que o algarismo duvidoso no caso do multímetro faz com as medidas variem no gráfico em detrimento das medidas do protótipo.

Figura 24 – Exemplo de mapa de memória de tensão fixa de 12,3V.



Fonte: Própria.

Figura 25 – Ensaio de tensão sem falhas.



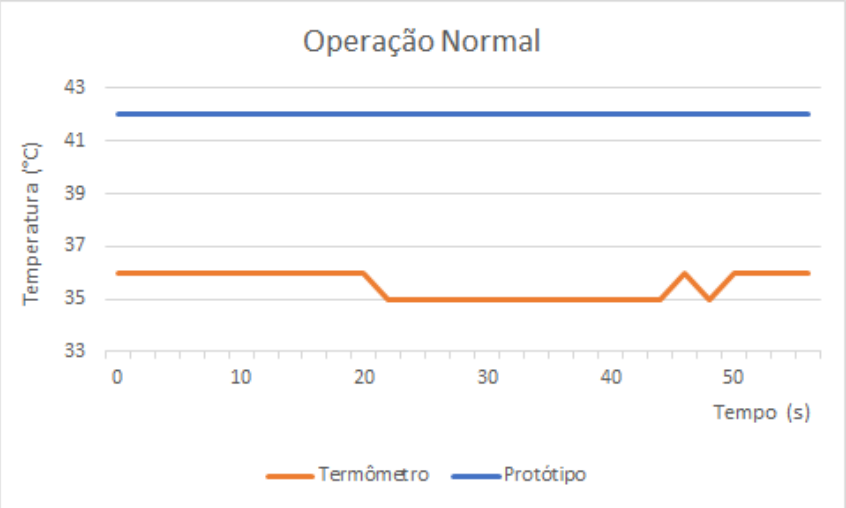
Fonte: Própria.

Para o parâmetro de temperatura, embora as temperaturas de serviço em linhas de alta tensão sejam maiores, variando de 70°C a 90°C, de acordo com as normas brasileiras ABNT NBRs específicas para instalações e condutores elétricos (PRYSMIAN GROUP, 2010), para fins de teste foi adotado o limite superior de 50°C.

Na figura 26 é possível verificar que a temperatura medida pelo protótipo e a temperatura medida pelo termômetro mantém os mesmos perfis de variação com diferenças

em torno de 6°C e 7°C. Isso se deve à diferença do ponto de medição do protótipo, cujo sensor é interno ao servomotor, e o ponto de medição do termômetro IR (infrared) que é externo à carcaça do dynamixel.

Figura 26 – Ensaio de temperatura sem falhas.



Fonte: Própria.

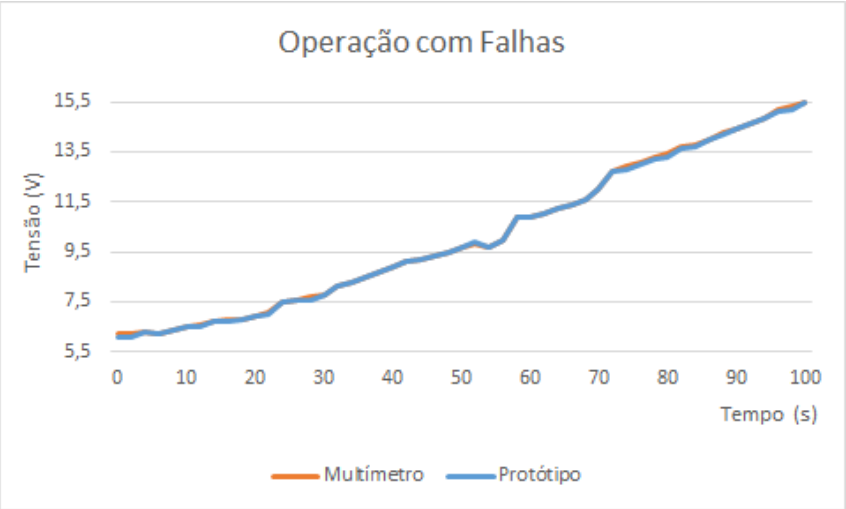
Dessa forma, há dissipação de calor e o resultado externo é ligeiramente menor do que o esperado. Considera-se aqui também que o algarismo duvidoso no caso do termômetro é variável pois o instrumento possui resolução maior que a resolução do protótipo.

4.5 Ensaios em operação com falhas

Nos ensaios com simulação de falhas foi utilizada uma fonte de alimentação variável de bancada, permitindo a variação da tensão fornecida ao servomotor de forma manual em intervalos de tempo regulares para representar oscilações no fornecimento de energia ao sistema e permitir ao sistema detectar erros. Neste caso, quaisquer valores abaixo de 11V ou acima de 13V representam uma falha (Fig. 27). No byte de erro do protocolo dxl, a falha de tensão é representada pelo valor 0x01 conmo pode ser verificado na Figura 20.

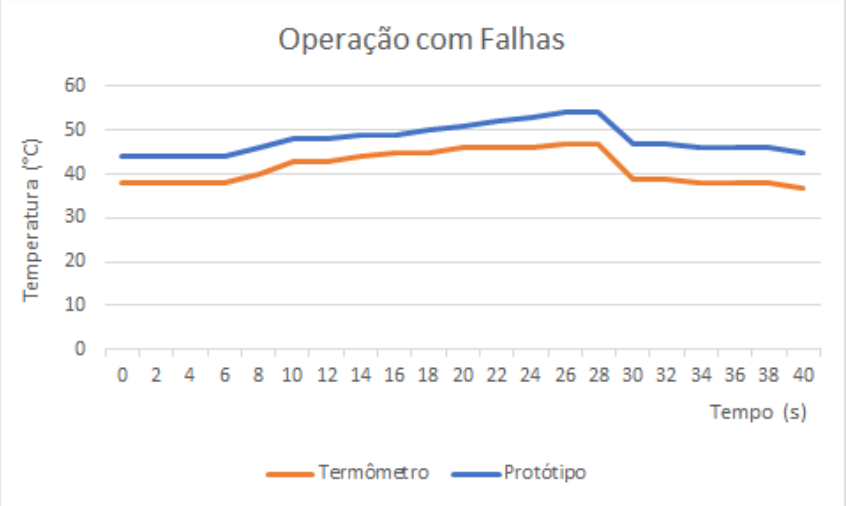
Da mesma forma que na operação normal, há dissipação de calor entre o interior do servomotor e o ambiente externo e o resultado do instrumento de referência é menor do que o apresentado pelo protótipo. Considera-se aqui também que o algarismo duvidoso no caso do termômetro varia pois o instrumento possui resolução maior que a resolução do protótipo e sofre arredondamento. Como pode ser visto na figura 28, em alguns pontos a temperatura excede o limite de 50°C estabelecido, o que representa falhas. No byte de erro do protocolo dxl, a falha é representada pelo valor 0x04.

Figura 27 – Ensaio de tensão com falhas.



Fonte: Própria.

Figura 28 – Ensaio de temperatura com falhas.



Fonte: Própria.

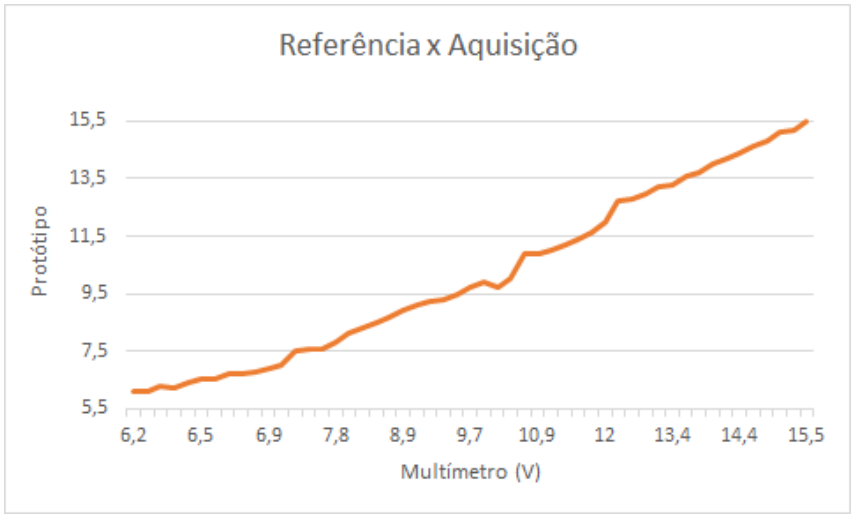
A partir dos gráficos das figuras 29 e 30 é possível confrontar os valores obtidos no sistema de aquisição e nos instrumentos de referência.

Considerando-se que foram feitos ensaios com resultados similares aos apresentados, pode-se concluir, portanto, que o método de monitoramento é válido e que o banco de dados gerado poderá ser utilizado como entrada para o subsistema de confiabilidade do PI-Ro realizar análises de falhas.

Sabendo-se que a função taxa de falha é a taxa instantânea de falha em um

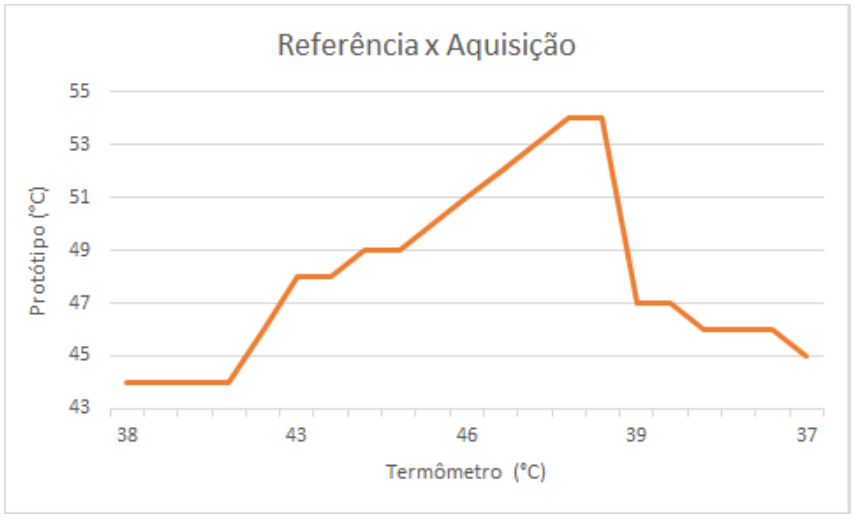
determinado tempo e que a taxa de falha (usualmente referida como λ) é o inverso do MTBF ($1/\lambda$), com o método proposto, esta taxa poderá ser recalculada continuamente ao longo da operação do PI-Ro.

Figura 29 – Sistema de aquisição versus instrumento de referência.



Fonte: Própria.

Figura 30 – Sistema de aquisição versus instrumento de referência.



Fonte: Própria.

4.6 Proposta de projeto

O sistema desenvolvido se propõe a executar, ao longo da vida útil do robô de inspeção, um monitoramento de grandezas para o subsistema de confiabilidade do robô PI-Ro.

Numa variação do que fora proposto por Shalev e Tiran (2007) (vide seção 2.3), os parâmetros de engenharia deverão ser monitorados concomitantemente com a ocorrência observada de falhas durante a operação do robô.

Considerando as informações a seguir, pode-se especificar a memória mínima necessária ao monitoramento das junta do PI-Ro em uma única missão:

- Datasets de 7 bytes que incluem leituras de tensão (*current voltage*, 1 byte), temperatura (*current temperature*, 1 byte) e carga (*current load*, 2 bytes) e respectivos bytes de erro;
- Autonomia de até 4h (14.400 s) do robô PI-Ro conforme previsto em II et al. (2014);
- Intervalos de aquisição de 1s;
- 18 servomotores.

A memória mínima necessária corresponde a $= 7bytes \times 1s \times 18servos \times 14400s$, portanto 1,8144 MB, aproximadamente 2MB.

Portanto, recomenda-se especificar um circuito integrado de memória Flash para compor o projeto do sistema embarcado de monitoramento do PI-Ro que atenda a essa especificação mínima considerando uma única missão. Dado o avanço na miniaturização e escalabilidade da tecnologia de semicondutores, estes CIs hoje costumam ser de baixo custo, podendo se considerar o uso de uma memória que cubra N missões. Um CI de memória flash SMD de 1GB (1000 MB), por exemplo, custa em média USD 5,00 (DIGIKEY ELECTRONICS, 2017) e nas condições apresentadas teria capacidade de manter um banco de dados de cerca de 500 missões de inspeção.

5 Conclusão

A seguir, discute-se o desfecho da pesquisa no que diz respeito ao atendimento dos objetivos propostos e desempenho no desenvolvimento do trabalho, assim como as sugestões de possíveis melhorias e expansão da plataforma apresentada.

5.1 Considerações finais

Diante dos resultados apresentados para dois dos possíveis parâmetros de engenharia dos servomotores, é possível concluir que o método proposto atende aos propósitos de monitoramento desejados e pode ser aplicado aos estudos de confiabilidade a serem realizados sob a forma de algoritmos nos subsistemas do robô de inspeção PI-Ro. O método de aquisição apresentou bons resultados quando comparado a medições realizadas com instrumentos externos de referência. A capacidade de armazenamento de dados proposta fará com que um banco de dados esteja disponível como entrada para o sistema de análise de confiabilidade e, portanto, levará o sistema a tomar suas decisões de operação com base nos mesmos.

Em concordância com os objetivos desta dissertação, conseguiu-se:

- Contextualizar a aplicação a um sistema de confiabilidade de um robô de inspeção;
- Desenvolver um *firmware* para monitoramento de parâmetros para confiabilidade;
- Oferecer um método de atualização da taxa de falha de um componente de acordo com a sua operação;
- Realizar testes individuais com os servomotores MX-28.

5.2 Sugestões de trabalhos futuros

Sendo o robô de inspeção uma solução para que operadoras de energia elétrica realizem inspeções em linhas vivas, evitando a interrupção do fornecimento de energia, a queda de seus indicadores de qualidade e impactos na economia, mesmo que ainda em fase de protótipo, é fundamental que o projeto do robô desde já preveja mecanismos que o tornem tão disponível quanto a energia que ele se dispõe a inspecionar. Dessa forma, provando a sua própria confiabilidade, além de torná-lo mais seguro, o protótipo possivelmente poderá acelerar etapas de homologação até se tornar um produto.

Em continuidade a este trabalho, sugere-se as seguintes ações futuras:

1. Projetar o sistema embarcado de monitoramento com memória compatível coma aplicação;
2. Integrar a plataforma definitiva ao sistema do robô PI-Ro 2.0, atualmente em construção;
3. Realizar testes de longa duração para levantamento de índices de falhas reais;
4. Projetar um sistema de emergência interligado ao sistema de aquisição de dados.

Referências

ACCUENERGY INC. *Wired RS485 Serial Network*. 2016. Disponível em: <https://www.accuenergy.com/>. Citado na página 32.

ALCANTARA, P. X. Sistema de posicionamento e detecção para ultrapassagem autônoma de obstáculos em robô de inspeção de linha de alta tensão. *Faculdade de Tecnologia SENAI CIMATEC*, 2015. Citado 2 vezes nas páginas 14 e 15.

ANALOG DEVICES. *ADM3483/ADM3485/ADM3488/ADM3490/ADM3491 Datasheet*. [S.l.], 2011. Citado na página 40.

ARDUINO. *Frequently Asked Questions*. 2016. Arduino FAQ. Disponível em: <https://www.arduino.cc/en/Main/FAQ>. Acesso em: 27 abr 2016. Citado na página 33.

B&B ELECTRONICS. *Basics of the RS-485 Standard*. 2016. Disponível em: <http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/Basics-of-the-RS-485-Standard.aspx>. Acesso em: 02 jun 2016. Citado na página 38.

BERMAN, B. A. Effective risk management and quality improvement by application of fmea and complementary techniques. *ParagonRx, LCC White Papers*, 2013. Citado na página 28.

BIES, L. *RS485 serial information*. 2015. Disponível em: <https://www.lammertbies.nl/comm/info/RS-485.html>. Citado 2 vezes nas páginas 38 e 39.

BRAGA, N. C. *Padrões de interfaceamento digital (ART339)*. 2016. Instituto Newton C. Braga. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/2189-art339>. Acesso em: 03 abr 2016. Citado na página 31.

CAMPOY, P. et al. An stereoscopic vision system guiding an autonomous helicopter for overhead power cable inspection. *Robot Vision, Volume 1998 of the series Lecture Notes in Computer Science pp 115-124*, 2001. Citado na página 21.

DAVIES, J. H. *MSP430 Microcontroller Basics*. [S.l.]: Newnes - Elsevier, 2008. Citado 3 vezes nas páginas 29, 30 e 32.

DHILLON, B.; FASHANDI, A. Safety and reliability assessment techniques in robotics. *Robotica*, Cambridge University Press, v. 15, p. 701–708, 1997. Citado 3 vezes nas páginas 22, 24 e 26.

DIGIKEY ELECTRONICS. *Micron IC FLASH 1GBIT 24TBGA*. 2017. Disponível em: <http://www.digikey.com/product-detail/en/micron-technology-inc/MT29F1G01ABAFD12-AAT-F-TR/557-1652-1-ND/6207191>. Citado na página 56.

DUAN, R. xing; ZHOU, H. lin. A new fault diagnosis method based on fault tree and bayesian networks. *International Conference on Future Electrical Power and Energy Systems*, 2012. Citado na página 23.

EMPRESA DE PESQUISA ENERGéTICA - EPE. *Balanço Energético Nacional 2015: Ano base 2014*. 2015. Citado na página 13.

FERREIRA, F. H. *As diferenças entre defeito, erro e falha*. 2007. Disponível em: <https://ferhenriquef.com/2012/10/24/as-diferenas-entre-defeito-erro-e-falha/>. Citado na página 13.

II, E. J. L. et al. Robô autônomo para inspeção de linhas de alta tensão. *XX Congresso Brasileiro de Automática*, 2014. Citado na página 56.

INTEL CORP. *Boards and Kits*. 2016. Intel - Data Center Solutions, IOT and PC Innovation. Disponível em: <http://www.intel.com/content/www/us/en/homepage.html>. Acesso em: 27 abr 2016. Citado na página 33.

JINGWEI, G. et al. A new condition monitoring and fault diagnosis method of engine based on spectrometric oil analysi. *International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011)*, 2011. Citado na página 23.

JONES, D. Power line inspection - an uav concept. *Autonomous Systems, the IEE Forum*, 2005. Citado na página 21.

KATRASNIK, J.; PERNUS, F.; LIKAR, B. A survey of mobile robots for distribution power line inspection. *IEEE Transactions on Power Delivery, Volume: 25, Issue: 1*, 2010. Citado na página 21.

KUHN, A. *Artificial Intelligence, Computational Intelligence, SoftComputing, Natural Computation - what's the difference?* 2016. Disponível em: <http://www.andata.at/en/answer/artificial-intelligence-computational-intelligence-softcomputing-natural-computation-whats-the-differe html>. Citado na página 12.

LI, Z. et al. Condition monitoring and fault diagnosis for marine diesel engines using information fusion techniques. *ELEKTRONIKA IR ELEKTROTECHNIKA, Vol 123, No 7*, 2012. Citado na página 23.

LUDAN, W. et al. Development and control of an autonomously obstacle-navigation inspection robot for extra-high voltage power transmission lines. *in Proceedings Int. Joint Conf. SICE-ICASE*, 2006. Citado na página 22.

MICROCHIP TECHNOLOGY INC. *Microcontrollers*. 2016. Disponível em: <http://www.microchip.com/design-centers/microcontrollers>. Acesso em: 14 mai 2016. Citado na página 34.

MIR INNOVATION. *Inspection and maintenance solutions for power transmission systems*. 2016. Disponível em: <http://mir-innovation.hydroquebec.com/mir-innovation/en/transmission-solutions-linescout.html>. Citado na página 22.

MONTAMBAULT, S.; POULIOT, N. Linescout technology: Development of an inspection robot capable of clearing obstacles while operating on a live line. *in Proc. IEEE 11th Int. Conf. Transmiss. Distrib. Construction, Oper. Live-Line Maintenance—ESMO*, 2006. Citado 2 vezes nas páginas 21 e 22.

MONTANI, S.; BOBBIO, L. P. A.; CODETTA-RAITERI, D. RADYBAN: A tool for reliability analysis of dynamic fault trees through conversion into dynamic bayesian networks. *Reliability Engineering and System Safety 93*, 2008. Citado na página 23.

NATIONAL INSTRUMENTS. *Uma rápida comparação das Interfaces de Comunicação Serial RS-232, RS-422, e RS-485*. 2016. NI KnowledgeBase. Disponível em: <http://digital.ni.com/public.nsf/allkb/DE153F74C4BF3AD8862576AB006C1AAF>. Acesso em: 03 abr 2016. Citado na página 31.

NELSON, W. R. REACTOR: an expert system for diagnosis and treatment of nuclear reactors. *AAAI-82 Proceedings*, 1982. Citado na página 23.

ONS. *Evolução da Capacidade Instalada do SIN*. 2016. Disponível em: <http://www.ons.org.br/download/biblioteca_virtual/publicacoes/DADOS2014_ONS/7_2.html>. Acesso em: 19 abr 2016. Citado na página 14.

ONS. *O que é SIN?* 2016. Disponível em: <http://www.ons.org.br/conheca_sistema/o_que_e_sin.aspx>. Acesso em: 19 abr 2016. Citado na página 13.

PAPADOPOULOS, Y. Model-based system monitoring and diagnosis of failures using statecharts and fault trees. *Reliability Engineering and System Safety 81*, 2003. Citado na página 23.

PENG, H. et al. On-line monitoring and diagnosis of failures using control charts and fault tree analysis (fta) based on digital production model. *Knowledge Science, Engineering and Management, Volume 4798 of the series Lecture Notes in Computer Science pp 544-549*, 2007. Citado na página 23.

PERGAM USA. *Aerial Power Line Inspection Patrols: Helicopter vs. UAV*. 2015. Disponível em: <http://pergamusa.com/2015/08/09/aerial-power-line-inspection-patrols-helicopter-vs-uav/>. Citado na página 20.

PRYSMIAN GROUP. *Manual Prysmian de Instalações Elétrica*. 2010. Disponível em: <http://br.prysmiangroup.com/br/files/manual_prysmian.pdf>. Citado na página 52.

RANGEL, R. K.; KIENITZ, K. H.; BRANDãO, M. P. Sistema de inspeção de linhas de transmissão de energia elétrica utilizando veículos áereos não-tripulados. *Brazilian Symposium on Aerospace Eng. & Applications*, 2009. Citado na página 14.

ROBOTIS INC. *e-Manual v1.27.00 - MX-28T / MX-28R*. 2010. Disponível em: <http://support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm>. Acesso em: 11 jan 2016. Citado 4 vezes nas páginas 35, 42, 44 e 46.

ROBOTIS INC. *Instruction Packet*. 2010. Disponível em: <http://support.robotis.com/en/product/dynamixel/communication/dxl_packet.htm>. Acesso em: 06 fev 2016. Citado 2 vezes nas páginas 41 e 42.

ROBOTIS INC. *Kind of Instruction*. 2010. Disponível em: <http://support.robotis.com/en/product/dynamixel/communication/dxl_instruction.htm>. Acesso em: 06 fev 2016. Citado na página 42.

ROBOTIS INC. *Overview of Communication*. 2010. Disponível em: <http://support.robotis.com/en/product/dynamixel/dxl_communication.htm>. Acesso em: 06 fev 2016. Citado na página 43.

SAWADA, J. et al. A mobile robot for inspection of power transmission lines. *IEEE Trans. Power Del., vol. 6, no. 1, pp. 309–309*, 1991. Citado na página 21.

SHALEV, D. M.; TIRAN, J. Condition-based fault tree analysis (CBFTA): A new method for improved fault tree analysis (FTA), reliability and safety calculations. *Reliability Engineering and System Safety*, v. 92, p. 1231–1241, 2007. Citado 3 vezes nas páginas 23, 24 e 56.

SIEGWART, R.; NOURBAKHSH, I. *Introduction to Autonomous Mobile Robots*. [S.l.]: The MIT Press, 2004. Citado na página 20.

SIMON, D. E. *An Embedded Software Primer*. [S.l.]: Addison-Wesley, 2005. Citado na página 29.

TANG, L.; WANG, H.; FANG, L. Development of an inspection robot control system for 500 kv extra-high voltage power transmission lines. *in Proceedings SICE 2004 Annual Conference*, 2004. Citado na página 22.

TAVARES, L.; SEQUEIRA, J. S. Riol—robotic inspection over power lines. *in Proc. 6th IFAC Symp. Intell. Autonom. Vehicles*, 2007. Citado na página 22.

TECHNAVIO. *Why Zigbee is the Fastest Growing Trend in Wireless Technology*. 2016. Disponível em: <http://www.technavio.com/blog/why-zigbee-fastest-growing-trend-wireless-technology>. Citado na página 31.

TECNOGERA. *Como funcionam as linhas de transmissão de energia elétrica*. 2015. Disponível em: <http://www.tecnogerageradores.com.br/blog/como-funcionam-linhas-de-transmissao-de-energia-eletrica/>. Acesso em: 20 abr 2016. Citado na página 14.

TECNOGERA. *Entenda a definição de efeito Corona e como age em isoladores*. 2015. Disponível em: <http://www.tecnogerageradores.com.br/blog/entenda-a-definicao-de-efeito-corona-e-como-age-em-isoladores/>. Acesso em: 20 abr 2016. Citado na página 20.

TEXAS INSTRUMENTS. *MSP430G2x53 Datasheet*. [S.l.], 2013. Citado na página 47.

TEXAS INSTRUMENTS. *MSP430x2xx Family User's Guide*. [S.l.], 2013. Citado na página 47.

TEXAS INSTRUMENTS. *MSP-EXP430G2*. 2016. Disponível em: <www.ti.com/ww/en/launchpad/launchpads-msp430-msp-exp430g2.html>. Acesso em: 06 mar 2016. Citado na página 37.

TEXAS INSTRUMENTS. *Ti LaunchPad*. 2016. TI LaunchPad Development Ecosystem. Disponível em: <http://www.ti.com/ww/en/launchpad/launchpad.html>. Acesso em: 26 abr 2016. Citado na página 34.

UNDERWOOD, W. E. A CSA model-based nuclear power plant consultant. *AAAI-82 Proceedings*, 1982. Citado na página 23.

U.S. NUCLEAR REGULATORY COMMISSION. *Fault Tree Handbook - NUREG-0492*. [S.l.]: US Government Printing Office, 1981. Citado 2 vezes nas páginas 25 e 27.

WEI, S. et al. Design and validation of a novel robot for power lines inspection. *Proceeding of the 2015 IEEE International Conference on Information and Automation*, 2015. Citado na página 22.

XIAO, X. H. et al. Dynamic simulation and experimental study of inspection robot for high-voltage transmissio nline. *J. Central South Univ. Technol., vol. 12, no. 6, pp. 726–731*, 2005. Citado na página 22.

ZHOU, G. et al. Robust real-time uav based power line detection and tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, 2016. Citado na página 21.

Apêndices

APÊNDICE A – Código Fonte

A.1 main.c

```
#include <msp430g2553.h>
#include <stdint.h>
#include "dynamixel.h"
                                  //  Header  ID  Len  Inst  P1  P2  ChSum
//Config
const uint8_t ALARM_LED[8]    ={0xFF,0xFF,0xFE,0x04,0x03,0x11,0x25,0xC6}; // alarm led setup
const uint8_t ALARM_SHTDWN[8]={0xFF,0xFF,0xFE,0x04,0x03,0x12,0x25,0xC5}; // alarm shutdown setup
const uint8_t MAX_TORQ[9]     ={0xFF,0xFF,0x01,0x05,0x03,0x0E,0xFF,0x03,0xE6}; //max torque(2.5Nm)
const uint8_t VOLT_LIM[9]     ={0xFF,0xFF,0x01,0x05,0x03,0x0C,0x5C,0x82,0x0C}; //limits(11-13V)


//Read
const uint8_t READ_TEMP[8]    ={0xFF,0xFF,0x01,0x04,0x02,0x2B,0x01,0xCC}; // read curr. temperat.
const uint8_t READ_VOLT[8]    ={0xFF,0xFF,0x01,0x04,0x02,0x2A,0x01,0xCD}; // read curr. voltage
const uint8_t READ_LOAD[8]    ={0xFF,0xFF,0x01,0x04,0x02,0x28,0x02,0xCE}; // read curr. load

int *addr = (int *)0xFE00;                    //(main) flash memory start address

int main(void) {

  WDTCTL = WDTPW | WDTHOLD;             // Stop watchdog timer
  DCO_init();                                          // Clock config
  PORT_init();                                         // I/O config
  UCA0_init();                                         // UART0 config


/*Dynamixel limits and alarm Config*/
  send_pkt(ALARM_LED);
  __delay_cycles(1000000);

  send_pkt(ALARM_SHTDWN);
  __delay_cycles(1000000);

  send_pkt(MAX_TORQ);
  __delay_cycles(1000000);

  send_pkt(VOLT_LIM);
  __delay_cycles(1000000);


 /*Erases segment*/
erase_flash(addr);


  while(1) {


          send_pkt(READ_VOLT);
          __delay_cycles(1000000);

          write_flash(addr);
```

```
            send_pkt(READ_TEMP);
            __delay_cycles(1000000);

            write_flash(addr);

            send_pkt(READ_LOAD);
            __delay_cycles(1000000);

            write_flash(addr);

}

}
```

A.2 dynamixel.c

```
#include <msp430g2553.h>
#include <stdint.h>
#include "dynamixel.h"


uint8_t tx_data[PKT_SIZE];                          // for debug
uint8_t *ptr_tx = tx_data;

uint8_t rx_data[16];
uint8_t *ptr_rx = rx_data;


void PORT_init (void) {
        //------------------ Configuring the LED's --------------------------//

        P1DIR  |=  BIT0;          // P1.0 output
        P1OUT  &= ~BIT0;          // P1.0 = 0

        //-------------- Configuring MAX485 Control Lines ------------------//

        P2DIR  |= BIT0 + BIT1;  // P2.0 -> DE output
                                                        // P2.1 -> ~RE output
        P2OUT   = 0x00;           // P2.0 and P2.1 = 0

        //---------- Setting the UART function for P1.1 & P1.2 ----------//

        P1SEL  |=  BIT1 + BIT2; // P1.1 UCA0RXD input
        P1SEL2 |=  BIT1 + BIT2; // P1.2 UCA0TXD output
}


void DCO_init (void) {
  DCOCTL  = 0;                              // Select lowest DCOx and MODx settings<
  BCSCTL1 = CALBC1_1MHZ;          // Set DCO
  DCOCTL  = CALDCO_1MHZ;
}


void UCA0_init (void) {
        //---------------- Configuring the UART(USCI_A0) --------------------//

        UCA0CTL0 = UCMODE_0;
        UCA0CTL1 |=  UCSSEL_3 + UCSWRST;
        UCA0BR0   =  18;
        UCA0BR1   =  0;
        UCA0MCTL  =  UCBRS0;
        UCA0CTL1 &= ~UCSWRST;    // Release UART

        //-------------------- Interrupts Enabling --------------------------//

        UC0IE = 0x01;                       //RX interrupt enabled
        __bis_SR_register(GIE); //Global Interrupt Enable
}


// Erase rx_data array
void flush_rxbuffer (void) {
        uint8_t i;
```

```
        for (i=0; i<16; i++) { rx_data[i] = 0x00; }
}


// Bus enable config for TX -or- RX (Half Duplex)
void bus (uint8_t sel) {
        if (sel) { P2OUT |=   BIT1 + BIT0 ; } // tx
        else     { P2OUT &= ~(BIT1 + BIT0); } // rx
}


// Send Instruction Packet
void send_pkt(const uint8_t *pkt) {
        uint8_t i;
        flush_rxbuffer();
        bus(TX);
        __delay_cycles(300);                                    // 300us
        ptr_tx = &tx_data;                                              // for debug
        for(i=0; i<PKT_SIZE; i++) {
                UCA0TXBUF = *pkt;
                        *ptr_tx = *pkt;                                 // for debug
                        *ptr_tx++;                                              // for debug
                *pkt++;
                while(!(UC0IFG & UCA0TXIFG));
        }
        __delay_cycles(300);                                    // 300us
        bus(RX);
        ptr_rx = rx_data;
        __delay_cycles(3000);                                   // 3ms (enough for the end of rx data)
}


// Delay function: this is not equivalent to __delay_cycles !
void delay(uint32_t delayTime) {
        uint32_t i = 0;
        while(i < delayTime) {i++;}
}



//------------------- Handling Flash Memory ------------------------//

void erase_all(int *addr){
        __bic_SR_register(GIE);                                 //Global Interrupt Disable

//      while(BUSY & FCTL3);                    // Check if Flash being used

        FCTL2 = FWKEY + FSSEL_1 + FN1;          // Clk = MCLK/3
        FCTL1 = FWKEY + MERAS;                  // Set Mass Erase bit
        FCTL3 = FWKEY;                          // Clear Lock bit
        *addr = 0;                              // Dummy write to erase Flash segment

//      while(BUSY & FCTL3);                    // Check if Flash being used

        FCTL1 = FWKEY;                          // Clear WRT bit
        FCTL3 = FWKEY + LOCK;                   // Set LOCK bit

        __bis_SR_register(GIE);                                 //Global Interrupt Enable
}
```

```
void erase_flash(int *addr){

        __bic_SR_register(GIE);                         //Global Interrupt Disable

        //while(BUSY & FCTL3);                  // Check if Flash being used

        FCTL2 = FWKEY + FSSEL_1 + FN1;        // Clk = MCLK/3
        FCTL1 = FWKEY + ERASE;                // Set Erase bit
        FCTL3 = FWKEY;                        // Clear Lock bit
        *addr = 0;                            // Dummy write to erase Flash segment

        //while(BUSY & FCTL3);                  // Check if Flash being used

        FCTL1 = FWKEY;                        // Clear WRT bit
        FCTL3 = FWKEY + LOCK;                 // Set LOCK bit

        __bis_SR_register(GIE);                         //Global Interrupt Enable
}


int write_flash(int *addr){
        __bic_SR_register(GIE); //Global Interrupt Disable

        int i = 0;
        FCTL2 = FWKEY + FSSEL_1 + FN1;        // Clk = MCLK/3
        FCTL3 = FWKEY;                        // Clear Lock bit
        FCTL1 = FWKEY + WRT;                  // Set WRT bit for write op

        for (i=0; i<16; i++){
                while(!(FCTL3 && WAIT));                        // When !&& is 0, ready to write
                *addr = rx_data[i];                     // Copy data to flash
                *addr++;

        return addr;
        }

        FCTL1 = FWKEY;                        // Clear WRT bit
        FCTL3 = FWKEY + LOCK;                 // Set LOCK bit

        __bis_SR_register(GIE);               //Global Interrupt Enable
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIAB0RX(void)
{
        *ptr_rx = UCA0RXBUF;
        *ptr_rx++;
        UC0IFG &= ~UCA0RXIFG;
}
```

A.3 dynamixel.h

```
#include <stdint.h>

#define TX                1
#define RX                0
#define PKT_SIZE 9        // max number of bytes TX can send

void DCO_init(void);
void PORT_init(void);
void UCA0_init(void);
void bus(uint8_t sel);
void send_pkt(const uint8_t *pkt);
void delay(uint32_t delayTime);
void erase_all(int *addr);
void erase_flash(int *addr);
void write_flash(int *addr);
```