Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

12 de Dezembro de 2017

# SISTEMA DE CONTROLE EMBARCADO DO ROBÔ BIPEDE LEO2

Vinicius Tostes Seixas

PUC RIO

Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

# SISTEMA DE CONTROLE EMBARCADO DO ROBÔ BÍPEDE LEO2


## Aluno: Vinicius Tostes Seixas

## Orientador: Wouter Caarls


Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

## Agradecimentos

Agradeço aos amigos e engenheiros André Felipe Barroso,Adler Linhares,Carlos Rodrigues Barroso,Matheus Ribeiro Castro,Thaís Joffe,Bruno Moreira,Renato Gabriel,Luan Alves,Alessandro Soares e Daniel Freitas por terem me acolhido na amizade e companheirismo durante o processo de graduação em engenharia de controle e automação,pois o companheirismo e troca de conhecimento recíproco foram fundamentais e vitais para enfrentar em conjunto as interpéries da vida acadêmica,propiciando amizades a serem cultivadas pelo resto da vida.

Agradeço ao amigo Matemático e Doutor Eric Cardona pela amizade íntegra durante a vida acadêmica.

Agradeço ao Professor William Barbosa pelos ensinamentos durante a vida acadêmica e por ensinar a lutar pelo que se sonha.

Agradeço ao Professor Doutor josé Luiz Lizarbe Chira por ter me ensinado a como se estudar engenharia.

Agradeço ao Ilmo. Professor Guilherme Penello temporão por ter me colocado em uma situação complicada,porém por ter confiado em mim na capacidade de superá-la.

Agradeço a Ilma. Professora Karla Teresa Figueiredo por ter me cedido a oportunidade como estagiário do laboratório do departamento de engenharia elétrica LIRA.

Agradeço ao Ilmo. Professor Moisés Henrique Szwarcman pelos profundos conhecimentos que me fizeram pensar de uma forma diferente devido a sua didática.

Agradeço ao Ilmo. Professor e amigo Eduardo Costa pelos ensinamentos,confiança e reconhecimento, assim como pela atenção fornecida nos assuntos sobre engenharia eletrônica.

Agradeço ao Ilmo. Professor Adriano Branco,pela paciência,atenção e pelos ensinamentos sobre engenharia de software,em relação ao processamento computacional de informações.

Agradeço a Ilma. Professora Ana Maria Beltran Pavani pelos conhecimentos e por meio de sua didática aprendi que a labunta se começa na madrugada.

Agradeço ao Ilmo. Professor Wouter Caarls pela oportunidade cedida em implementar um projeto de fim de curso em um sistema robótico real,onde tive que aprender conceitos completamente novos e implementá-los em uma forma que me era pouco habitual,tirando-me da zona de conforto.Levando a ter a visão da importância da programação e dos processos de comunicação de dados na engenharia de controle e automação contemporâneas.Agradeço também por toda a paciência cedida.

Agradeço ao Ilmo. Diginíssimo Professor Washington Braga Filho,por ter me dignado confiança de vencer perante as dificuldades.

Agradeço a Vanessa de Almeida Souza pelo carinho e companheirismo durante todos os anos de graduação,assim como pela confiança recíproca,propiciando-me uma rotina de disciplina e desempenho afim de atingir um bem maior.

Agradeço a minha mãe Tania Maria Maioli Tostes pela perseverança na vida,assim como exemplo de criatividade.

Agradeço a meu tio Renato Maioli Tostes pelo exemplo de trabalho duro e por ser um vencedor na vida.

Agradeço ao meu avô José Barroso Tostes pelo exemplo de homem e ensinar que com dedicação,humildade e servidão,o conhecimento se transforma em grande bem a sociedade.

Agradeço por fim a todos os amigos,dos quais tive que me manter afastado devido a esta empreitada,porém que sempre estiveram dentro do coração e serviram de grandes exemplos a enfrentar as interpéries da vida acadêmica.

## Resumo

O projeto em questão tem por objetivo a implementação de um sistema de controle embarcado afim de realizar a dinâmica 2D do robô bípede LEO2. Desenvolvendo tanto o software de processamento de dados do sistema quanto a instalação física de seus componentes eletro-eletrônicos, realizando testes de comportamento em tempo real, coletando dados relativos a transmissão de informação para que seja possível assim analisar a dinâmica do robô e otimizar seu comportamento.

**Palavras-chave: Robô; Rede de Comunicação;Dynamixel;Transdutores;Controlador Principal;Software desenvolvido;Ambiente de programação;RS-485;Comunicação Serial;Monitoramento;Máquina de Estados;Jitter de comunicação**

## Sumário

## Lista de Figuras

## Lista de Tabelas

## Lista de Siglas e Abreviaturas

mm.....milímetros

lb.....libra-força

msec.....milisegundos

rpm.....radianos por minuto

mA.....miliampères

V.....Volt

Kohm.....KiloOhm

Nm.....Newton-Metro

Bit.....unidade numérica base binária

Byte.....conjunto padrão computacional de 8 bits

Gp.....ganho do algorítimo do controlador proporcional do ambiente de programação do computador

BJT.....bipolar junction transistor

Kp.....ganho proporcional de um algorítmo de controle PID

Ki.....ganho integral de um algorítmo de controle PID

Kd.....ganho derivativo de um algorítmo de controle PID

Bps.....bits por segundo

Mbps.....um milhão de bits por segundo

MHZ.....mega-hertz

TTL.....Transistor-Transistor-Logic Level

USB.....Universal Serial BUS

IDE.....integrated develop enviroment

Rx.....reception serial communication LED

Tx.....transmition serial communication LED

RS-485....standard that defines electrical characteristics of components of a serial communications system

# 1. INTRODUÇÃO

## 1.1. Visão Geral do Projeto

O projeto em questão tem por objetivo a implementação de um sistema de controle embarcado afim de realizar a dinâmica 2D do robô bípede LEO2. Desenvolvendo tanto o software de processamento de dados do sistema quanto a instalação física de seus componentes eletro eletrônicos, realizando testes de comportamento em tempo real, coletando dados relativos a transmissão de informação para que seja possível assim analisar a dinâmica do robô e otimizar seu comportamento.

A dinâmica física do robô é realizada pelo seu sistema de controle embarcado que conecta seus componentes eletrônicos e processa seus sinais em um software desenvolvido para operar como uma rede de transmissão de dados,que estabelece a recepção e transmissão de dados entre os componentes eletrônicos e interliga os dados referentes a estes componentes processado-os computacionalmente.Gerando sinais de controle para os atuadores do robô conforme uma lei de controle dinâmico.

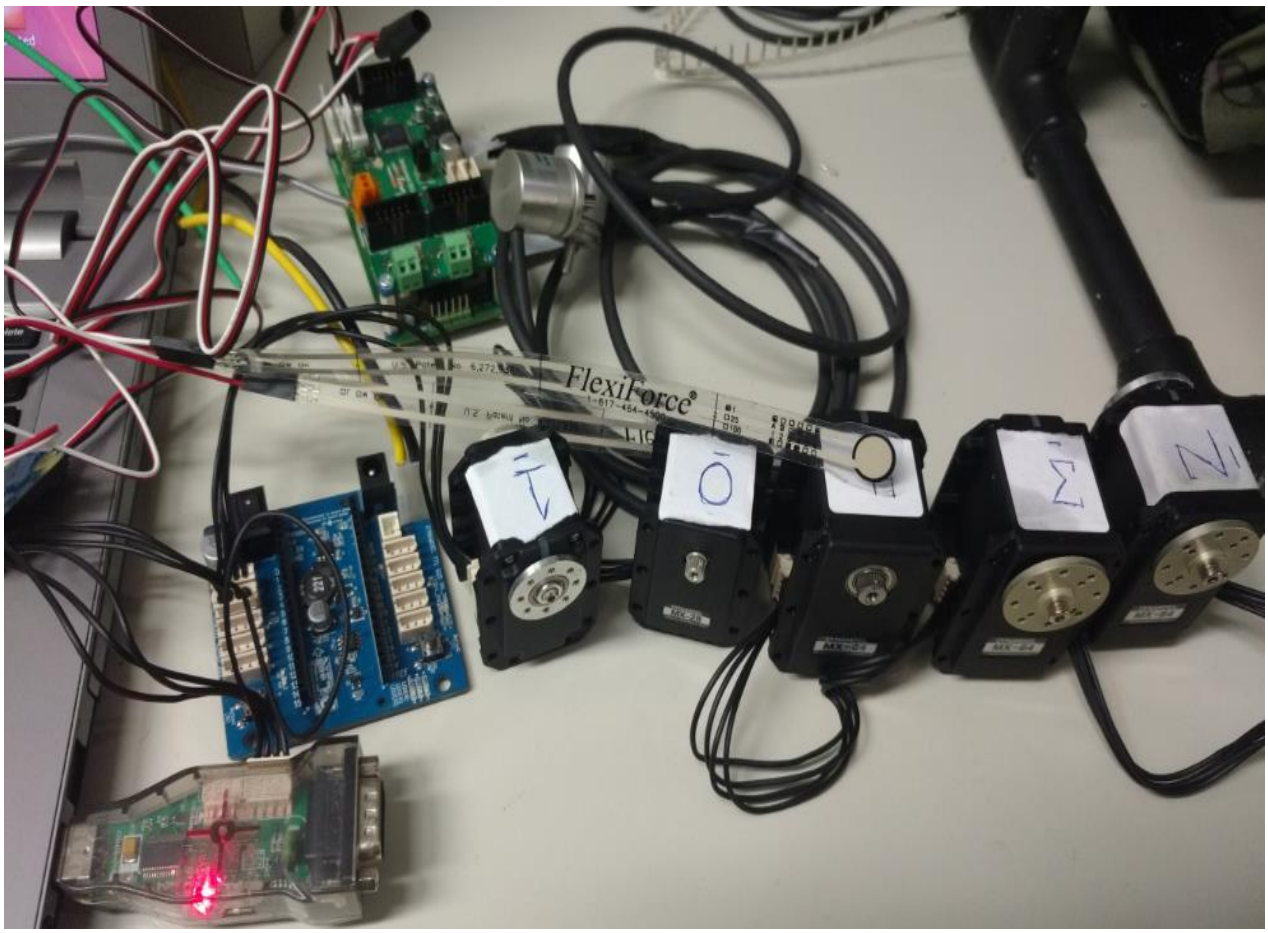


**Figura 1 - Visão Geral do Projeto**

## 1.2. Descrição do Robô Bípede LEO2 e Aplicações

O Robô Bípede LEO2 é constituído por componentes mecânicos e eletro eletrônicos que interagem entre si tendo sido projetado com base no seu antecessor LEO, desenvolvido no laboratório de biomecânica da faculdade de engenharia mecânica, marítima e de materiais da universidade técnica de Delft, porém com uma configuração visando gerar maior robustez (quanto aos impactos durante as quedas) e menor complexidade(visando otimizar o desempenho quanto a dinâmica). Suas aplicações são relativas a pesquisa e ensino e possui como objetivo realizar a tarefa de andar sobre um percurso conforme a aplicação de diversas políticas de controle.

**Figura 2 – Esquemático do Robô Bípede LEO2**

Os componentes mecânicos de LEO2 são:

- Os pés do robô (semicirculares - para que seja realizado o contato físico com o solo de forma circular e gradual);
- Juntas articuladas, que conectam as partes mecânicas com os componentes eletrônicos de monitoramento (transdutores) e atuação direta(servos motores) do robô;
- Sistema de duas hastes flexíveis conectadas transversalmente a um eixo livre para rotação(possui instalação elétrica para energizar o sistema), permitindo ao robô realizar movimentos indiretos, que não caia para os lados e garantindo uma locomoção em duas dimensões e uma trajetória circular.

Algumas peças mecânicas do robô possuem uma estrutura material composta de resina e plástico moldado, como os pés e o sistema de amortecimento, enquanto outras como os conectores das partes mecânicas com eletrônicas são de alumínio.

Os componentes eletrônicos do robô são servos motores e transdutores.

Os servos motores possuem a função de realizar a dinâmica física direta do sistema, correspondendo aos termos de entrada, atribuem torque e velocidade aos eixos do robô, fazendo com que este varie sua posição . Também ditam a quantidade de graus de liberdade do robô.
LEO2 contém cinco servos motores, logo possui cinco graus de liberdade.

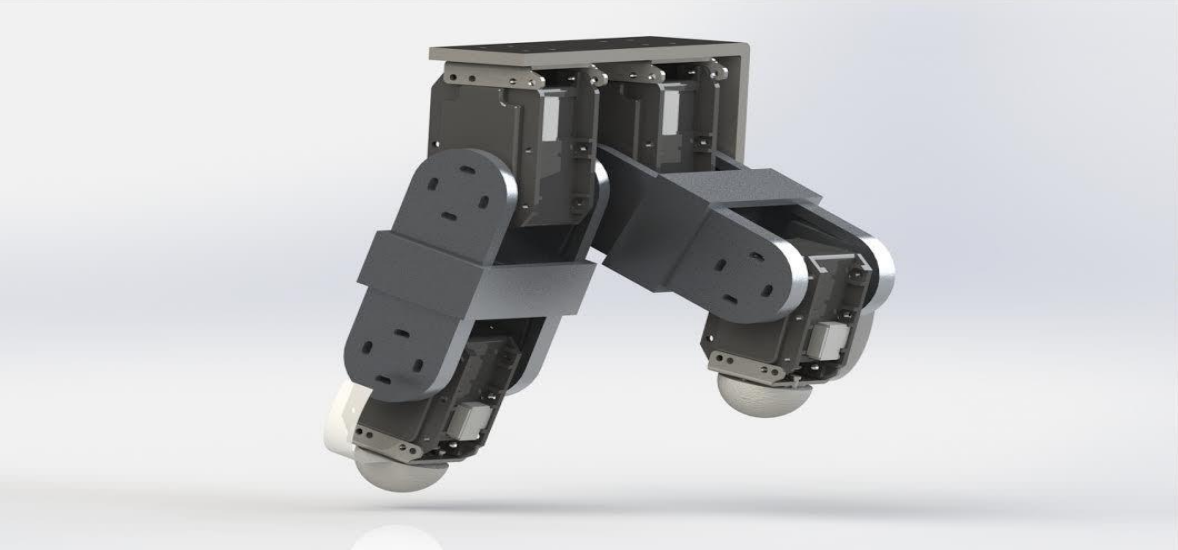

**Figura 3 – Esquemático de LEO2 realizando a dinâmica bípede**

Dois servos motores realizam a função da dinâmica do joelho(pernas inferiores), dois implementam a dinâmica das pernas(pernas superiores) e um instalado sobre o final do conjunto de hastes perpendiculares simula o movimento do braço . Este último servo motor mencionado, entra em atuação quando o robô cai, e ocorrendo este evento, entra em ação e levanta novamente o robô, colocando-o em uma posição definida como alta,em pé, afim de posicionar o robô para reiniciar a locomoção bípede.

Os transdutores são os responsáveis pelo monitoramento da dinâmica do sistema, logo fornecem o estado do sistema, ou seja, o comportamento dinâmico do robô devido aos comandos de atuação nos servos motores . Desta forma, é possível monitorar a sua dinâmica via informações da posição e velocidade das juntas robóticas diretas e indiretas, assim como sinais provindos dos transdutores de força, sendo capaz de distinguir se o robô está de pé ou não . Caso simultaneamente as leituras fornecidas por estes transdutores de força forneçam dados que levem a inferir que os dois pés do robô não estão no chão simultaneamente, pode-se concluir que o robô caiu e o servo motor relativo ao braço é acionado afim de levantar o robô.
LEO2 contém um total de dez transdutores, sendo estes de dois tipos distintos: encoderes magnéticos absolutos sem contato de alta resolução e transdutores de força.

## 2. ESTRUTURA FÍSICA DO SISTEMA DE CONTROLE EMBARCADO

Para realização do controle embarcado do robô foram usados:

1. Computador
2. Placa conversora de sinais padrão de comunicação USB2DYNAMIXEL
3. Placa controladora modelo OpenCM9.04c
4. Placa de expansão OpenCM4.85exp
5. Duas placas conversoras de dados 3Mxel
6. Os servos motores Dynamixel do robô LEO2
7. Os encoderes da fabricante Reninshaw modelo RE22
8. Os transdutores de força FlexiForce da fabricante Texiskan
9. Cabo de conexão padrão USB
10. Cabo de conexão padrão de comunicação TTL,5 volts de 3 pinos
11. Cabos de conexão tipo Spox
12. Cabos de conexão padrão RS485 de quatro pinos

## 2.1.Encoderes

Os Encoderes possuem a função de monitorar as posições e velocidades das articulações do robô. LEO2 possui oito encoderes, sendo:

- Cinco referentes aos encoderes acoplados pelo fabricante em relação aos próprios eixos de saída de cada servo motor, monitorando a posição e velocidade de cada motor .Desta forma é possível obter a posição e velocidade dos joelhos, pernas e braço do robô.

- Três encoderes referentes a obtenção das velocidades e posição das articulações indiretas do robô, sendo um instalado no final do conjunto de hastes articuladas, afim de monitorar o movimento do quadril e dois solidários ao eixo de rotação da base, sendo um encoder para monitorar posições e velocidades referentes ao movimento 2D do eixo transversal do robô e outro que monitora a posição e velocidade da trajetória de percurso circular do robô.

Os encoders usados no sistema embarcado do robô LEO2 e que monitoram as juntas indiretas são do modelo RE22, é um encoder magnético rotativo compacto de alta velocidade que possui um chip em seu corpo que fornece leituras com resolução para até 13bits . Alimentado com tensão de 5 volts e operando com uma corrente de 20 mA. Possui um corpo com diâmetro de 22mm e possui capacidade de operar com velocidades angulares de até 20.000 rpm.

**Figura 4 – Encoder RE22**

## 2.2. Transdutores de Força

LEO2 contém duas células de carga que são transdutores de força, cada um instalado entre um pé do robô e o servo motor relativo a uma perna inferior . Cada célula de carga gera um sinal de tensão elétrica diretamente proporcional a pressão sofrida em sua área de contato, devida a variação de sua resistência,criando um divisor de tensão.
Esta pressão varia conforme o robô se move, podendo assim detectar a partir de suas leituras quais a posições dos pés de LEO2 em relação ao solo . Monitorando se os pés estão em contato ou não com o solo.
O transdutor de força do robô bípede LEO2 é fabricado pela Tekscan,sendo padrão FlexiForce Sensors modelo A201-1,  cuja faixa de medição é entre zero e uma lb.

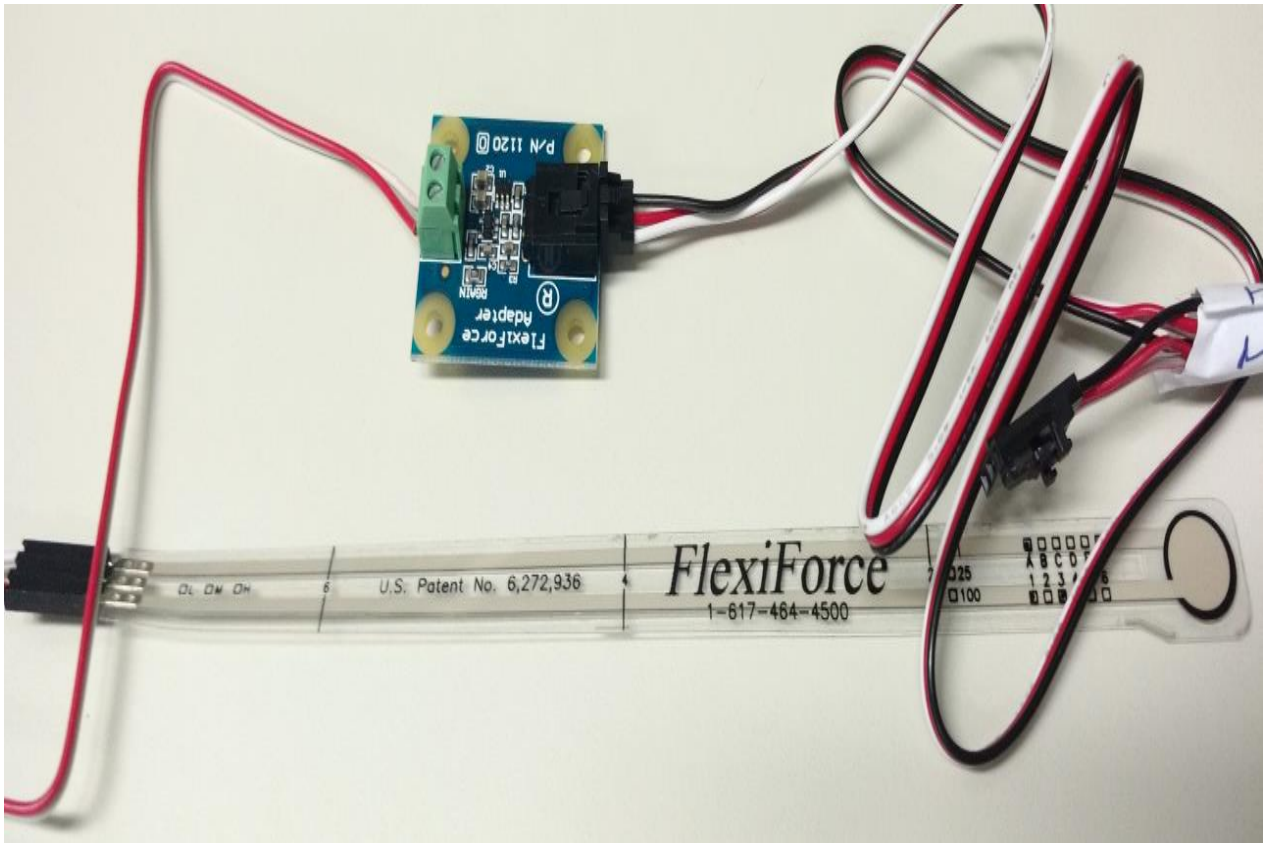


**Figura 5 – Transdutor de Força Utilizado no Robô LEO2**

## 2.3. Dynamixel

Os servos motores usados em LEO2 são da linha Dynamixel fabricados pela empresa Sul Coreana Robotis. O hardware foi abordado para realizar a dinâmica direta do robô LEO2 por ser um sistema de atuação completo, inteligente e modular . Projetados para serem acoplados mecanicamente uns aos outros, formando uma estrutura integrada de juntas de conexão exclusivas de um robô ou sistema mecânico.

O servo motor Dynamixel é uma estrutura modular completa e integrada, pois contém um motor DC, um encoder magnético absoluto sem contato, uma caixa de redução, um controlador, um driver e uma rede de comunicação interna(network).


**Figura 6 – Servo motor Mx-28**


**Figura 7 – Servo motor Mx-64**

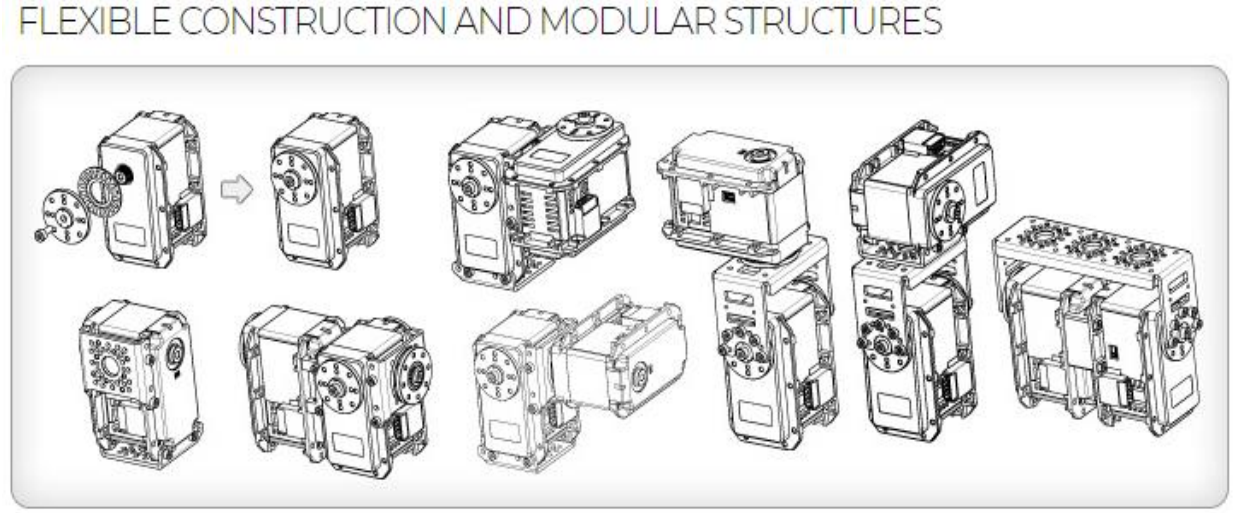


**Figura 8 – Exemplos de estruturas modulares feitas à partir da integração de vários módulos Dynamixel**


**Figura 9 – Módulo Dynamixel: Caixa de Redução, Encoder, Controlador, Driver, Rede de Transmissão de Dados**

O servo motor integrado Dynamixel possui como unidade de processamento de dados(CPU) um micro controlador interno do tipo MCU : ST CORTEX-M3 ( STM32F103C8 @ 72MHZ, 32BIT) .

Os atuadores Dynamixel se comunicam com o controlador principal de um sistema embarcado através do meio físico padrão RS-485 Multi-Drop Bus e com comunicação serial assíncrona de 8 bits, 1 bit de parada e sem paridade.

São conectados eletricamente uns aos outros na forma de encadeamento (Daisy Chain), ou seja, um servo está conectado ao controlador principal enquanto os demais são conectados uns aos outros, já

que cada Dynamixel possui duas portas de quatro pinos para de conexão com outros hardwares do sistema(BUS para RS485).

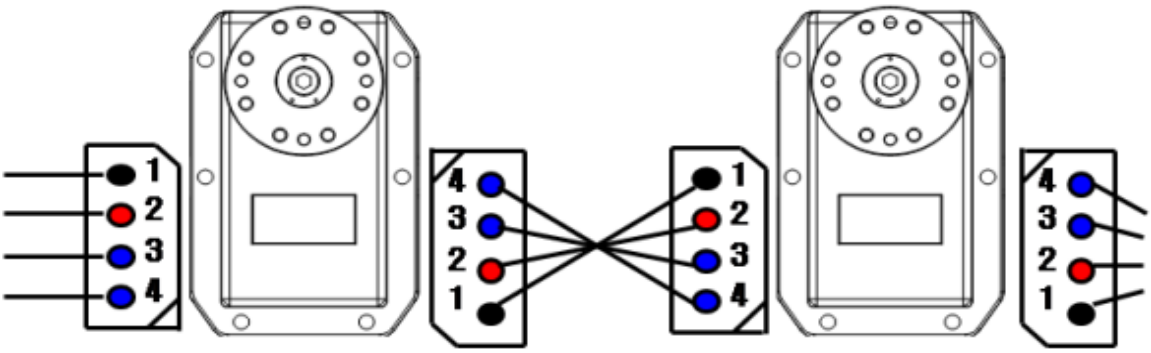


**Figura 10 – Dynamixels Conectados em Encadeamento Chain Mode**

Programável, possui um controlador de 32bits com memória para leitura e escrita, dispondo de diversas funções de monitoramento(leitura) e comando(escrita). Desta forma, o Dynamixel é capaz de fornecer seu estado, ou seja posição e velocidade em seu eixo de saída.

Possui comando para operar em rotação completa, ou seja, realizando um giro de 360 graus em torno de seu eixo(Endless Turn), comportando-se como um motor DC puro.

Possui comandos para escrita e leitura de posição. Seu encoder é do tipo absoluto magnético sem contato e produz informação confiável da posição angular de seu eixo de saída com resolução de 12 bits, ou seja, 4096 posições distintas cobrindo um ângulo de 360 graus, com uma acurácia de 0. 088 graus a cada bit.

Possui comandos de escrita e leitura de velocidade de rotação de seu eixo de saída, com uma resolução de 10 bits, ou seja, 1023 velocidades distintas com módulo entre zero e cerca de 114 rpm , tendo uma precisão de 0,11 rpm, rotacionando em ambos os sentidos, anti-horário e horário.

Possui comando para habilitar e desabilitar o torque do eixo de saída do motor, podendo também limitar a porcentagem do torque aplicado em relação ao torque máximo que o servo consegue produzir em seu eixo de saída com uma acurácia de 10bits, ou seja, 1023 passos.

Exemplificando, 1023 corresponde ao máximo torque possível a ser produzido no eixo de saída do motor e 512 configura que o eixo de saída do servo motor fornece cinquenta por cento do torque máximo.

Contém PID como algoritmo de controle, sendo os ganhos Kp, Kd e Ki ajustáveis . Desta forma o driver do Dynamixel possui uma saída mais confiável, pois o algoritmo de controle PID visa filtrar efeitos de possíveis perturbações no sistema mecânico provindos da caixa de redução do servo motor.

A taxa de transmissão de dados entre o controlador principal e o Dynamixel pode ser configurada para operar entre 8000 bps e 4,5Mbps.

## 2.4. Protocolo de Comunicação Dynamixel

O controlador principal (Master) se comunica com as unidades Dynamixel (Slaves) através de um protocolo de comunicação fornecido pelo fabricante . O controlador transmite e recebe sinais de forma serial e assíncrona através de um pacote de dados previamente implementado pelo fabricante.

Estes pacotes de dados são de dois tipos:

- Pacote de instruções (enviados do controlador principal e recebido pelos Dynamixels)
- Pacote de estado (enviados dos pelos Dynamixels e recebidos pelo controlador principal)

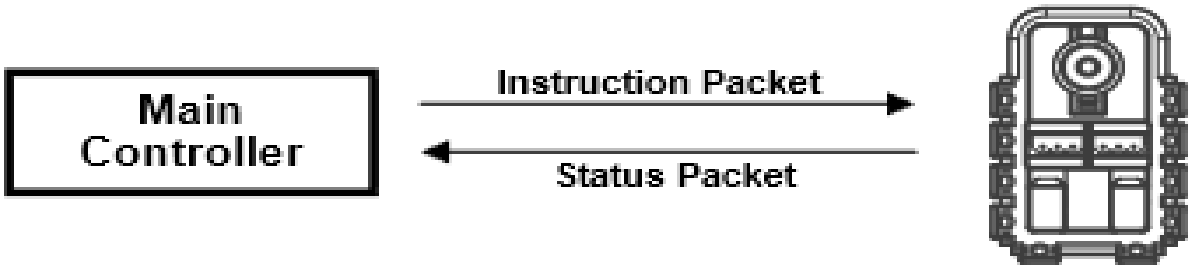


**Figura 11 – Comunicação Controlador Principal e Dynamixels**

É possível atribuir a cada motor uma identidade podendo enviar comandos de atuação provindos do controlador principal (Master) e realizar leituras de estado relativo a um determinado motor (Slave) em questão.

No exemplo abaixo, caso o controlador principal transmita um pacote de instruções com a identidade do Dynamixel configurada com identidade 'N', apenas o módulo Dynamixel cuja identidade é configurada como 'N' irá responder,,ou seja,irá processar os dados do comando do pacote de instrução transmitido pelo controlador e irá retornar para o controlador dados de seu respectivo pacote de estado.

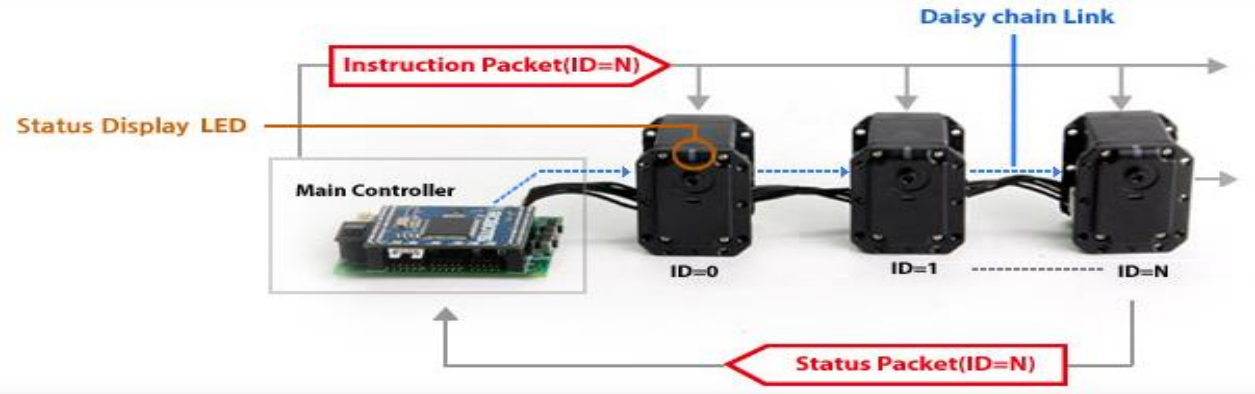


**Figura 12 – Pacote de Instrução e Resposta Relativo ao Dynamixel Identidade 'N'**

## 2.5. Tabela de Controle Dynamixel

As informações sobre o funcionamento e estado do módulo Dynamixel estão contidas em sua Tabela de Controle. O atuador é operado conforme funções de escrita e de leitura, que acessam na memória de seu controlador interno,o endereço de seus registradores. Desta forma o atuador é operado escrevendo valores nas áreas de memória de seus registradores definidas para escrita e o seu estado(monitoramento) é obtido lendo valores de áreas de memória reservadas para leitura, conforme sua tabela de controle.

| Area | Address (Hexadecimal) | Name | Description | Access | Initial Value (Hexadecimal) |
|---|---|---|---|---|---|
| EEPROM | 0 (0X00) | Model Number(L) | Lowest byte of model number | R | 29 (0X1D) |
| | 1 (0X01) | Model Number(H) | Highest byte of model number | R | 0 (0X00) |
| | 2 (0X02) | Version of Firmware | Information on the version of firmware | R | - |
| | 3 (0X03) | ID | ID of Dynamixel | RW | 1 (0X01) |
| | 4 (0X04) | Baud Rate | Baud Rate of Dynamixel | RW | 34 (0X22) |
| | 5 (0X05) | Return Delay Time | Return Delay Time | RW | 250 (0XFA) |
| | 6 (0X06) | CW Angle Limit(L) | Lowest byte of clockwise Angle Limit | RW | 0 (0X00) |
| | 7 (0X07) | CW Angle Limit(H) | Highest byte of clockwise Angle Limit | RW | 0 (0X00) |
| | 8 (0X08) | CCW Angle Limit(L) | Lowest byte of counterclockwise Angle Limit | RW | 255 (0XFF) |
| | 9 (0X09) | CCW Angle Limit(H) | Highest byte of counterclockwise Angle Limit | RW | 15 (0X0F) |
| | 11 (0X0B) | the Highest Limit Temperature | Internal Limit Temperature | RW | 80 (0X50) |
| | 12 (0X0C) | the Lowest Limit Voltage | Lowest Limit Voltage | RW | 60 (0X3C) |
| | 13 (0X0D) | the Highest Limit Voltage | Highest Limit Voltage | RW | 160 (0XA0) |
| | 14 (0X0E) | Max Torque(L) | Lowest byte of Max. Torque | RW | 255 (0XFF) |
| | 15 (0X0F) | Max Torque(H) | Highest byte of Max. Torque | RW | 3 (0X03) |
| | 16 (0X10) | Status Return Level | Status Return Level | RW | 2 (0X02) |
| | 17 (0X11) | Alarm LED | LED for Alarm | RW | 36 (0X24) |
| | 18 (0X12) | Alarm Shutdown | Shutdown for Alarm | RW | 36 (0X24) |
| | 20 (0X14) | Multi Turn Offset(L) | multi-turn offset least significant byte (LSB) | RW | 0 (0X00) |
| | 21 (0X15) | Multi Turn Offset(H) | multi-turn offset most significant byte (MSB) | RW | 0 (0X00) |
| | 22 (0X16) | Resolution Divider | Resolution divider | RW | 1 (0X01) |
| RAM | 24 (0X18) | Torque Enable | Torque On/Off | RW | 0 (0X00) |
| | 25 (0X19) | LED | LED On/Off | RW | 0 (0X00) |
| | 26 (0X1A) | D Gain | Derivative Gain | RW | 0 (0X00) |
| | 27 (0X1B) | I Gain | Integral Gain | RW | 0 (0X00) |
| | 28 (0X1C) | P Gain | Proportional Gain | RW | 32 (0X20) |
| | 30 (0X1E) | Goal Position(L) | Lowest byte of Goal Position | RW | - |
| | 31 (0X1F) | Goal Position(H) | Highest byte of Goal Position | RW | - |
| | 32 (0X20) | Moving Speed(L) | Lowest byte of Moving Speed (Moving Velocity) | RW | - |
| | 33 (0X21) | Moving Speed(H) | Highest byte of Moving Speed (Moving Velocity) | RW | - |
| | 34 (0X22) | Torque Limit(L) | Lowest byte of Torque Limit (Goal Torque) | RW | ADD14 |
| | 35 (0X23) | Torque Limit(H) | Highest byte of Torque Limit (Goal Torque) | RW | ADD15 |
| | 36 (0X24) | Present Position(L) | Lowest byte of Current Position (Present Velocity) | R | - |
| | 37 (0X25) | Present Position(H) | Highest byte of Current Position (Present Velocity) | R | - |
| | 38 (0X26) | Present Speed(L) | Lowest byte of Current Speed | R | - |
| | 39 (0X27) | Present Speed(H) | Highest byte of Current Speed | R | - |
| | 40 (0X28) | Present Load(L) | Lowest byte of Current Load | R | - |
| | 41 (0X29) | Present Load(H) | Highest byte of Current Load | R | - |
| | 42 (0X2A) | Present Voltage | Current Voltage | R | - |
| | 43 (0X2B) | Present Temperature | Current Temperature | R | - |
| | 44 (0X2C) | Registered | Means if Instruction is registered | R | 0 (0X00) |
| | 46 (0X2E) | Moving | Means if there is any movement | R | 0 (0X00) |
| | 47 (0X2F) | Lock | Locking EEPROM | RW | 0 (0X00) |
| | 48 (0X30) | Punch(L) | Lowest byte of Punch | RW | 0 (0X00) |
| | 49 (0X31) | Punch(H) | Highest byte of Punch | RW | 0 (0X00) |
| | 73 (0X49) | Goal Acceleration | Goal Acceleration | RW | 0 (0X00) |

**Figura 13 – Tabela de Controle Dynamixels Modelos Mx-28 e Mx-64**

Acessando os endereços de escrita da tabela de controle do Dynamixel, é possível operar em diferentes modos. Há registradores para limite de ângulo de rotação(Endereços 0x06 e 0x07), limitador de torque(0x0E e 0x0F) e para ajuste o ganho do algoritmo de controle PID do controlador(0x1A, 0x1B, 0x1C), conforme a escrita em seus registradores de ganho.

Também é possível obter diversos dados de seu operacionamento em tempo real como posição (0x24 e 0x25) e velocidade (0x26 e 0x27), temperatura interna do sistema (0x2B), taxa de transmissão de dados(0x04), identidade do atuador em questão(0x03) e voltagem de operação do motor(0x29 e 0x2A).

Modos de operação principal usados no sistema embarcado do robô LEO2:

- Position Mode:

À partir de uma posição inicial, realiza o movimento para uma nova posição desejada. Geralmente usado em robores com múltiplas juntas, ou seja, vários graus de liberdade, onde cada junção possui ângulos de restrição para operação . Neste modo, não há controle sobre velocidade, apenas posição, indo de uma posição inicial até um determinado setpoint, sendo aplicada a máxima velocidade do motor durante o percurso entre a posição atual e desejada. Neste modo de operação, a acurácia do posicionamento do eixo de saída dos servos motores é otimizado pelo algoritmo PID de controle de seu controlador interno.

- Wheel Mode:

Neste modo de operação o eixo de saída do Dynamixel possui rotação livre de 360 graus, sem limite, ou seja, o atuador opera como um motor DC. É possível à partir de uma posição, se movimentar no sentido de uma direção do movimento circular, horária ou anti-horária, com uma velocidade controlada . Ou seja, podemos configurar a movimentação do eixo de saída do servo motor com uma velocidade desejada de deslocamento em sentido de uma direção desejada.

Registradores de escrita e leitura de dados usados na programação do software do controlador principal do sistema de controle embarcado do robô LEO2:

- Torque Enable:

Neste modo de operação, é possível habilitar e desabilitar a produção de torque no eixo de saída do servo motor.

Também é possível monitorar e obter informações em tempo real através de funções de leitura aplicadas em seus registradores, conforme a tabela de controle:

- PRESENT POSITION:

A leitura deste registrador fornece a posição do eixo de saída do servo motor com uma resolução de 12bits.

- PRESENT SPEED:

A leitura deste registrador fornece a velocidade e o sentido do eixo de rotação do motor com uma resolução de 10bits.

O robô bípede LEO2 utiliza cinco servos motores da linha Dynamixel, porém dois modelos distintos cada um com uma especificação.

As pernas superiores do robô LEO2 são servos motores Dynamixel do modelo Mx-64, enquanto as pernas inferiores e o quadril são do modelo Mx-28. Esta escolha foi feita devido a exigência de maior

torque a ser produzido pelas articulações das pernas superiores em relação as articulações das pernas inferiores na dinâmica bípede do caminhar humano.

O motor Mx-64 é mais robusto, pesado e maior que o Mx-28, porém possui uma capacidade de produzir em seu eixo de saída um torque de parada (stall torque) máximo de 6 Nm quando alimentado com 12 Volts, enquanto o Mx-28 produz um torque de parada máximo em seu eixo de saída de 2,5 Nm.

Data

| | Unit | Data |
|---|---|---|
| Weight | g | 77 |
| Dimension | mm | 35.6 x 50.6 x 35.5 |
| Gear Ratio | type/material | 193:1 (Spur/Metal) |
| Network | - | TTL / RS-485 |
| Position Sensor (resolution) | - | Contactless Absolute Encoder (360°/ 4096) |
| Motor | - | Maxon Motor |
| Operation Voltage | V | 10~14.8 |
| Stall Torque | N.m | 2.5 at 12V |
| Stall Current | A | 1.4 at 12V |
| No Load Speed | RPM | 55 at 12V |

**Figura 14 – Especificações Mx-28**

Data

| | Unit | Data |
|---|---|---|
| Weight | g | 135 |
| Dimension | mm | 40.2 x 61.1 x 41 |
| Gear Ratio | type/material | 200:1 (Spur/Metal) |
| Network | - | TTL / RS-485 |
| Position Sensor (resolution) | - | Contactless Absolute Encoder (360°/ 4096) |
| Motor | - | Maxon Motor |
| Operation Voltage | V | 10~14.8 |
| Stall Torque | N.m | 6.0 at 12V |
| Stall Current | A | 4.1 at 12V |
| No Load Speed | RPM | 63 at 12V |

**Figura 15 – Especificações Mx-64**

## 2.6. Controlador Principal do Sistema Placa OpenCM9.04c

O controlador principal usado no sistema de controle embarcado do robô LEO2 é o modelo OpenCM9.04c, da mesma fabricante do servo motor integrado Dynamixel , Robotis. Projetado para ser o controlador principal para servos motores da linha Dynamixel, é baseado na tecnologia 32bit ARM Cortex-M3 CPU.

Possui uma porta Micro-B USB, possibilitando a sua conexão com um computador via um cabo USB para que seja programada e para fornecer dados ao computador à partir desta porta . Quando conectada pela porta USB a um computador, a placa é suprida com uma alimentação de 5volts.

Possui clock interno de 72Mhz , uma CPU STM:32F103CB(ARM Cortex-M3), dimensões de 27x66, 5mm e um peso de 13gramas.

É uma placa controladora de código aberto, ou seja, pode embarcar o ambiente de programação de qualquer controlador para ser programada.

No caso da programação do ambiente da placa no software de controle embarcado foi usado a IDE do controlador Arduino, assim como a visualização dos dados de comunicação e processamento foram obtidos e analisados pela interface monitor serial desta IDE.

Possui portas de três pinos para comunicação serial em nível TTL e uma porta de quatro para de comunicação assíncrona serial cujo meio físico padrão é RS-485.

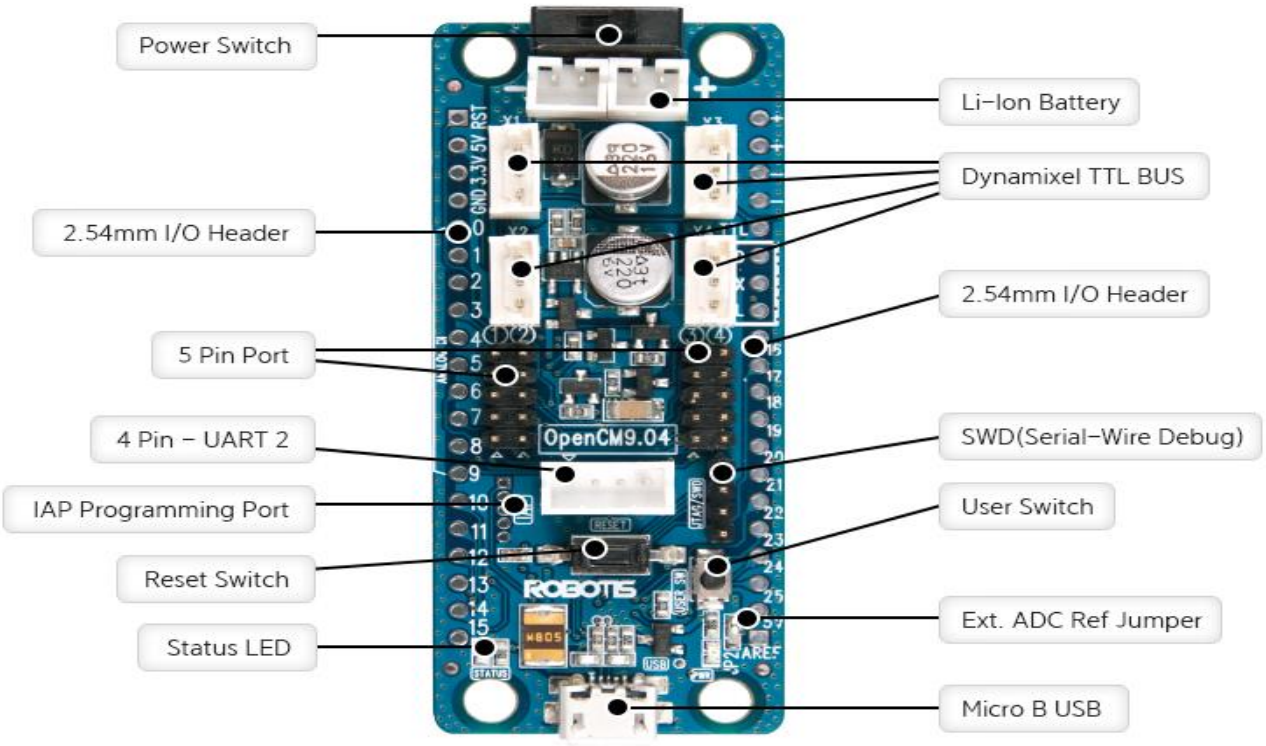


**Figura 16 – Placa OpenCM9.04c**

A figura abaixo exemplifica o modo de conexão do canal de programação da placa OpenCM9.04c e um computador,via meio físico USB.

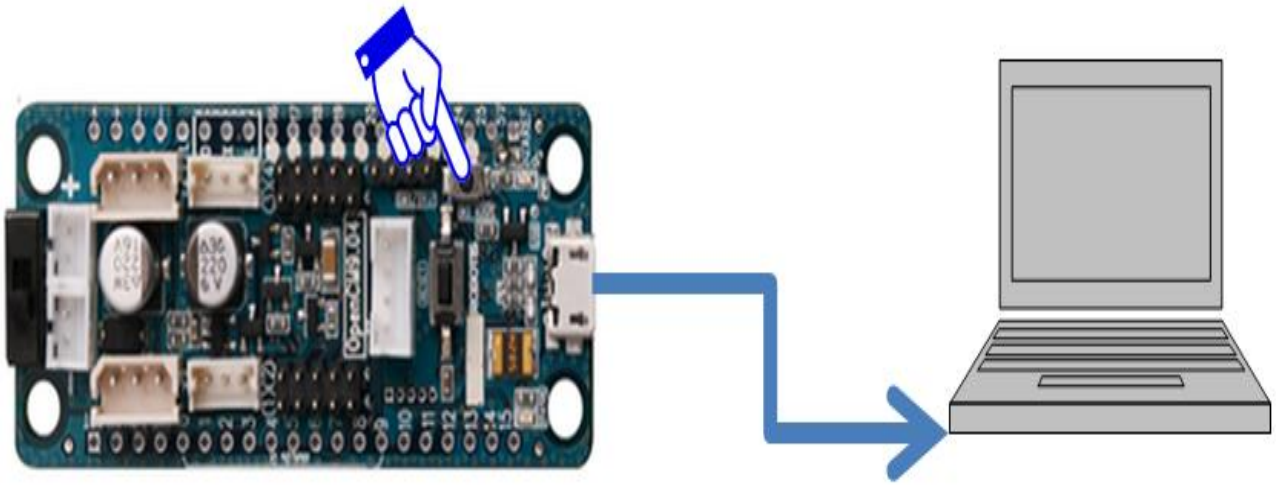

**Figura 17 – Modelo de Conexão Via USB entre Controlador Principal e Computador**

## 2.7. Placa Expansora de Conexões OpenCM4.85

A placa OpenCM4.85 é uma placa de expansão que é acoplada ao controlador OpenCM9.04c. Possui um BUS com quarto portas de conexão com 4 pinos para padrão RS-485. Conecta-se as duas placas conversoras 3Mxel em duas destas portas e ao menos um Dynamixel é conectado em outra devido ao fato de poderem serem conectados uns aos outros em modo de encadeamento.

Possui uma possui chave de liga e desliga usada tanto para a OpenCM4.85 quanto para os Dynamixel e placas conversoras 3Mxel conectados em seu BUS.

Contém três LEDs digitais e dois botões digitais que podem ser programados através de comandos da IDE adotada para o controlador.

**Figura 18 – Placa OpenCM9.04c e Placa OpenCM4.85 Integradas**

## 2.8. Placa de Aquisição de Dados 3Mxel

Todos os transdutores do sistema embarcado do robô LEO2, encoders não intrínsecos aos Dynamixel , e os transdutores de força são conectados diretamente a uma placa de conversora de dados 3Mxel.
Esta placa possui a capacidade para o acoplamento de dois encoderes e dois transdutores de força, pois possui duas portas para conexão de encoderes e 2 portas para conexão dos transdutores de força, logo o sistema embarcado do robô LEO2 possui duas placas 3Mxel.
A placa 3Mxel aquisita o sinal do encoder e depois o processa a níveis do sistema para que sejam aquisitados pelo controlador.
O meio físico de comunicação entre a placa 3Mxel e o controlador principal do sistema é RS-485 e a transmissão de dados assíncrona serial.
A placa 3Mxel utiliza o mesmo protocolo de comunicação Dynamixel, e se comunica com o controlador principal da mesma forma, conforme o protocolo Dynamixel, recebendo pacote de instruções e transmitindo pacote de estado . Contém registradores próprios, ou seja, uma expansão da tabela de controle do Dynamixel.

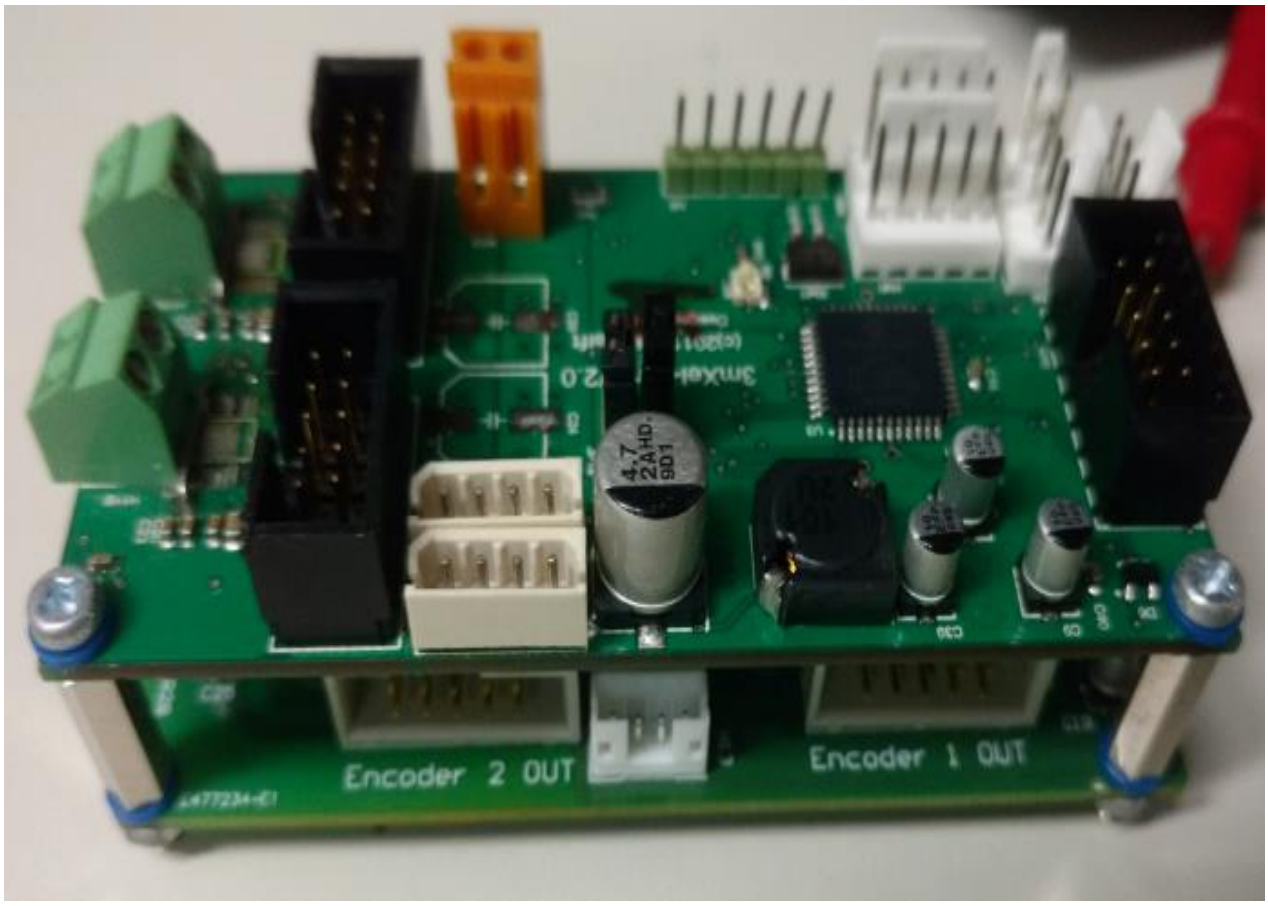


**Figura 19 – Placa de Aquisição de Dados dos Transdutores do Sistema 3Mxel**

## 2.9. Placa de Conversão de Sinais USB2DYNAMIXEL

A placa USB2DYNAMIXEL é o meio físico de interface de comunicação de dados entre o computador e os módulos Dynamixel ou entre o computador e o controlador principal do sistema embarcado. No caso do sistema de controle embarcado do robô bípede LEO2, USB2DYNAMIXEL é a interface física de comunicação entre o computador e a placa controladora principal OpenCM9.04c.

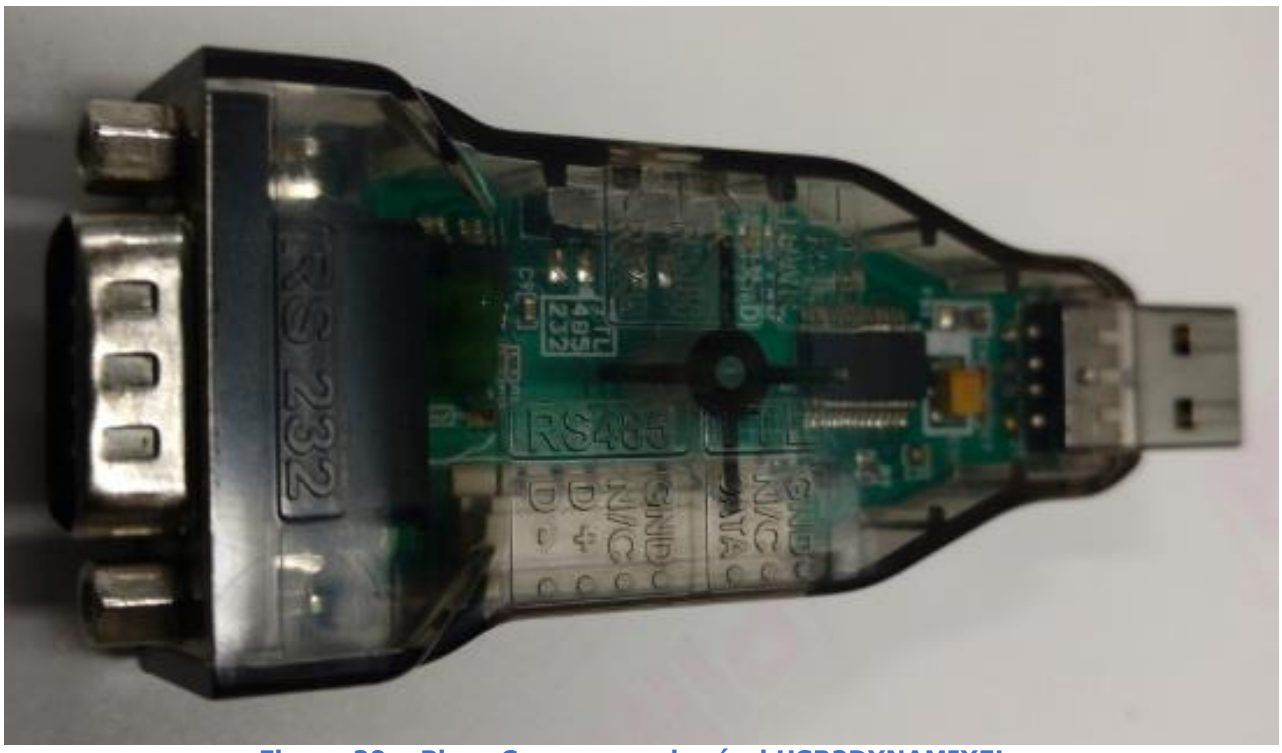


**Figura 20 – Placa Conversora de nível USB2DYNAMIXEL**

Possui um Chip FTDI que realiza a conversão de dados do tipo USB provindos do computador para serial, afim de comunicar o computador com o controlador, já que a transmissão de dados entre os componentes do sistema embarcado de LEO2 é assíncrono serial.

Possui uma entrada para nível USB e três saídas, sendo uma saída de três pinos de conexão nível TTL, 0 Volt a 5 Volts, uma saída de 4 pinos de conexão padrão RS-485 e outra saída de 9 pinos para conector serial padrão RS-232.
De acordo com o posicionamento de uma pequena alavanca, escolhe-se um padrão de comunicação entre o computador e o controlador de saída, nível TTL, RS-485 ou RS-232.

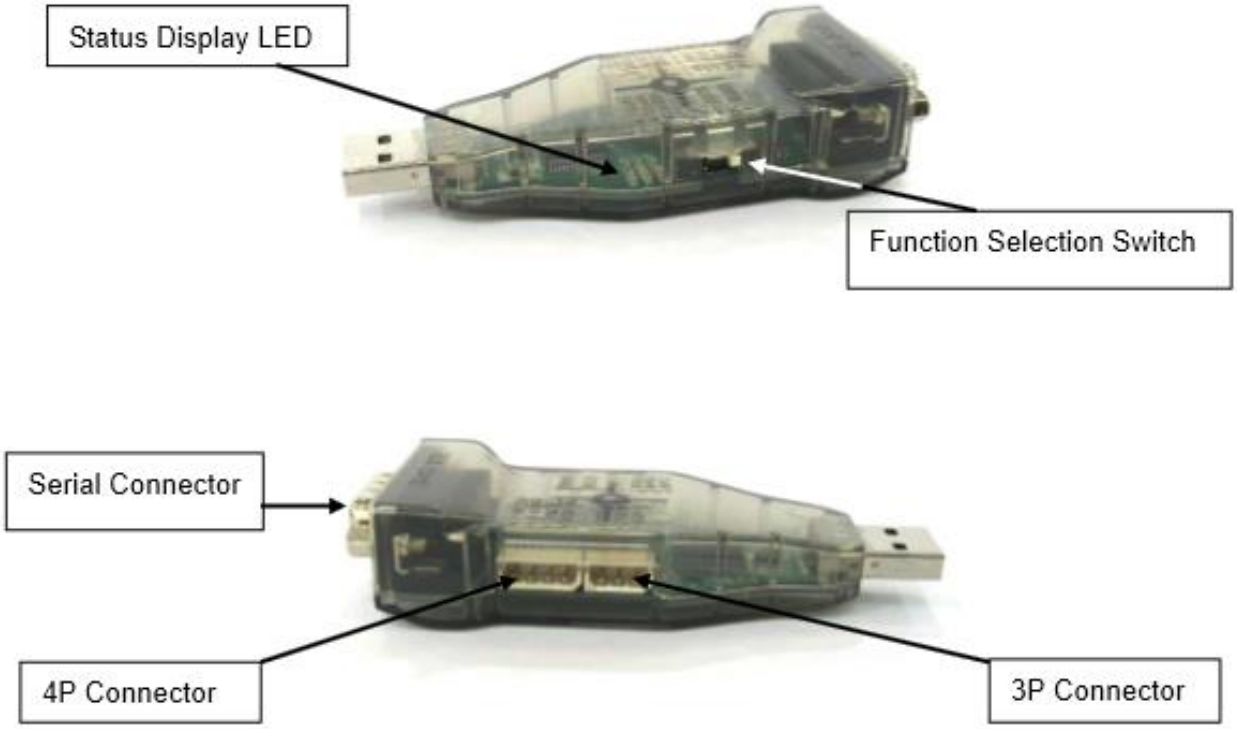


**Figura 21 – Placa Conversora USB2DYNAMIXEL com Enfase na Chave de Seleção e Conexões**

O sistema de controle embarcado de LEO2 usa a saída nível TTL, ou seja, entre 0 volts e 5 volts, da placa USB2DYNAMIXEL. A conexão com o controlador principal é realizado através de um cabo modelo Spox que se conecta aos 3 pinos de saída desta porta . O outro lado do cabo é conectado a placa controladora principal OpenCM9.04c em um canal de seu BUS para sinais em nível TTL de 3 pinos, sendo usados os pinos de transmissão e recepção Tx1 e Rx1. Conforme o manual de instruções deste controlador, estes são os pinos para comunicação em serial em níveis TTL(LEDS Rx - Tx).

# 3. SOFTWARE DESENVOLVIDO

## 3.1. Ambientes de Programação e Fluxo de Dados

A transmissão e comunicação de dados do sistema de controle embarcado do robô LEO2 é implementada computacionalmente por um software desenvolvido para este fim . O software estabelece a comunicação serial de dados entre os ambientes que o compõe com uma determinada taxa de comunicação. Enviando comandos de atuação para os servos motores do robô e recebendo dados de monitoramento de seus transdutores em tempo real . Processa estes dados conforme uma lei de controle proporcional, estabelecendo desta forma a dinâmica do robô.
O software computacional do sistema é subdividido em três ambientes de distintos.

- Ambiente do Computador
- Ambiente do controlador principal
- Ambiente dos Dynamixel e de monitoramento.

Foi estabelecido primeiramente uma taxa de transmissão de dados de 57600bps e posteriormente, esta taxa foi modificada para 1Mbps.

O fluxo de dados do sistema se dá na seguinte forma:

1. O computador transmite comandos de termos de controle para o controlador principal através de um protocolo de comunicação implementado.
2. O controlador por sua vez recebe estes comandos e os envia para os atuadores Dynamixel conforme o protocolo de comunicação do fabricante.
3. Os Dynamixels por possuirem um controlador interno enviam um pacote de estado para o controlador principal, conforme o protocolo de comunicação do fabricante.
4. O controlador envia os dados de estado para o computador conforme o protocolo de comunicação implementado .
5. O ciclo de transmissão de dados se reinicia.

O software implementado possui dois programas rodando simultaneamente em paralelo:

- Programa do Computador
- Programa do controlador principal

O Programa do , implementado nas linguagens de programação C e C++, foi programado no ambiente LINUX UBUNTU versão 16. 04 e realiza o controle da dinâmica do sistema. O programa do Computador é a interface entre o usuário do sistema e o robô LEO2.

O programa do controlador foi implementado no ambiente da IDE do Arduino, já que o controlador principal do sistema embarcado OpenCM9.04c é uma placa de código aberto, ou seja, embarca qualquer ambiente de programação para controladores.

## 3.2. Modos de Operação do Sistema

O software possibilita que o sistema opere em três modos distintos, dependendo da natureza da variável de controle:

Modo de posicionamento (Malha Aberta):

Não há controle sobre velocidade ou torque na dinâmica do robô. As posições finais dos eixos de saída dos servos motores são alcançadas à partir de suas posições iniciais com velocidade máxima e torque máximo de cada motor.

Controle de velocidade (Malha Fechada):

Há o controle da velocidade dos eixos de saída dos servos motores. A velocidade do eixo de saída dos servos motores do robô vão variando conforme vão se aproximando ou se afastando das suas posições finais desejadas, conforme uma lei de controle proporcional de realimentação de estados.

Controle de torque (Malha Fechada):

Há o controle do torque dos eixos de saída dos servos motores. É enviado um comando de velocidade máxima aos atuadores em relação a um sentido de rotação e limita-se o torque dos servos motores. Desta forma, os servos motores tentam atingir a velocidade máxima, porém não conseguem alcançar a velocidade máxima e produzem o torque desejado.

## 3.3. Protocolo de Comunicação Desenvolvido

O Programa do Computador ,implementado nas linguagens de programação C e C++, foi programado no ambiente LINUX UBUNTU versão 16. 04 e realiza o controle da dinâmica do sistema. O programa do Computador é a interface entre o usuário do sistema e o robô LEO2.

O programa do controlador foi implementado no ambiente da IDE do Arduino, já que o controlador principal do sistema embarcado OpenCM9.04c é uma placa de código aberto, ou seja, embarca qualquer ambiente de programação para controladores.

O ambiente do computador se comunica com o ambiente do controlador principal transmitindo dados seriais de comando para a placa controladora e recebendo dados seriais de estado provindos da mesma através de um protocolo de comunicação estabelecido afim de obter uma interface de comunicação entre estes dois ambientes. O protocolo foi implementado através de pacotes de dados que contém em seus campos de informação os dados seriais que tanto o ambiente do computador quanto ambiente da placa controladora principal reconheçam e sejam capazes de processar.

As estruturas de dados implementadas conforme o protocolo de comunicação que ditam a interface entre os ambientes do computador e do controlador principal são:

- leo2_command_package_t: Pacote de Comando

Pacote de dados referente aos comandos de atuação dos servos motores que o ambiente do computador transmite para o controlador e o ambiente do controlador por sua vez o reconhece e o recebe do computador.

Possui dois campos de reconhecimento de um pacote de dados de comando, os bytes de headers, se referindo como o cabeçalho do pacote, designados como 255 em base decimal, ou seja , 0XFF em hexadecimal.

Possui um campo para designar o modo de atuação do sistema.

Contém um campo cujo valor é referente ao dobro da quantidade de servos motores conectados ao sistema, já que muitos dos comandos de escrita ou leitura em cada módulo Dynamixel acessam dois endereços de registrador, representados pelo byte de alto e byte baixo.

Possui um campo com os comandos a serem implementados pelos servos motores do sistema, que é um vetor de bytes (buffer) em que cada dois elementos consecutivos se refere a um servo motor em questão. Varia de tamanho de acordo com o dobro quantidade de servos motores em uso no robô.

Por último há também um byte de validade, byte checksum, que é o complemento do somatório de todos os bytes do pacote de comandos exclusive a si próprio.

Pacote de dados de comando conforme o protocolo de comunicação estabelecido:

1. Byte header_1 = 0xFF;
2. Byte header_2 = 0XFF;
3. Byte command;
4. Byte length [2*NÚMEROS DE ATUADORES];
5. Vetor de bytes de dados de comando [2*NÚMEROS DE ATUADORES];
6. Byte checksum;

- leo2_status_package_t:

Este pacote de dados contém os dados de monitoramento do sistema, ou seja, seu estado completo, posição e velocidade dos encoderes e leituras de tensão dos transdutores de força do robô LEO2.

Possui dois campos de reconhecimento de um pacote de dados, os bytes de headers, designados como 255 em base decimal e 0xFF em hexadecimal .

Possui um campo que indica o estado atual do robô:

- Estado de Erro
- Estado Normal

Possui um campo cujo valor é referente ao dobro da quantidade de transdutores conectados ao sistema, devido ao protocolo Dynamixel acessar dois registradores para obter uma informação relativa a cada variável de estado fornecida por um transdutor.

Possui três campos em que cada um é um vetor de bytes com os dados.
O pacote de dados de estado possui um vetor de bytes referente a leitura das posições dos encoderes, um vetor de bytes com os dados de velocidades angulares dos encoderes e outro vetor com as informações em hexadecimal referente a leitura de tensão dos transdutores de força.

Possui um campo de validade do pacote de estado, o byte checksum, corresponde ao complemento do somatório de todos os bytes do pacote excludente a si mesmo.

Pacote de dados de estados conforme o protocolo de comunicação:

- Byte header_1 = 0xFF;
- Byte header_2 = 0xFF;
- Byte status;
- Byte length[4 * NÚMERO DE ENCODERES + 2 * NÚMERO DE TRANSDUTORES DE FORÇA];
- Vetor de bytes de dados de posições dos encoderes[2 * NÚMERO DE ENCODERES];
- Vetor de bytes de dados de velocidades dos encoderes [2 * NÚMERO DE ENCODERES];
- Vetor de bytes de tensões dos transdutores de força[2 * NÚMERO DE TRANSDUTORES DE FORÇA];
- Byte checksum;

Em ambas as estruturas de dados definidas para implementar o pacote de comunicações entre os ambientes de programação do computador e do controlador principal, pacote de comando e pacote de estado, possuem um campo denominado length, que recebe um byte referente ao comprimento dos dados de atuação(pacote de comando) ou estados(pacote de estados). Como cada servo motor ou transdutor do robô LEO2 aborda o pacote de comunicação do fabricante Dynamixel para transmitir ou receber dados, há dois bytes referentes a cada comando de atuação (posição, velocidade ou torque) ou de monitoramento(posição, velocidade e tensão). O campo length da estrutura de dados pacote de comando implementado possui como valor um byte referente a duas vezes o número de atuadores enquanto o campo length da estrutura de dados pacote de estado possui como valor um byte referente a duas vezes a quantidade de encoderes do sistema somado a quantidade de transdutores de força.

## 3.4. Ambiente de Programação do Computador

O ambiente do Computador possui duas estruturas de dados cujos campos são comandos e estados do robô em unidades de grandeza do sistema internacional:graus referente a posição ângular, radianos por segundo referente a velocidade angular dos encoderes e voltagem referentes aos transdutores de força do robô.
Recebe pacotes de estados provindos do ambiente do controlador principal, conforme o protocolo de comunicação estabelecido entre os dois ambientes e verifica sua validade e integralidade.

Caso haja algum erro de comunicação de dados neste processamento, é verificado e realizado um tratamento de erro relativo a cada situação possível:

- Erro de abertura da interface física FTDI, que realiza a conversão de dados do ambiente do computador em padrão de comunicação USB para padrões de comunicação serial em nível TTL, afim de realizar a comunicação do computador com o controlador OpenCM9.04c da forma assíncrona e serial.

- Erro de estabelecimento de taxa de transmissão entre o computador e a placa controladora. Ou seja, a taxa de transmissão de dados na interface de comunicação entre o computador e o controlador não é igual a estabelecida.

- Erro devido a integralidade do pacote de estado, ocorre quando um pacote de estados é recebido pelo ambiente do computador e o tamanho deste pacote é diferente do pacote estabelecido pelo protocolo de comunicação implementado.

- Erro devido a validade do cabeçalho do pacote de estado recebido pelo ambiente do computador, ocorre quando o computador recebe um pacote e é feita a leitura dos bytes de header dos pacotes, ou seja, estes são diferentes do seu valor estabelecido 0xFF.

- Erro de validade devido a verificação do byte de checksum. Ocorre quando ao receber um pacote de estados transmitido pela placa controladora, o programa do computador verifica que o resultado do complemento do somatório de todos os bytes excludentes o byte de checksum não condiz com o byte de checksum do pacote recebido. Detectando o erro de validade do pacote de dados.

Quando há algum erro de validade ou integralidade no recebimento de um pacote de estados, pelo ambiente de programação do computador, este pacote é descartado.

Caso não haja erros de transmissão de informação, as informações seriais de estado provindas da placa são recebidas pelo ambiente do computador em um pacote de estado implementado que é convertido e alocado em uma estrutura de dados de estado do ambiente do computador.

Os dados em hexadecimal do pacote de estados implementado é convertido para unidades de grandeza do sistema decimal e preenchem uma estrutura de estado do computador. A estrutura de dados de estado do ambiente do computador possui um campo referente ao estado atual do sistema e outros três campos que são vetores de dados de posição e velocidade dos encoderes e tensão fornecida pelo transdutor de força em unidades do sistema internacional de grandezas físicas.

Caso o modo de operação do sistema seja de posicionamento (malha aberta), a estrutura de dados de comando do ambiente do computador(sistema internacional) é convertida para o sistema hexadecimal e preenche um pacote de comandos conforme o protocolo de comunicação estabelecido entre o computador e o controlador principal e é transmitido de forma serial do ambiente do computador para o ambiente do controlador.

Caso o modo de operação seja de malha fechada, operação em controle de velocidade ou controle de torque, os dados da estrutura de comando do ambiente do computador são provindos de sua estrutura de controle incorporada em seu ambiente.
A estrutura de controle implementa um controle proporcional, possui um campo referente ao ganho do algoritmo e um campo que é um vetor de dados referentes as posições desejadas dos atuadores(setpoints).

Uma função de controle integra os parâmetros da estrutura de dados de controle(Ganho e Setpoints) com o estado do sistema(posição e velocidade de cada encoder e voltagem em cada célula de carga). Preenche os campos de dados da estrutura de comando do computador, conforme a lei de controle U:= Gp *(Setpoint – Posição atual).

A estrutura de comando do ambiente do computador é convertida para um pacote de dados de comando, conforme o protocolo de comunicação que estabelece a interface entre os ambientes do computador e da placa controladora do sistema.
O computador transmite de forma serial o pacote de comando para o controlador.

## 3.5. Ambiente de Programação do Controlador Principal

O ambiente de programação do controlador principal do sistema embarcado é a IDE do controlador Arduino. O programa do controlador implementa a máquina de estado. Pois o controlador é o responsável pelo envio de comandos aos atuadores Dynamixel e pela leitura de estado dos Dynamixel, encoderes e transdutores de força, conforme o protocolo de comunicação fornecido pelo fabricante e sua tabela de controle.

O controlador do sistema embarcado, OpenCM9.04c se comunica com o computador através do protocolo de comunicação implementado. O computador transmite um pacote de comandos de dados e a placa o reconhece e o recebe.

O programa do controlador verifica se o pacote de comandos está sendo recebido integralmente e se esta é valido.
Caso haja algum possível erro de processamento de informação durante a comunicação ao receber um pacote de comandos, a placa controladora realiza os tratamentos de dados relativo ao erro em questão:

Erro de tempo de comunicação durante a transmissão pelo ambiente do computador e a recepção pelo ambiente da placa de um pacote de comando. Ocorre quando a placa controladora recebe um pacote de comando do computador e o tempo de demora da transmissão deste pacote é maior que um período estabelecido. Este período foi estabelecido com base no tempo de transmissão de um pacote de estados do ambiente do computador e recepção do ambiente da placa, conforme a taxa de transmissão de dados desta interface e o tamanho do pacote de comando. Quando o tempo de transmissão e recepção de um pacote de estado através da interface de comunicação entre computador e placa controladora é maior que o estabelecido, o dado lido pelo controlador é descartado.

Erro de validade do pacote de dado recebido. Ocorre quando algum dado do cabeçalho do pacote, os bytes de headers, é incondizente com o estabelecido, ou seja, o valor 0XFF. O ambiente do computador também descarta este dado recebido caso se encontre estas incondizencias.

Também há o erro de validade do pacote quando o resultado do complemento do somatório de todos os bytes do pacote excludente o byte de checksum seja diferente do byte de checksum do pacote de comando recebido.

Erro de validade de recebimento de uma pacote de estados dos módulos Dynamixel e transdutores conectados a placa 3Mxel. O controlador principal recebe um pacote de estado conforme o protocolo Dynamixel e converte este protocolo do fabricante para um pacote de estado conforme o protocolo de comunicação estabelecido entre o computador e a placa. Durante este processo de conversão e preenchimento dos campos do pacote de estados, o byte de validade, checksum, é preenchido. Após este preenchimento é realizada uma operação que fornece como resultado o complemento do somatório de todos os bytes do pacote de estados. Caso este resultado esteje incondizente com o byte de validade, checksum, este erro é apenas visualizado através da interface serial da IDE do arduino, Serial Monitor, sendo possível detectar o ocorrimento deste tipo de erro de transmissão de dados.

Caso o pacote de comando tenha sido recebido integralmente e validado, o programa da placa controladora converte o pacote de comando provindo do computador para um pacote de comando com a estrutura do fabricante, Dynamixel.

O pacote de comandos convertido para o protocolo do fabricante Dynamixel é transmitido do controlador principal para os servos motores Dynamixel através de seus comandos de escrita nas áreas de memória da tabela de controle reservadas para esta finalidade. Conforme o protocolo de comunicação Dynamixel.
Os transdutores conectados as placas de aquisição de dados 3Mxel e os encoderes absolutos integrados nos módulos Dynamixel transmitem para o controlador principal, OpenCM9.04c, pacotes de estado conforme o seu próprio protocolo de comunicação, Dynamixel.

O programa da placa converte este pacote de estado recebido no formato do protocolo Dynamixel em um pacote de estados conforme o protocolo de comunicação implementado da interface do computador e o controlador principal.
Este pacote de estados, contendo informações das posições e velocidade dos transdutores e das tensões referentes aos transdutores de forças em hexadecimal, são enviados de forma serial para o ambiente do computador.

Desta forma, o software do programa realiza uma função de operar como uma rede de comunicação de dados. O computador envia comandos de controle para o controlador principal do sistema e por sua vez o controlador principal do sistema envia comandos de estado dos transdutores para o computador. Conforme o protocolo de comunicação estabelecido para comunicação desta interface. Caso o modo de operação do sistema seja de malha fechada, controle de velocidade ou torque, o controlador integra as informações de estado e comando desejado, tendo como saída, comandos de atuação conforme uma lei de controle proporcional. O controlador principal recebe o pacote de comandos do computador e converte este pacote para o protocolo de comunicação do fabricante Dynamixel e os transmite aos atuadores, que por sua vez enviam dados sobre seu estado, conforme o seu protocolo de comunicação, Dynamixel. O controlador recebe estes dados e o converte para o protocolo de comunicação da interface computador e placa controladora. Por fim, o controlador principal transmite o pacote de estados estabelecido para o computador. Durante este processo de comunicação serial assíncrona, são realizados os processos de verificação de erro de transmissão e recepção de dados.

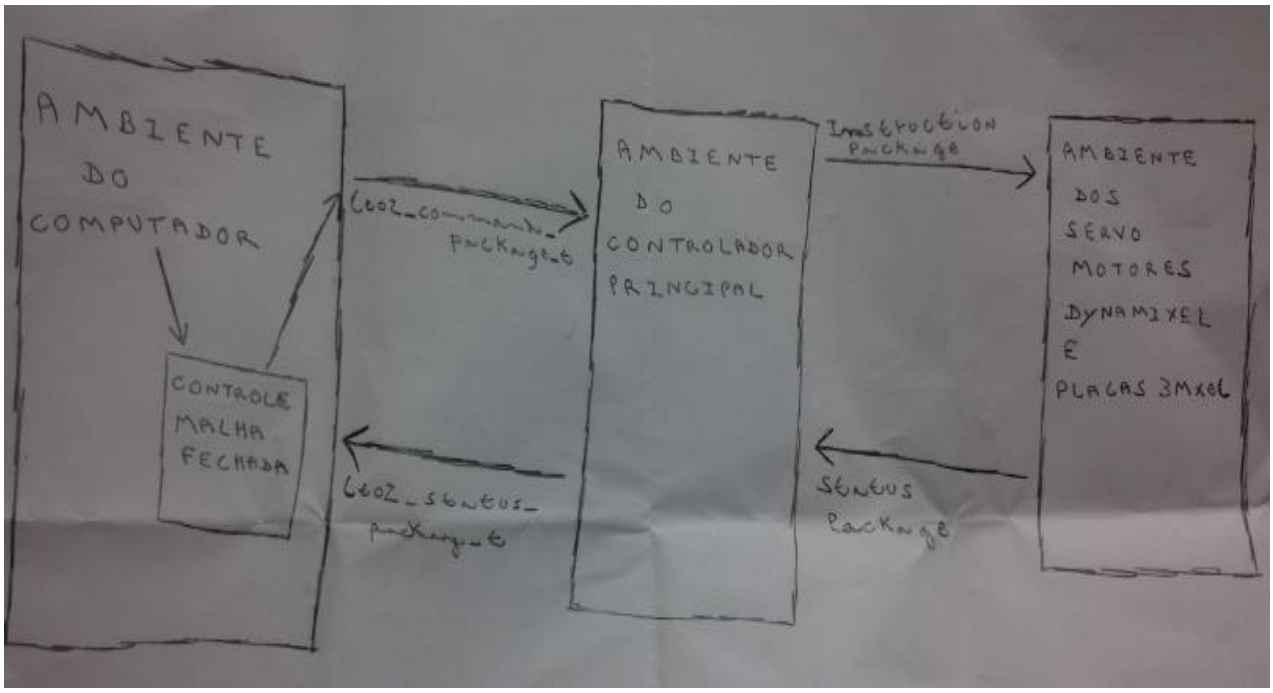


**Figura 22 – Fluxo de Dados do Sistema(esboço)**

# 4. CONCEITOS BÁSICOS

## 4.1. Comunicação Serial

Comunicação serial é uma a forma de transmissão e recepção de dados digitais entre dois ou mais hardwares distintos através de duas interfaces físicas de conexão. Seu padrão de comunicação é a transmissão e recepção de um conjunto de bytes, em que estes bytes são subdivididos e transmitidos à partir de um hardware transmissor de forma sequencial bit a bit e são recepcionados por outro hardware remoto bit a bit, e após esta recepção, o conjunto de bits sequenciais recebidos são convertidos novamente para um conjunto de bytes. Os conjuntos de bytes representam caracteres da tabela ASCII e são armazenados em uma estrutura de dados definida por um protocolo de comunicação comum aos dispositivos de transmissão e recepção do sistema.

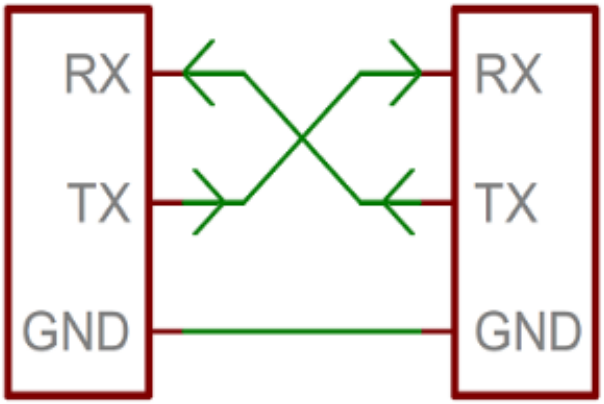


**Figura 23 – Esquema de um Processo de Comunicação Serial**

A comunicação serial pode ser de duas formas, síncrona e assíncrona, cada uma com seu próprio protocolo de transmissão de dados e característica de amostragem:

- Comunicação Serial Síncrona:

É a comunicação serial cuja amostragem dos dados enviados e transmitidos depende de um sinal de clock comum aos hardwares do sistema de comunicação. O reconhecimento da amostragem de um bit para transmissão e recepção é devida a frequência do sinal deste sinal de clock. Logo a forma de comunicação síncrona necessita de um fio para este sinal de clock.

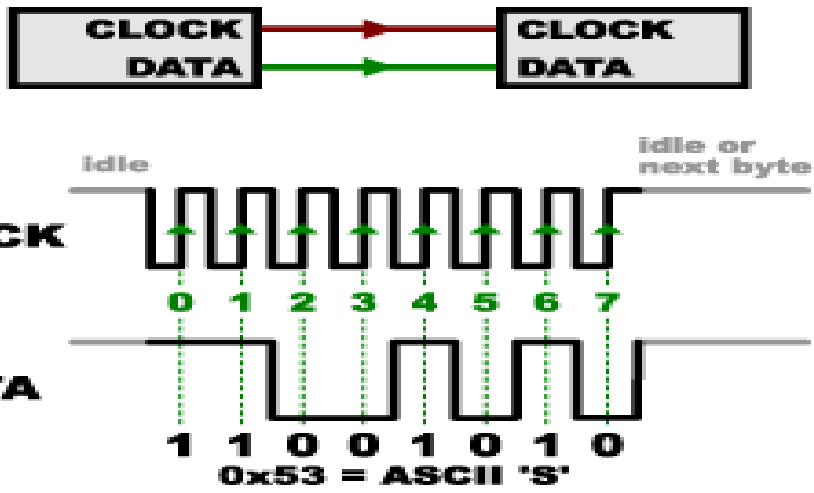


**Figura 24 – Formas de Ondas e Processo de Amostragem Comunicação Serial Síncrona**

- Comunicação Serial Assíncrona:

O protocolo de comunicação serial assíncrono não usa um sinal de clock para estabelecer a amostragem da comunicação de sinais entre dois dispositivos que se comunicam desta forma. Logo, não há a necessidade de um fio para sinal de clock conectando os dispositivos.

A transmissão e recepção de dados é dada pela configuração de uma stream de bits, ou seja, uma sequência de bits de tamanho pré-estabelecida e periódica, contendo bits de dados cuja quantidade pode ser arbitrada entre 5 a 9, e bits especiais de reconhecimento, sincronização e validade. A amostragem dos bits depende diretamente destes bits especiais e da taxa de transmissão de dados (baudrate).

Os bits de validade e integralidade de uma sequência periódica de bits são o bit de start, o bit de paridade e o bit de stop.

O bit de start e o bit de stop são os bits de reconhecimento de uma stream de bits. O start bit está presente no início da stream de bits e indica o início de uma transmissão ou recepção de uma stream, enquanto o stop bit está presente geralmente no final de uma stream de bits e indica o final de uma transmissão ou recepção desta stream. Uma stream de bits pode ter um ou dois bits configurados como stop bit.

A paridade de uma stream de bits é dada por conta da quantidade de bits de dados que estão em nível lógico alto. Esta paridade pode ser classificada como par ou ímpar. Caso a quantidade de bits de dados em nível lógico alto for par, a paridade é ímpar e é adicionado um bit ao final ou início dos bits de dados da stream, sendo o bit de paridade representado em nível lógico alto. Caso a quantidade de bits de dados da stream que estão em nível lógico alto for ímpar, a paridade é classificada como par, sendo o bit de paridade representado em nível lógico baixo.


**Figura 25 – Stream de Bits do Processo de Comunicação Serial Assíncrona**

A taxa de amostragem, ou seja, a frequência que leva para o dispositivo transmissor enviar um bit ou a frequência que leva para um dispositivo receptor receber um bit é configurado como taxa de transmissão de dados (baudrate). Em um processo de comunicação serial assíncrono, ambas as interfaces de transmissão e recepção seriais tem que estar configurada com a mesma taxa de transmissão de dados (baudrate). Pois desta forma garante-se a confiabilidade da comunicação de dados. Caso os dispositivos transmissor e receptor do processo estejam com taxas de transmissão de dados distintas (baudrates diferentes), haverá perda de informações do durante a transmissão, inviabilizando a comunicação.

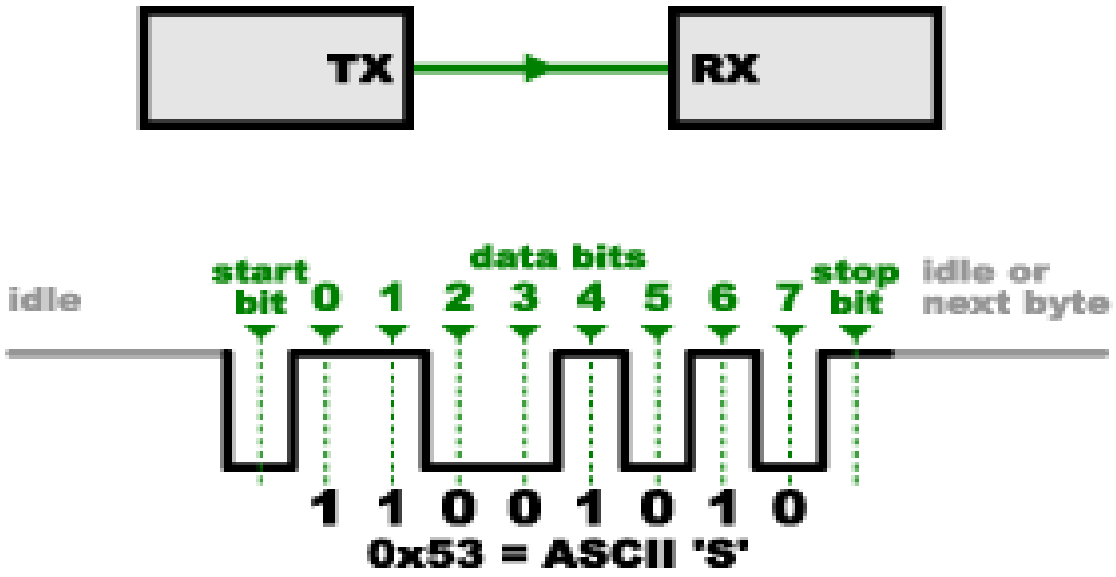


**Figura 26 – Processo de Comunicação Serial Assíncrona e Amostragem de Dados**

Os dispositivos que constituem o meio físico do canal de transmissão de dados serial e assíncrona podem ser classificados quanto ao sentido da transmissão de dados. Caso um dispositivo possa tanto transmitir quanto recepcionar uma sequência de bits ao mesmo tempo, este é classificado como full-duplex, caso o dispositivo possa apenas realizar um dos dois processos de cada vez, ou transmitir ou enviar sequência de bits, o dispositivo é classificado como half-duplex.

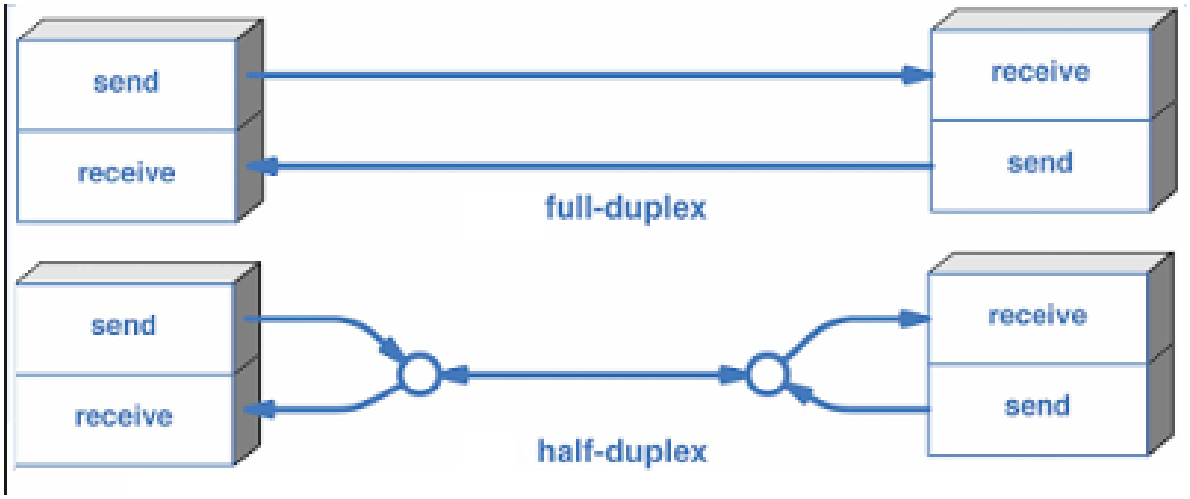


**Figura 27 – Processo de Comunicação Modelos Full-Duplex e Half-Duplex**

No caso do robô bípede LEO2, a comunicação de dados do seu sistema de controle embarcado é serial assíncrona half-duplex.

## 4.2. Meios Físicos de Comunicação Abordados no Sistema de Controle Embarcado

O meio físico padrão RS-485 é adotado para realizar a comunicação entre o controlador principal OpenCM9.04c, a placa controladora OpenCM4.85 , as placas 3Mxel de aquisição de dados dos transdutores e os servos motores integrados Dynamixel. O protocolo de comunicação entre estes dispositivos é o fornecido pelo fabricante Dynamixel, com a configuração de uma stream de bits, contendo 8 bits de dados, 1 bit de parada e sem bit de paridade.

Esta interface de comunicação, RS-485, utiliza um sinal diferencial para a recepção e envio de dados, logo necessita de dois fios para comunicação. A cada etapa do processo de comunicação, os dois fios transmitem ou emitem sinais ao mesmo tempo e na mesma taxa de comunicação, porém com a peculiaridade de que enquanto um sinal está sendo transmitido ou recepcionando por um fio de comunicação, designado D+, o outro fio, D-, está transmitindo ou recepcionando um sinal que é o oposto ao transmitido ou recepcionando por D+. Ou seja, caso o sinal que percorre o canal D+ em um determinado instante esteja em nível lógico alto, o sinal que percorre o canal D- no mesmo instante estará em nível lógico baixo. Esta característica de transmissão de dados filtra efeitos de ruído em transmissões em altas frequências, garante a qualidade dos sinais de comunicação a longa distância(cerca de 2000 metros)entre interface de transmissão e recepção, filtra efeitos de interferência eletromagnéticas que os fios de comunicação sofrem devido a recepção de ruídos, provindos do meio externo como fontes e cargas indutivas, e transmissão de ruídos devido ao processamento em altas taxas de comunicação de dados.

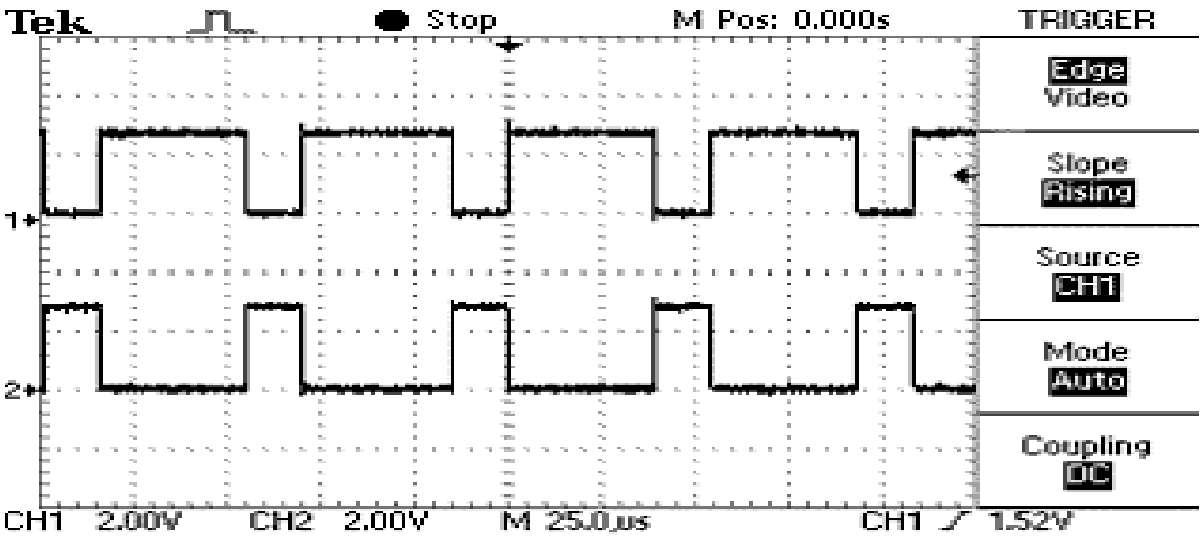


**Figura 28 – Formas de Onda Nos Canais de Comunicação Diferenciais de um dispositivo RS-485, Forma de Onda 1 Canal D+ e Forma de Onda 2 Canal D-**

A tecnologia apresentada pela interface RS-485 possibilita a comunicação de múltiplos transmissores e múltiplos receptores em um sistema de comunicação, configurando uma rede de comunicação de dados em que os transmissores são classificados como o dispositivos Master e os receptores são classificados como dispositivos Slave do sistema.

No sistema de controle embarcado do robô bípede LEO2, há a configuração de um dispositivo Master e sete dispositivos Slave. O dispositivo Master é o controlador principal do sistema, a placa

OpenCM9.04c, enquanto os dispositivos Slave são os hardwares que estão conectados aos canais do BUS de 4 pinos para comunicação assíncrona serial padrão RS-485 da placa expansora OpenCM4.85 , os servos motores integrados Dynamixels e as duas placas 3Mxel de aquisição de sinais dos transdutores. Em uma configuração Master-Slave, o dispositivo Master é o dispositivo de comando do sistema, ou seja, o transmissor, enquanto os dispositivos Slave são dispositivos receptores, que atendem ao comando do transmissor Master e enviam para este o seu sinal de resposta(monitoramento). O Master transmite um sinal direcionado para um dispositivo remoto específico a cada transmissão, pois uma rede padrão RS-485 permite a identificação os dispositivos que o compõem.

A máxima quantidade de dispositivos conectados a uma rede de comunicação de dados padrão RS-485 é dita pela unidade 'unit-load'. Esta unidade em geral está embutida no datasheet do dispositivo e indica uma medida indicativa do quanto cada dispositivo sobrecarrega o sistema. A quantidade máxima teórica em que uma rede de comunicação padrão RS-485 suporta é de 32 'unit-load', sendo que cada dispositivo é caracterizado por uma fração ou múltiplo desta unidade.

O sinal diferencial do padrão RS-485 é devido a uma configuração dos dois fios de dados de mesmo comprimento, D+ e D-, denominada 'par trançado'. Nesta configuração, os dois fios são enrolados um ao outro de forma que os dois fiquem o mais próximo possível um do outro, afim de reduzir a distância física entre eles, de forma que as interferências eletromagnéticas de sinal no processo de comunicação sejam filtradas de uma maneira mais eficaz, garantindo a preservação do sinal transmitido em relação ao sinal recepcionando durante a comunicação das interfaces transmissora e receptora.

A norma TIA-EIA-485 e norma padrão da indústria TSB89 ditam diretrizes para o projeto físico da fiação de uma rede de comunicação baseada no padrão RS-485. Elementos de circuitos são sugeridos a serem acoplados a rede, de forma a otimizar a transmissão de dados para que o sistema atenda as exigências de uma comunicação em uma rede RS-485. Dentre estas especificações estão os resistores de terminação e os resistores de fail-safe bias.

Os fios de comunicação da rede RS-485 possuem uma impedância característica de cerca de 120 ohms. As terminações dos fios, ou seja, os nós que se conectam com os transmissores e receptores do sistema, tem que possuir uma impedância casada com a impedância característica do cabo. Esta questão é solucionada pela instalação de dois resistores em cada extremidade dos fios de comunicação denominados resistores de terminação, cada um com um valor de cerca de 120 ohms, já que este processo é regido pela equação (Rt Zo)/(Zo + Rt), na qual Zo condiz com a impedância característica dos cabos de comunicação e Rt são os valores nominais dos resistores de terminação.
Caso os resistores de terminação dos fios de transmissão não estejam corretamente instalados, ocorrem problemas de 'jitter' de comunicação, que são reflexões de sinais que podem ocorrer durante o processo de comunicação. Ou seja, durante um ciclo de comunicação, o transmissor transmite uma stream de bits e o receptor a recebe, porém antes que o receptor termine de receber esta stream de bits, o dispositivo transmissor envia outra stream de bits. Causando falha e erro de comunicação entre as interfaces como perda de dados de comunicação.

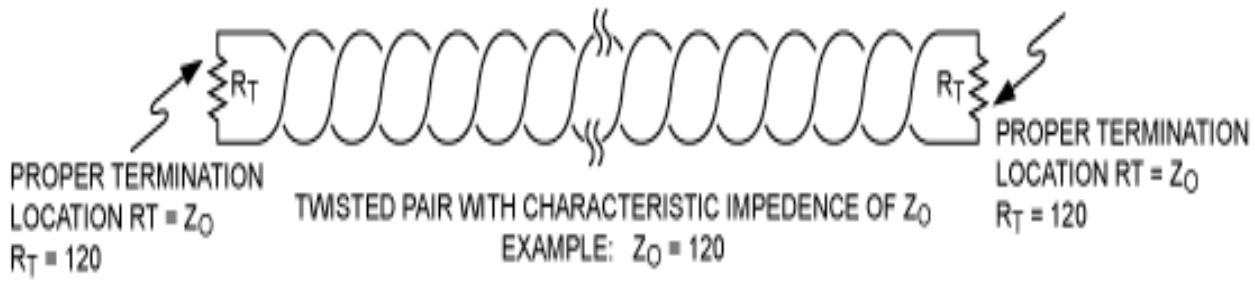


**Figura 29 – Configuração de Cabeamento em Par Trançado do Padrão RS-485 com Respesctiva Impedância Característica do Cabo e Seus Resistores de Terminação**

Durante a operação de comunicação de dados, no período em que o transmissor não está enviando dados, a tensão de entrada dos dispositivos receptores possuem um valor de módulo baixo, a tensão flutua entre -200mV e 200mV, e as saídas dos receptores ficam em um estado indeterminado. Logo durante este período, um receptor pode interpretar que um sinal está sendo enviado e responder a um sinal de ruído, gerando reflexões do sinal, correspondendo ao problema de 'jitter ' na comunicação de dados. Logo há a necessidade da instalação de resistores denominados fail-safe-bias. Estas resistências são instaladas nos terminais de entrada dos receptores. Sendo um resistor de pull-up instalado entre o terminal de dados D+ e uma fonte de 5volts e um resistor de pull-down instalado entre o terra comum do sistema e o terminal de dados D-. Esta configuração entre essas duas resistências e os cabos de transmissão de dados, terra e um nó de polarização, geram um divisor de tensão, o que produz uma polarização nos dispositivos receptores do sistema de comunicação.
Desta maneira, impõe-se ao receptor uma saída em nível lógico alto quando sua entrada está em flutuação ou em nível lógico baixo, condizendo com o estado ligado, ON, e a saída possui uma resposta em nível lógico baixo quando sua entrada possui nível lógico alto, condizendo com o estado desligado OFF. Sendo assim os estados das saídas dos dispositivos de recepção bem definidos durante a comunicação entre interfaces.

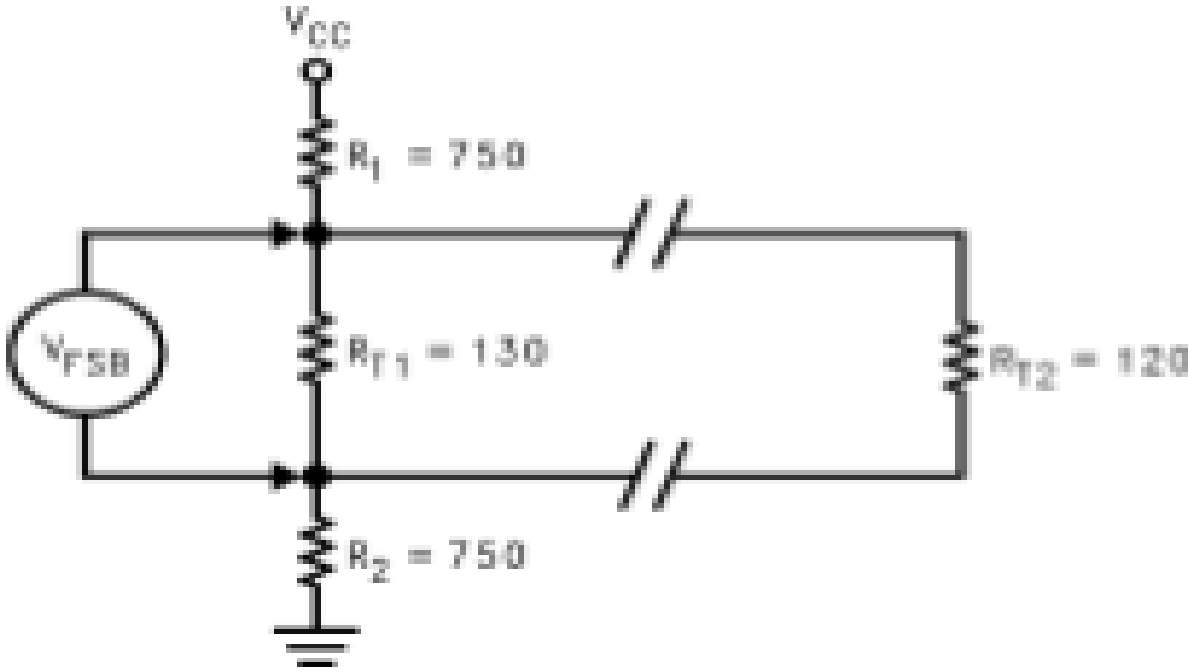


**Figura 30 – Filtro 'Fail Safe Bias', com Ênfase para as Resistências de Pull-Up e Pull-Down**

Os valores nominais dos resistores de pull-up e pull-down dependem das características de velocidade de transmissão e da medida de unit-loads do sistema.

A placa controladora do sistema OpenCM9.04c possui um canal de comunicação de 4 pinos para padrão RS-485, enquanto a placa de expansão OpenCM4.85 possui um BUS com cinco destes canais de comunicação, a placa 3Mxel possui um deste canal e cada módulo Dynamixel possui dois.

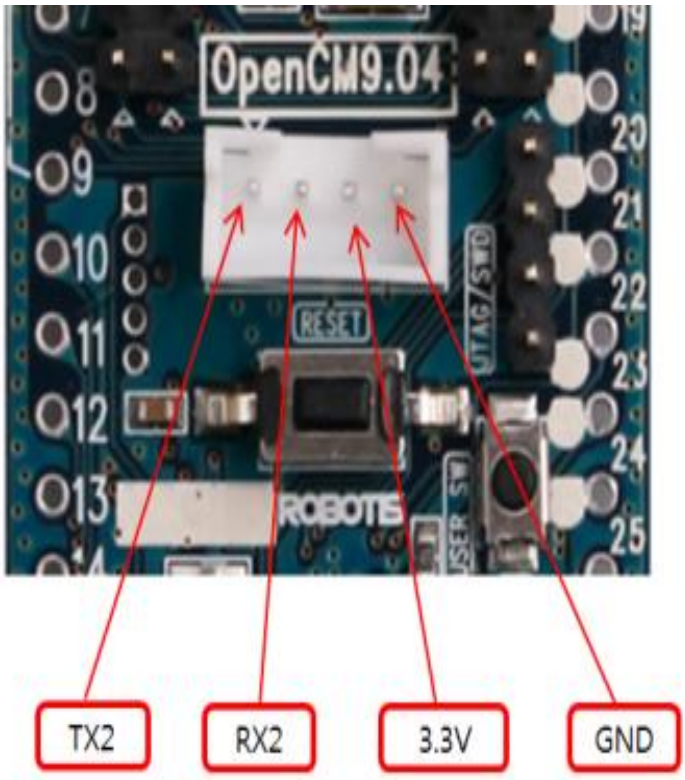


**Figura 31 – Canal de Comunicação Padrão RS-485 do Controlador Principal OpenCM9. 04**

**Figura 32 – Canal de Comunicação Padrão RS-485 da Placa Expansora OpenCM4.85**

O meio físico de comunicação de dados entre o computador e o controlador principal é uma a saída de três pinos para comunicação serial assíncrona da placa conversora USBTODYNAMIXEL e uma porta de conexão de três pinos para comunicação serial em nível TTL. O protocolo de comunicação adotado

entre estes dois ambientes foi o desenvolvido e implementado para realizar esta finalidade, conforme descrito e exemplificado na parte do projeto relativa ao desenvolvimento do software do controle embarcado.

Um pino desta interface corresponde ao sinal de Tx, ou seja, destinado para transmissão de sinal, o outro corresponde ao sinal de Rx, correspondendo a recepção de sinal, enquanto o outro é a referência para o terra.

Nesta interface apenas é possível a comunicação entre apenas dois dispositivos, um transmissor e um receptor e estes tem que estar a uma distância máxima de aproximadamente 12 metros, definindo a baixa imunidade a ruído do canal de comunicação.

Os níveis de comunicação neste padrão de interface são de 0V até 5Volts, já que é constituida por transistores BJT, sendo um sinal entre 0V e 0.8V aproximadamente adotado como um sinal de nível lógico baixo e um sinal entre aproximadamente 3.3V e 5.0V adotado como um sinal de nível lógico alto.

Os canais de comunicação desta interface são um para transmissão de dados, um para recepção de dados e outro para um caminho para o terra (ground).

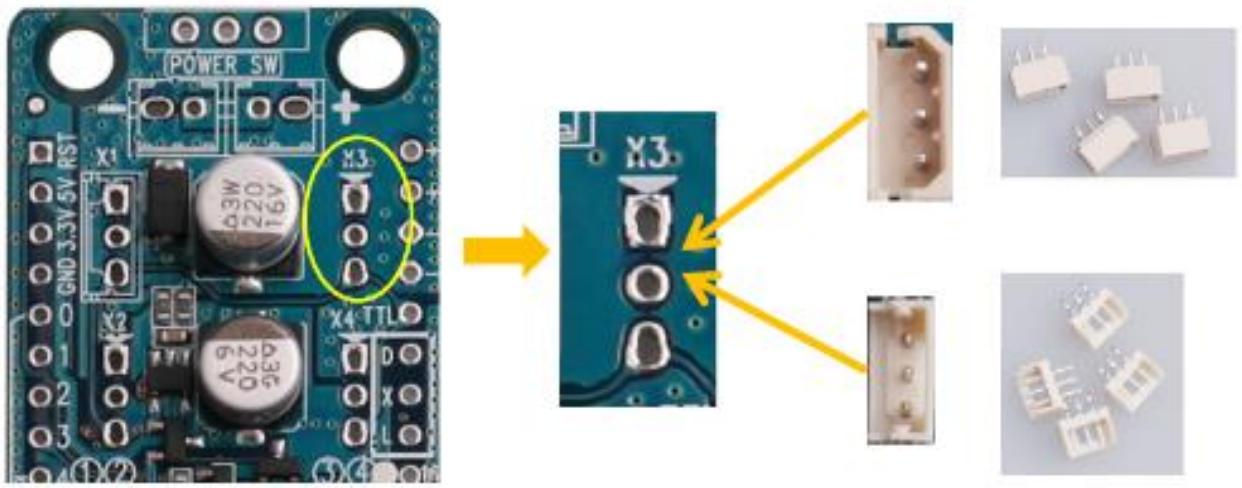


**Figura 33 – Canais de Comunicação Serial Assíncrona Para Nível TTL da Placa OpenCM9.04c**

## 4.3. Questões Relativas aos Meios Físicos de Comunicação e Transmissão de Dados do Sistema

Cabos de transmissão de sinais modelo Dynamixel de quatro pinos conectam os dispositivos de comunicação do sistema que operam conforme o padrão de interfaces RS-485.

**Figura 34 – Cabo de Conexão para Comunicação Serial Assíncrona Half-Duplex padrão RS-485 entre o Controlador, Placas de Aquisição de Dados 3Mxel e Dynamixels**

Durante os testes em tempo real do sistema embarcado do robô bípede LEO2, os dados dos processos foram coletados. Com o auxílio de um osciloscópio, foi possível também analisar visualmente os sinais no processo de comunicação, sendo verificado a existência de problemas na comunicação entre o controlador principal e os Dynamixel condizentes com 'jitter de comunicação'. Este fenômeno acarretou em aumento do tempo morto de comunicação, ou seja, o tempo que leva para um receptor responder após uma instrução de comando ter sido enviada por um transmissor.

Com o auxílio de um multímetro com a chave de seleção em modo ohímetro, foram verificadas todas as combinações entre todos os quatro pinos dos canais de comunicação de quatro pinos para comunicação padrão RS-485 da placa de expansão OpenCM4.85 e dos canais dos Dynamixel, ou seja, foram mensuradas as resistências das combinações de todos os pinos, ground, terra, D+ e D-.

Verificou-se que todas as leituras tinham medidas de alta impedância, ou seja, maior que 20Kohm. Portanto foi verificado a inexistência dos resistores de terminção e fail-sabe bias nestes canais.

Foi sugerida a instalação do circuito de filtragem, afim de harmonizar as questões de 'jitter de comunicação', visando corresponder a um menor índice de tempo morto no sistema de comunicação.

# 5. TESTES EM TEMPO REAL DA REDE DE COMUNICAÇÃO DO SISTEMA

Os Gráficos traçados e os dados estatísticos nesta seção foram obtidos com o auxílio do software Windows Excel, do pacote Office da Microsoft. Enquanto os testes eram feitos em tempo real, uma rotina de programação no ambiente do computador coletava os dados de comando e estado do sistema.

Os testes descritos são referentes ao modo de operação em malha fechada controle de velocidade. Logo, os dados obtidos são relativos ao comando de velocidade dos eixos dos servos motores, correspondendo a atuação direta, e do estado dos encoderes, posição e velocidade, do robô bípede LEO2.

A captura dos dados são relativas ao tempo . Desta forma foi possível capturar dados de tempo morto do sistema e inferir funções estatísticas, gerando informações relativas a questão de 'jitter de comunicação'.

Todos os testes foram realizados a uma taxa de comunicação de 1Mbps.

Durante os testes, ocorreu o aquecimento do microprocessador STM:32F103CB da placa controladora do sistema OpenCM9.04c. Averiguações primárias levam a constatar que o ocorrido pode ter acontecido devido a soltura da solda de um dos canais de comunicação serial assíncrona em níveis TTL entre o computador e a placa durante um dos testes do projeto. Este ocorrido pode ter induzido uma falha no regulador de tensão do microprocessador. Desta forma, a tensão fornecida ao microprocessador é de 5V ao invés da tensão de 3.3V fornecida na saída do regulador de tensão.

Afim de dar continuação ao projeto foi proposto pelo Ilmo. Professor orientador a implementação feita pelo mesmo de um programa computacional na linguagem C++ que visa simular o processo implementado pelo microcontrolador no sistema de controle embarcado, ou seja, um ambiente de placa controladora virtual. Neste programa, é reservado um buffer, ou seja, um vetor que armazena bytes, variáveis do tipo char, onde implementam a função do ambiente de comunicação implementado entre o ambiente de programação do computador e o ambiente de programação do controlador principal do sistema . A forma de transmissão de dados é preservada, ou seja, assíncrona serial com taxa de comunicação de 1Mbps.

A forma de conexão entre os componentes é preservada, porém não há o cabo de conexão USB, entre o controlador principal e o computador, meio físico que comunica o computador e o controlador durante o processo de programação do controlador principal.

O computador se comunica com os Dynamixels através da placa de conversão de sinais USB2DYNAMIXEL, que por sua vez possui a alavanca de seleção de padrão de comunicação na posição para comunicação RS-485 ao invés de níveis TTL . Um cabo para como o da figura 35 é usado para conectar a saída RS-485 da placa USB2DYNAMIXEL a um canal do BUS de 4 pinos para comunicação padrão RS-485 da placa expansora OpenCM4.85 . Os Dynamixels conectados em chain-mode e as placas de aquisição e conversão dos sinais dos transdutores 3Mxel continuam conectadas a placa expansora OpenCM4.85 da mesma forma.

O ciclo de comunicação do sistema é preservado, porém agora desta forma:

1. Ambiente do computador transmite um pacote de comando para o ambiente da placa virtual conforme o protocolo de comunicação estabelecido e implementado que realiza a interface de comunicação entre o ambiente do computador e o ambiente do controlador.

2. Ambiente da placa virtual recebe estes dados os processa e os transmite aos servos motores Dynamixel conforme seu próprio protocolo de comunicação Dynamixel definido pelo fabricante.

3. O ambiente de programação da placa virtual recebe os dados seriais assíncronos de estado dos servos motores Dynamixel e das placas de aquisição e conversão de sinais dos transdutores do sistema 3Mxel , conforme o protocolo de comunicação estabelecido pelo fabricante.

4. O ambiente da placa virtual transmite para o ambiente de programação do computador estes dados de estado conforme o protocolo de comunicação estabelecido e implementado que realiza a interface entre os ambiente de programação do computador e da placa controladora do sistema, na forma de um pacote de estados.

5. O ciclo de reinicia.

Todas as averiguações e tratamentos de erro de comunicação do sistema foram preservados . Assim como a implementação das rotinas de programação referentes a máquina de estados foram programadas nesta configuração.

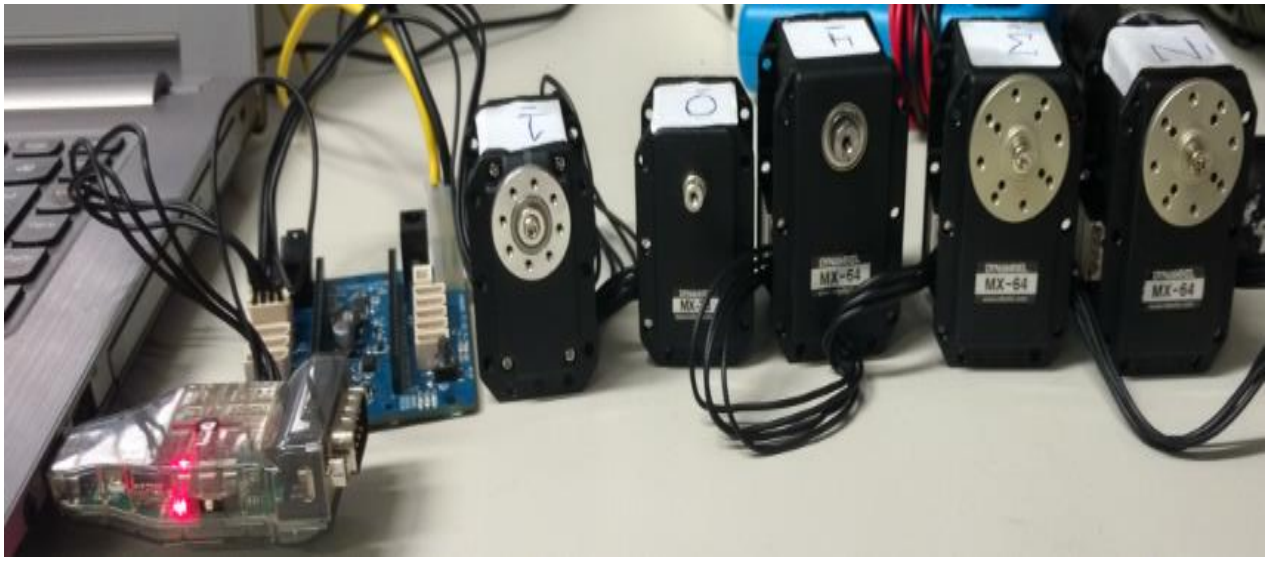


**Figura 35 - Sistema de Controle Embarcado Atual**

## 5.1. Motor Dynamixel Modelo Mx-28 Ganho do controlador 0. 1 e 0. 4

Todos os gráficos abaixos são relativos a testes dos motores do robô bípede LEO2 configurados em modo de controle de velocidade,ou seja,controle de realimentação em malha fechada.

Em todos os gráficos abaixo,a grandeza física velocidade,correspondente a um sinal de comando ou estado,está na unidade rpm.Enquanto a grandeza escalar de estado posição está descrita na unidade grau. A grandeza física tempo está definida na unidade milisegundo e por fim a grandeza para análise do fenômeno de jitter de comunicação definida como Amostras é admensional.

O fenômeno de jitter de comunicação causa efeitos de retardo de tempo entre referentes comandos e respostas do sistema,ou seja,durante o processamento de comunicação, algumas vezes o ambiente do controlador principal tenta ler um byte de comando transmitido pelo do ambiente do computador,porém não consegue receber este byte e a rotina computacional faz com que o controlador principal tente ler novamente este byte.

Outro fator para a contribuição do fenômeno de jitter de comunicação se deve ao fato que apesar da taxa de comunicação do sistema embarcado do robô ser igual em todos os ambientes,1Mbps,o tempo de processamento de informações do computador e do controlador são diferentes.

O efeito de jitter também se deve ao fato que o computador estar rodando outras rotinas de programação processadas em sua CPU ao mesmo tempo que roda a rede de comunicação de dados do robô LEO2.

- **Ganho do Controlador do ambiente de programação do computador, Gp = 0. 1:**

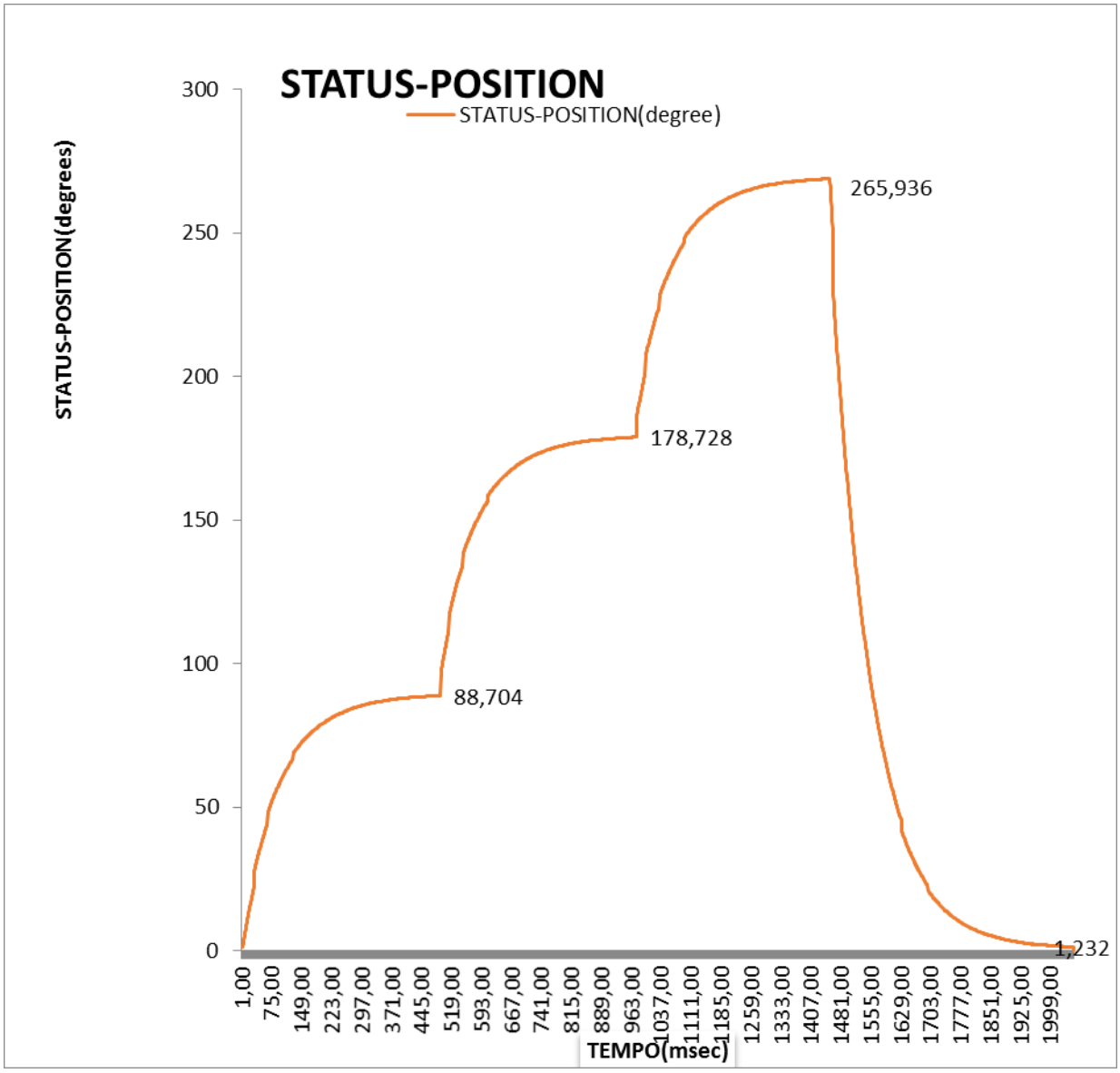


**Figura 36 – Gráfico de Estado, Posição x Tempo :Motor Mx-28 Ganho do Controlador 0. 1, Posição inicial 0 grau, Setpoints sequenciais 90, 180, 270, 0 graus**

O gráfico acima descreve a posição do eixo de saída do servo motor Dynamixel Mx-28 usado como uma articulação direta do robô LEO2 em relação ao tempo.

No ambiente de programação do computador,os campos da variável de controle é configurada para o posicionamento em setpoints sequenciais,à partir da posição inicial de zero graus: noventa graus,cento e oitenta graus,duzentos e setenta graus e por fim retorno a posição inicial zero grau.

Logo o motor realiza  primeiramente um ciclo sequencial em torno de seu eixo de zero e duzentos e setenta graus em sentido horário e por fim um ciclo de duzentos e setenta graus(posição final) até zero graus,posição inicial, em torno de seu eixo no sentido anti-horário.

Analisando o primero ciclo,ou seja,rotação no sentido horário,percebe-se que a taxa de variação de posição do eixo do motor em relação ao tempo,ou seja,sua velocidade angular, comporta-se de forma semelhante nas três sequencias do movimento,realizando o percurso de um quadrante em um tempo total de aproximadamente meio segundo.

Analisando o ciclo final do movimento,ou seja,no sentido anti-horário de rotação,percebe-se o eixo de saída do servo motor MX-28,realiza o percurso de três quadrantes em um tempo total de aproximadamente meio segundo.Logo a uma taxa de variação média de posição relativamente maior que a taxa de variação média de posição do ciclo de sentido horário.Este fato se deve ao controle de velocidade proporcional implementado no ambiente do computador,já que o setpoint da última sequencia de movimentos é mais distante de sua posição inicial do que em relação as três primeiras sequências de movimento.

Em todas as sequências de movimento,percebe-se que a taxa de variação de posição do eixo de saída do servo motor é maior quando este está mais afastado em relação a sua posição instantânea e o setpoint configurado e esta taxa de variação de posição diminui na medida que a posição instantânea do eixo de saída do servo motor se aproxima do setpoint configurado.

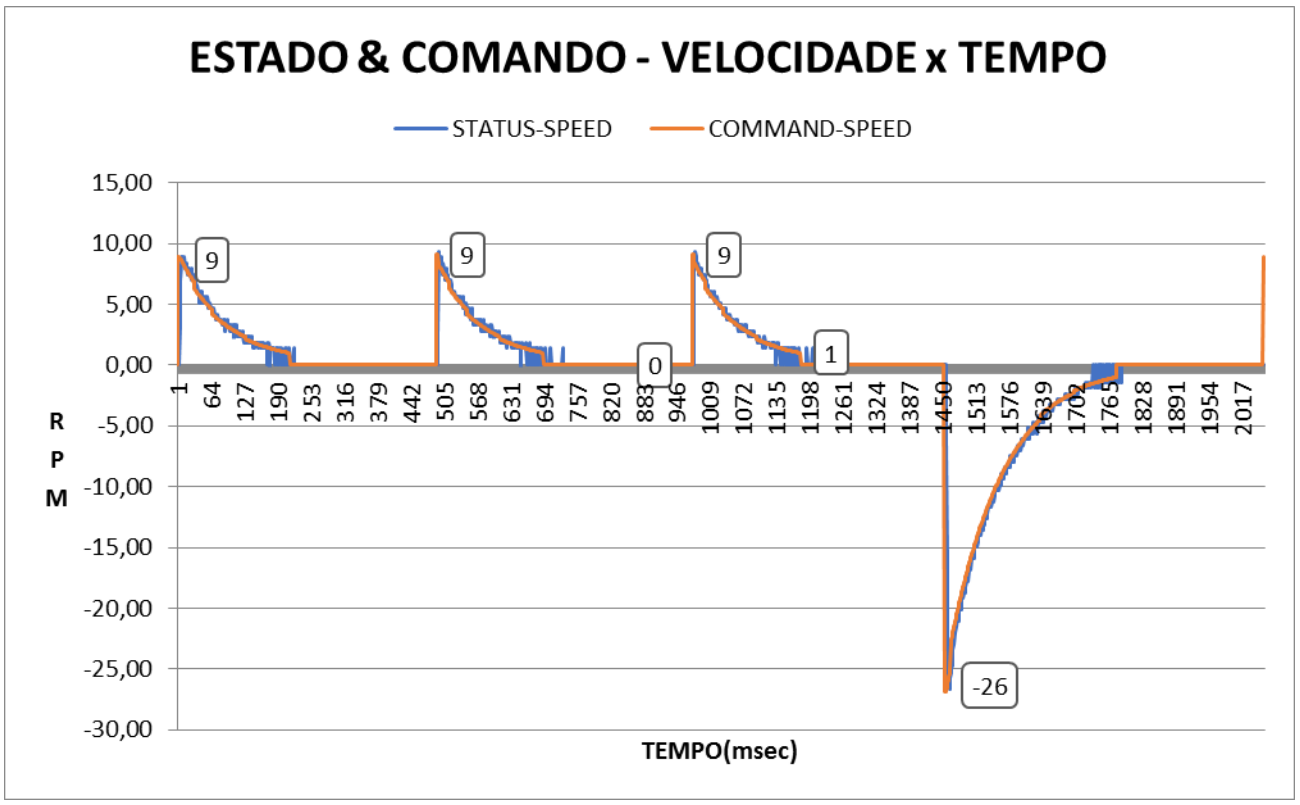


**Figura 37 – Gráfico de Estado & Comando, Velocidade x Tempo : Motor Mx-28 Ganho do Controlador 0.1, Posição Inicial 0 grau & Setpoints Sequenciais 90, 180, 270, 0 graus**

O gráfico acima descreve a variação da posição,ou seja, velocidade angular do eixo de saída do servo motor Dynamixel Mx-28 usado como uma articulação direta do robô LEO2 em relação ao tempo.

A curva vermelha se refere ao comando de velocidade do sistema,enquanto a curva azul se refere a variável de estado velocidade.

Conforme mencionado na análise o gráfico posição versus tempo da figura 36, no início do movimento de cada sequência de posicionamento,a taxa de variação da posição angular do eixo de saída do servo motor é a grande e vai decaindo conforme se aproxima do setpoint referido.Conforme demostra o comportamento das curvas do gráfico acima.Porém quando o eixo do motor se aproxima de seu setpoint,a curva de medição da variável de estado velocidade,azul,não consegue acompanhar a curva da variável de atuação comando de velocidade,vermelha.A curva de estado oscila em torno da curva de comando que tende ao setpoint referido. Após este período de tentativa de estabilização em torno do setpoint,a curva azul,monitoramento,mostra que o eixo de saída do servo motor se estabiliza no setpoint configurado e se sobrepõe a curva de comando vermelha.

- **Ganho do Controlador do ambiente de programação do computador, Gp = 0. 4:**

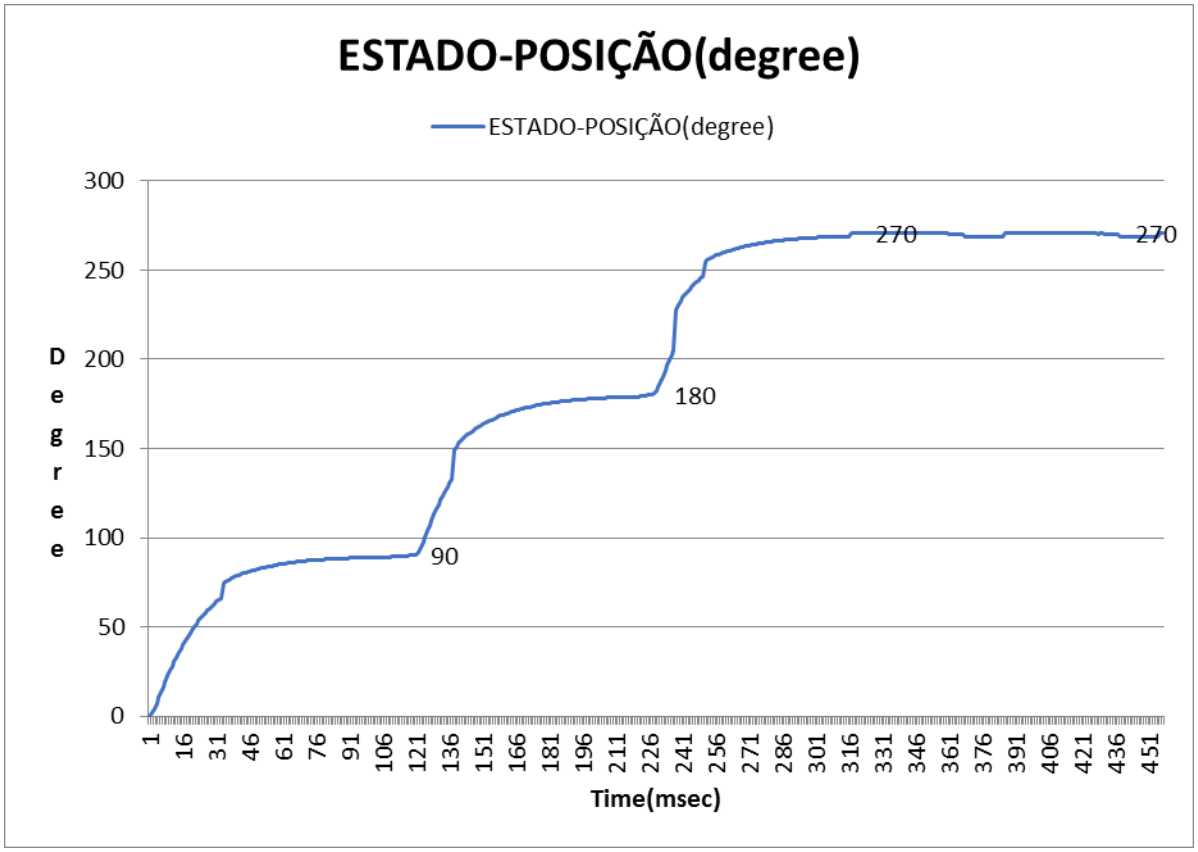


**Figura 38 – Gráfico de Estado, Posição x Tempo : Motor Mx-28 Ganho do Controlador 0. 4, Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 graus**

O gráfico acima descreve a posição do eixo de saída do servo motor Dynamixel Mx-28 usado como uma articulação direta do robô LEO2 em relação ao tempo.

No ambiente de programação do computador,os campos da variável de controle é configurada para o posicionamento em setpoints sequenciais,à partir da posição inicial de zero graus: noventa graus,cento e oitenta graus,duzentos e setenta graus.

Logo o motor realiza um ciclo sequencial de movimento em torno de seu eixo de zero a duzentos e setenta graus em sentido horário.

Analisando o ciclo de movimentos,temos que cada uma das três sequências possui um tempo total de estabilização em torno dos setpoints de 100 milisegundos,logo a taxa de variação de posição média destas sequências são semelhantes,já que a distância entre as posições iniciais e finais do eixo de saída do servo motor são idênticas em todas elas.

Em todas as sequências de movimento,percebe-se que a taxa de variação de posição do eixo de saída do servo motor é maior quando este está mais afastado em relação a sua posição instantânea e o setpoint configurado e esta taxa de variação de posição diminui na medida que a posição instantênea do eixo de saída do servo motor se aproxima do setpoint configurado.

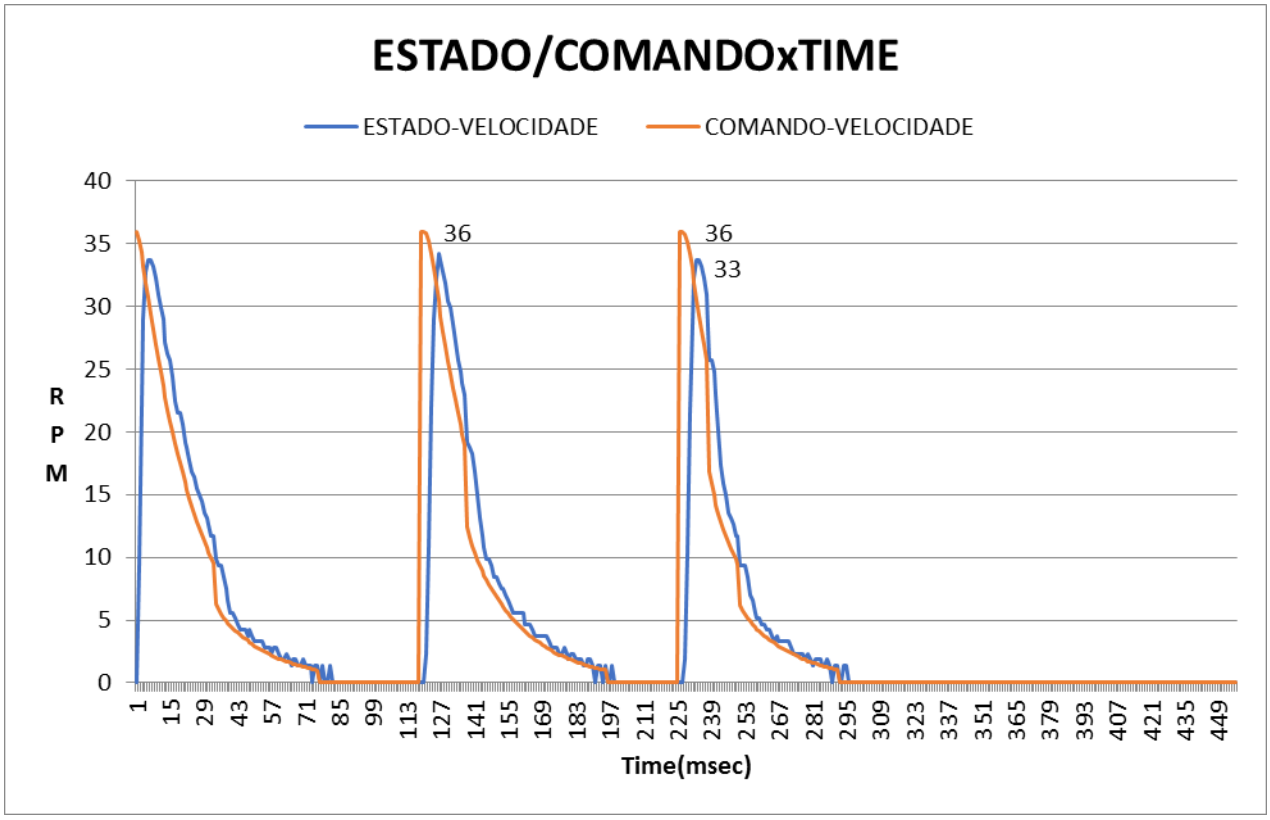


**Figura 39 - Gráfico de Comando & Estado Posição x Tempo : Motor Mx-28 Ganho do Controlador 0. 4 Posição Inicial 0 grau e Setpoints sequenciais 90, 180 e 270 graus**

O gráfico acima descreve a variação da posição,ou seja, velocidade angular do eixo de saída do servo motor Dynamixel Mx-28 usado como uma articulação direta do robô LEO2 em relação ao tempo.

A curva vermelha se refere ao comando de velocidade do sistema,enquanto a curva azul se refere a variável de estado velocidade.

Conforme mencionado na análise o gráfico posição versus tempo da figura 38, no início do movimento de cada sequência de posicionamento,a taxa de variação da posição angular do eixo de saída do servo motor é a grande e vai decaindo conforme se aproxima do setpoint referido.Conforme demostra o comportamento das curvas do gráfico acima.Porém quando o eixo do motor se aproxima de seu setpoint,a curva de medição da variável de estado velocidade,azul,não consegue acompanhar a curva da variável de atuação comando de velocidade,vermelha.A curva de estado oscila em torno da curva de comando que tende ao setpoint referido. Após este período de tentativa de estabilização em torno do setpoint,a curva azul,monitoramento,mostra que o eixo de saída do servo motor se estabiliza no setpoint configurado e se sobrepõe a curva de comando vermelha.

Fazendo uma conjectura de dados de comando e estado dos dois testes com configurações de ganho do controlador de velocidade distintas,percebe-se que a taxa de variação média de posição para os intervalos com mesmas posições iniciais e finais são maiores para o teste cujo ganho do controlador foi de 0.4 e menores para o teste cujo ganho do controlador proporcional foi de 0.1.

Os gráficos e a tabela referentes a análise de tempo morto do sistema são iguais para os dois testes,já que a variação de setpoint ou ganho do controlador de malha fechada não interferem no efeito de jitter de comunicação do sistema de controle embarcado.

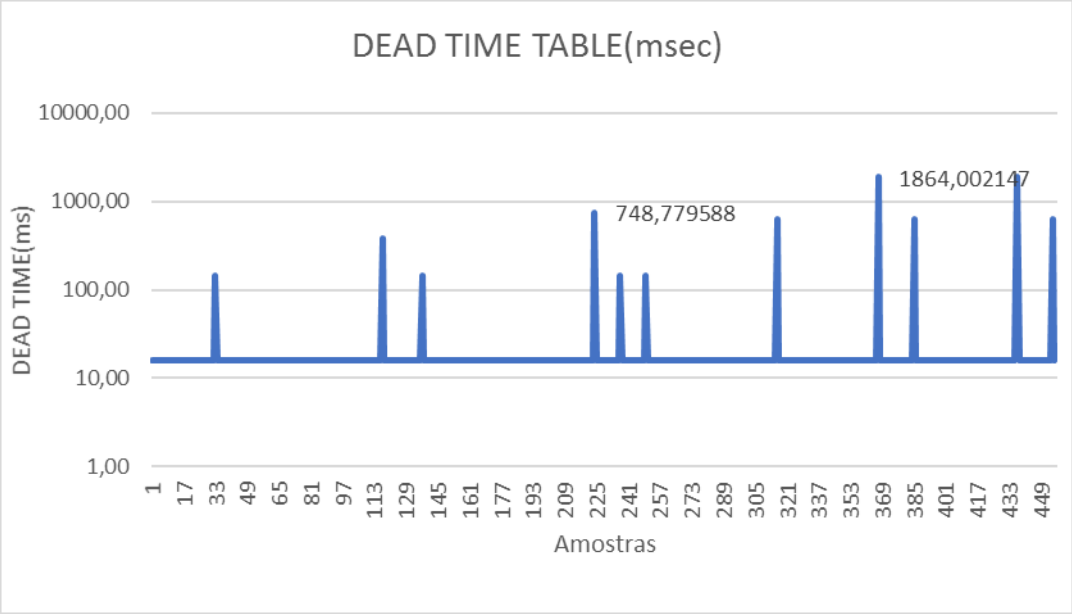


**Figura 40 – Gráfico de Tempo Morto x Amostras(Tabela) : Motor Mx-28 Ganho do Controlador 0. 1 e 0.4, Posição Inicial 0 grau & Setpoints Sequenciais 90, 180, 270, 0 graus**

O gráfico acima relaciona o tempo morto do sistema por amostra,analisando o gráfico,percebe-se um padrão de valor constante de tempo morto entre amostras em algumas faixas de valores de amostras,porém aleatóriamente durante o processamento de comunicação do sistema,há picos de valores de tempo morto entre amostras.Estes efeitos tendem a desestabilizar o processo de dinâmica do robô LEO2 devido a retardos na comunicação do sistema.

| TIME(msec) | DEAD TIME(msec) | STATISTIC FUNCTIONS | TABLE | SAMPLES |
|---|---|---|---|---|
| 121, 32 | 15, 91 | **MEDIAN** | 15, 98 | 15, 97 |
| 137, 23 | 15, 98 | **PATTERN DEVIATION** | 696, 4903321 | 0, 064809218 |
| 153, 22 | 15, 88 | **MEAN ABSOLUTE VALUE** | 63, 74006411 | 0, 058552113 |
| 169, 10 | 15, 96 | | | |
| 185, 06 | 15, 91 | | | |
| 200, 97 | 16, 00 | | | |
| 216, 98 | 15, 98 | | | |
| 232, 95 | 15, 91 | | | |
| 248, 87 | 15, 98 | | | |
| 264, 84 | 16, 01 | | | |
| 280, 85 | 15, 85 | | | |
| 296, 70 | 16, 03 | | | |
| 312, 73 | 15, 86 | | | |
| 328, 60 | 16, 01 | | | |
| 344, 61 | 15, 97 | | | |
| 360, 58 | 15, 91 | | | |
| 376, 48 | 15, 98 | | | |
| 392, 47 | 16, 02 | | | |
| 408, 48 | 15, 85 | | | |
| 424, 33 | 16, 03 | | | |
| 440, 37 | 15, 99 | | | |
| 456, 36 | 15, 85 | | | |
| 472, 21 | 16, 01 | | | |
| 488, 22 | 15, 87 | | | |
| 504, 09 | 15, 99 | | | |
| 520, 09 | 15, 88 | | | |
| 535, 96 | 16, 00 | | | |
| 551, 97 | 16, 03 | | | |
| 568, 00 | 15, 87 | | | |
| 583, 86 | 15, 97 | | | |
| 599, 84 | 15, 86 | | | |
| 615, 69 | 16, 03 | | | |
| 631, 72 | 143, 36 | | | |

**Tabela 1 – Amostragem e Funções Estatísticas Relativas ao Teste com Motor Mx-28 Ganho do Controlador 0. 1, Posição Inicial 0 grau & Setpoints Sequenciais 90, 180, 270, 0 graus**

A tabela acima refere-se a um intervalo de tempo de processamento do sistema onde o tempo morto entre as amostras permanece em um padrão quase constante, excludente a última amostra,que equivale a um elevado valor aleatório de tempo morto entre amostras no processo de comunicação do controle embarcado do robô LEO2.

A tabela também possui dados estatísticos de mediana,desvio padrão e valor médio absoluto relativos ao fenômeno de jitter de comunicação. Os valores da coluna TABLE são referentes aos valores das funções estatísticas cujos domínios são relativos a todo o período do processo de comunicação durante a dinâmica do teste referido.Enquanto os valores da coluna SAMPLES são referentes aos valores das funções estatísticas cujos domínios são referentes a uma faixa de comunicação livre de picos de tempo morto entre amostras,conforme a tabela 1,excludente o valor referente a última amostra.

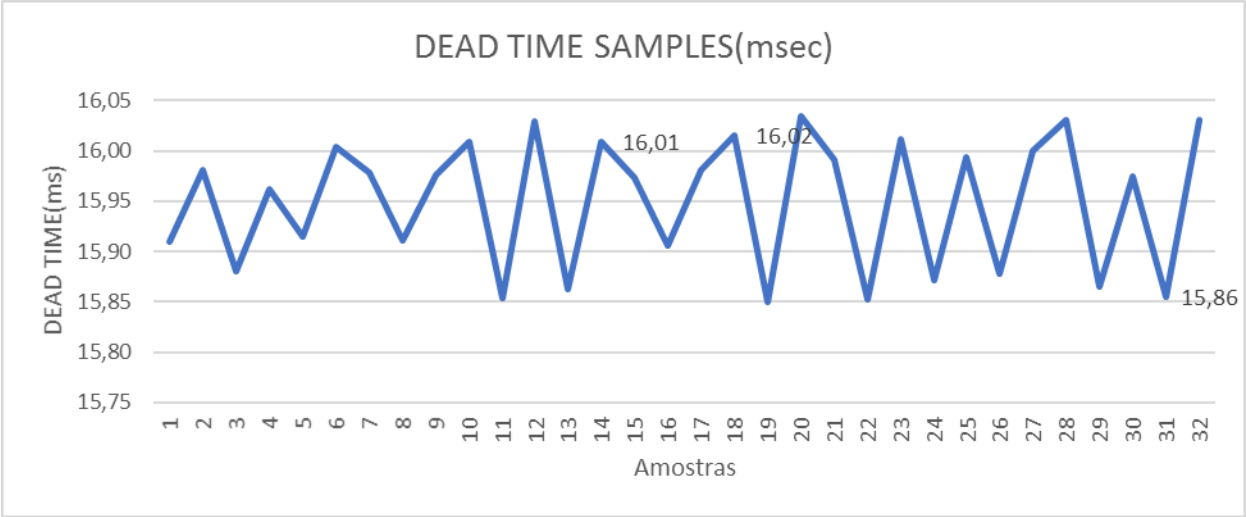


**Figura 41 – Tempo morto x Amostras(Amostragem) : Motor Mx-28 Ganho do Controlador 0. 1 e 0.4, Posição Inicial 0 grau & Setpoints Sequenciais 90, 180, 270, 0 graus**

O gráfico acima relaciona o tempo morto do sistema por amostra,conforme a tabela 1,analisando o gráfico,apesar que o tempo morto entre amostras apresente um padrão quase constante em algumas faixas de amostras da figura 40,percebe-se que este padrão não é constante,variando em uma faixa de valores de tempo morto entre aproximadamente 15,85 segundos e 16,05 segundos.Caracterizando o efeito de jitter de comunicação.

Os dados estatísticos mostram:

O valor da função estatística mediana cujo domínio é o intervalo inteiro do processamento de dados e o valor da função estatística mediana cujo domínio são os valores da tabela 1,excludente a última amostra, são quase idênticos,bem próximo um do outro.O que mostra que apesar dos picos de tempo morto durante o processamento de dados,os valores destas funções medianas do jitter de comunicação são semelhantes e pouco variam.

Os valores da funções estatísticas desvio padrão e valor médio absoluto para o intervalo inteiro do processamento de dados,coluna TABLE, e do valor selecionado na tabela 1,coluna SAMPLE, são bastante diferentes.O valor do desvio padrão e do valor médio absoluto cujo domínio é todo o intervalo de processamento,coluna TABLE, é muito mais elevado do que o valor do desvio padrão e do valor médio absoluto cujo domínio são as amostras da tabela 1 excludente a última amostra.

Este efeito se caracteriza pelos picos aleatórios dos valores de tempo morto quando o domínio é relativo a todas as mostras do processamento de dados,levando a um alto valor de desvio padrão e valor médio absoluto,enquanto os valores de tempo morto quando o domínio é relativo as amostras da tabela 1 ,apesar de variarem,estes ficam contidos em uma faixa de valores,logo os valores das funções desvio padrão e valor médio absoluto com este domínio são baixos em relação aos valores destas funções cujo domínio é relativo a todas as amostras do processamento de dados.

## 5.2. Motores Dynamixel Modelos Mx-28 e Mx-64 Conjuntos Ganho do controlador 0. 15

- **Ganho do Controlador do ambiente de programação do computador, Gp = 0. 15:**

Nos gráficos de estados abaixo,figuras 42 e 43,a curva azul se refere ao servo motor de identidade ID0 modelo Mx-28,enquanto a curva vermelha se refere ao servo motor de identidade ID1 modelo Mx-28 e a curva azul se refere ao motor de identidade ID2 Mx-64.Percebe-se que o motor com identidade ID0 se mantém na posição de setpoint por um período maior que identidade ID1 que por sua vez se mantém na posição de setpoint por um período maior do que o motor cuja identidade é ID2.Ou seja, um motor se estabiliza primeiro que o outro.Este fato se deve que a comunicação do sistema de controle embarcado do robô bípede LEO2 é serial e assíncrona na forma master-slave.

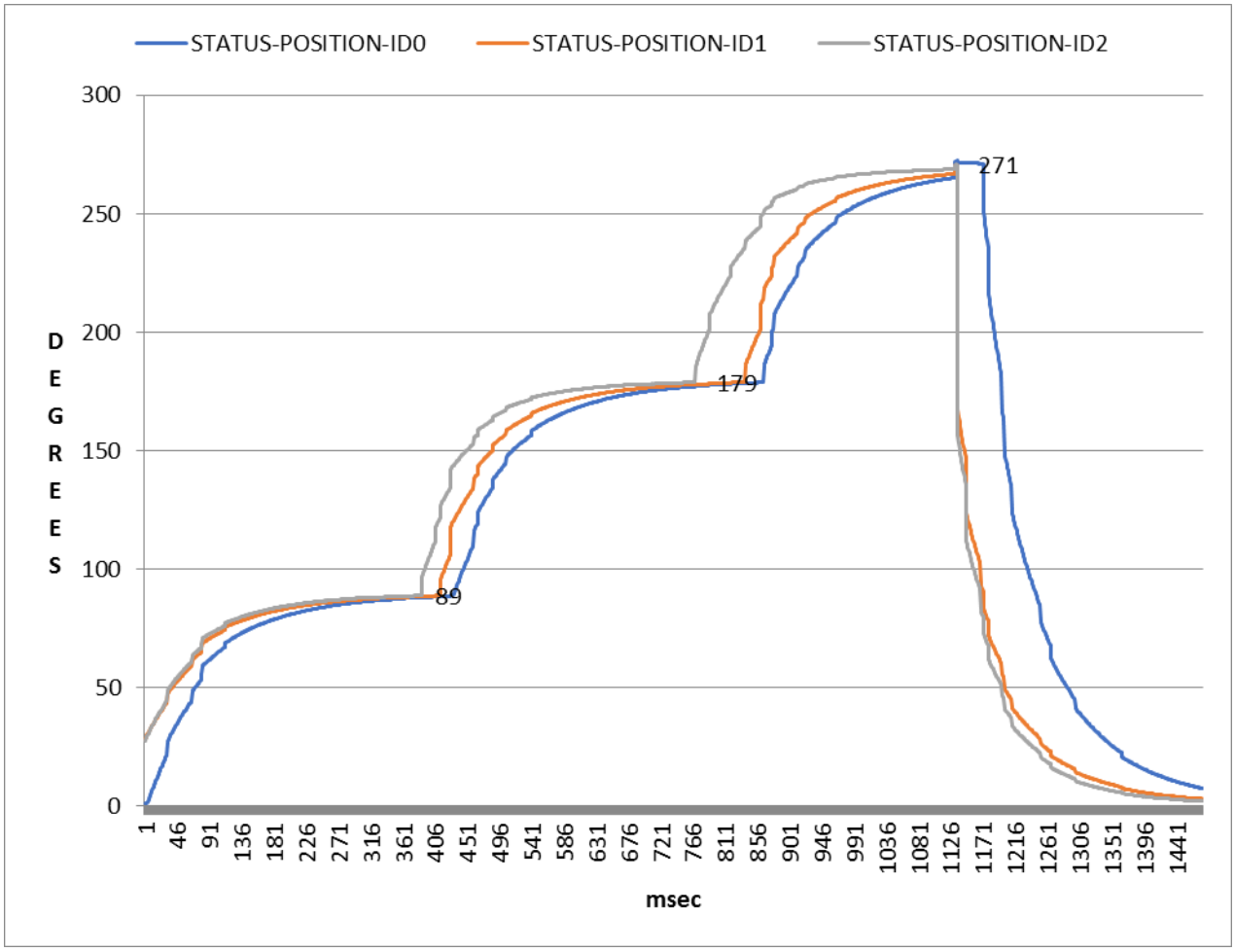


**Figura 42 – Gráfico de Estado, Posição x Tempo: Motores Mx-28 ID0 e ID1 e Motor Mx-64 ID2 Ganho do Controlador 0.15,Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 e 0 graus**

O gráfico acima descreve a posições dos eixos de saída dos servos motores Dynamixel Mx-28 e Mx-64 usados como articulações diretas do robô LEO2 em relação ao tempo.

No ambiente de programação do computador,os campos das variáveis de controle relativas a cada motor são configuradas para o posicionamento em setpoints sequenciais,à partir das posições iniciais de zero graus: noventa graus,cento e oitenta graus,duzentos e setenta graus e por fim retorno as posições iniciais zero grau.

Logo cada motor realiza primeiramente um ciclo sequencial em torno de seu eixo de zero e duzentos e setenta graus em sentido horário e por fim um ciclo de duzentos e setenta graus(posição final) até zero graus,posição inicial, em torno de seu eixo no sentido anti-horário.

Analisando o primero ciclo,ou seja,rotação no sentido horário,percebe-se que os eixos de cada motor em relação ao tempo,ou seja,sua velocidade angular, comportam-se de forma semelhante nas três sequencias do movimento,realizando o percurso de um quadrante em um tempo total aproximadamente menor que meio segundo.

Analisando o ciclo final do movimento,ou seja,no sentido anti-horário de rotação,percebe-se que o eixo de saída de cada servo motor,realiza o percurso de três quadrantes em um tempo total aproximadamente menor do que meio segundo.Logo a uma taxa de variação média de posição relativamente maior do que a taxa de variação média de posição relativa as sequencias de movimento do ciclo de sentido horário.Este fato se deve ao controle de velocidade proporcional implementado no ambiente do computador,já que o setpoint da última sequencia de movimentos de cada motor é mais distante de sua posição inicial do que em relação as três primeiras sequências de movimento.

Em todas as sequências de movimento,percebe-se que a taxa de variação de posição do eixo de saída dos servos motores são maiores quando estes estão mais afastados em relação as suas posições instantâneas e o setpoint configurados e estas taxas de variação de posição diminuem na medida que as posições instantâneas dos eixos de saída do servos motores se aproximam do setpoints configurados.

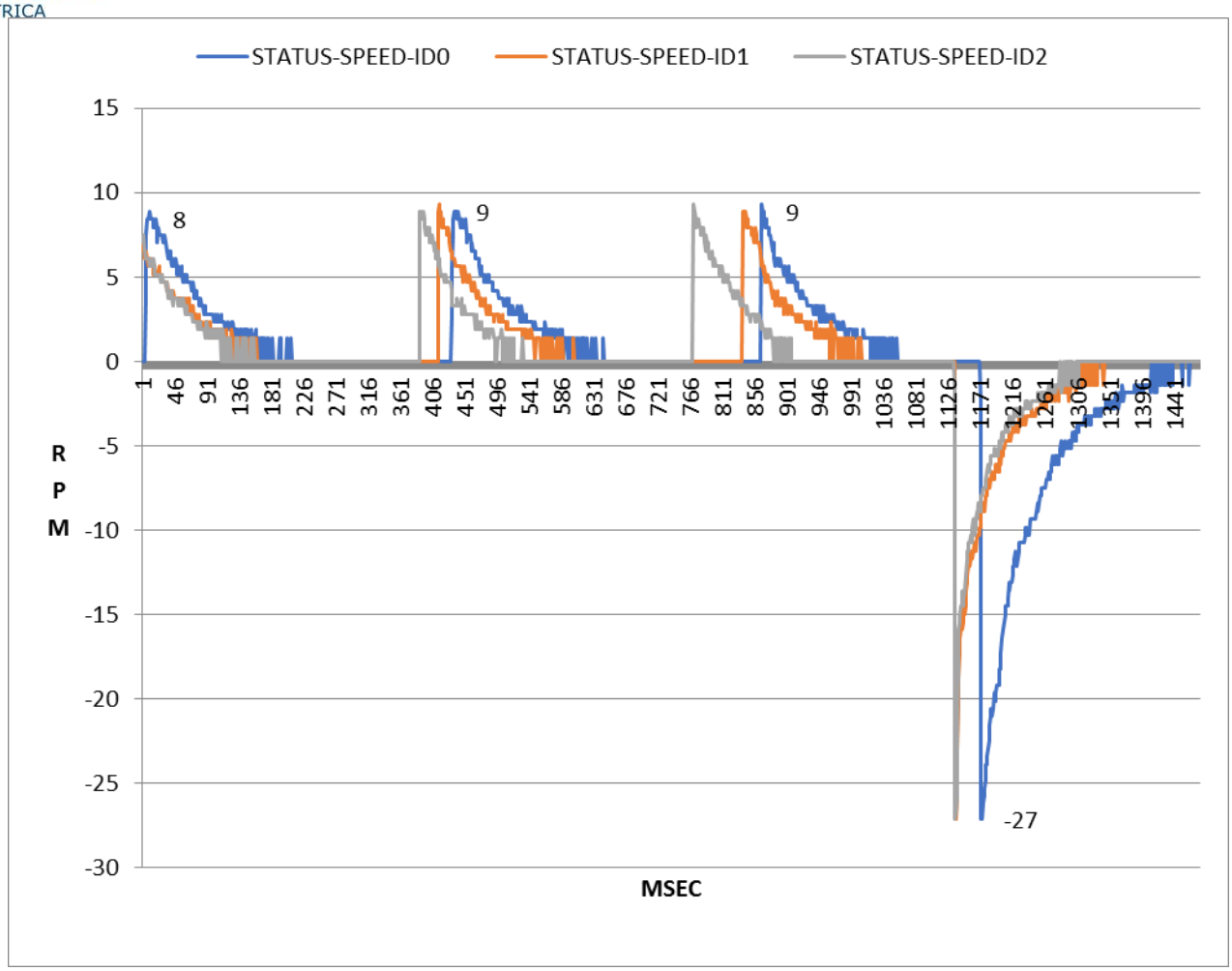


**Figura 43 – Gráfico de Comando & Estado, Velocidade x Tempo : Motores Mx-28 ID0 e ID1 e Motor Mx-64 ID2 Ganho do Controlador 0.15, Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 e 0 graus**

O gráfico acima descreve a variação das posições,ou seja, velocidades angulares dos eixos de saída dos servos motores Dynamixel Mx-28 e Mx-64 usados como uma articulações diretas do robô LEO2 em relação ao tempo.

Conforme mencionado na análise o gráfico posição versus tempo da figura 42, no início dos movimentos de cada sequência de posicionamento,a taxa de variação angular de cada posição do eixo de saída dos servos motores é a grande e vai decaindo de conforme se aproxima dos setpoints referidos.Conforme demostra o comportamento das curvas do gráfico acima.

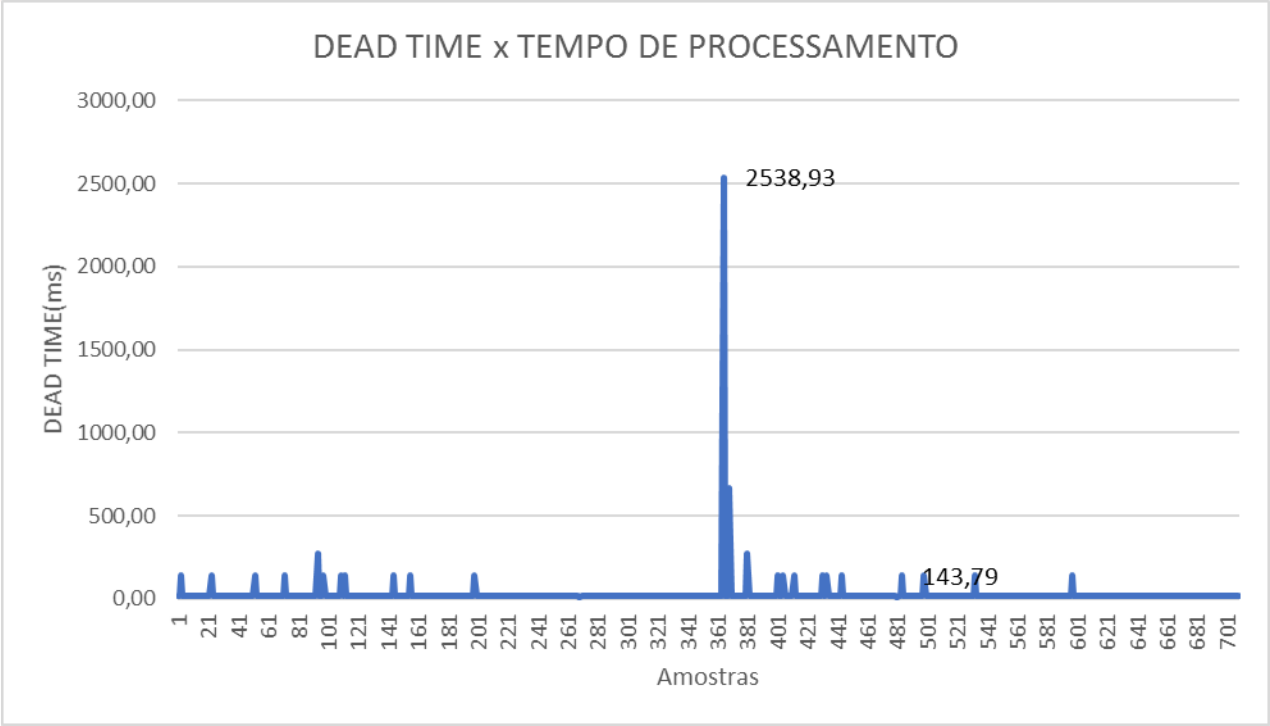


**Figura 44 - Gráfico de Tempo Morto x Amostras (Tabela) : Motores Mx-28 ID0 e ID1 e Motor Mx-64 ID2 Ganho do Controlador 0.15, Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 e 0 graus**

O gráfico acima relaciona o tempo morto do sistema por amostra,analisando o gráfico,percebe-se um padrão de valor constante de tempo morto entre amostras em algumas faixa de valores de amostras,porém aleatóriamente durante o processamento de comunicação do sistema,há picos de valores de tempo morto entre amostras.Estes efeitos tendem a desestabilizar o processo de dinâmica do robô LEO2 devido a retardos na comunicação do sistema.

| Time(msec) | Dead Time(msec) | STATISTIC FUNCTIONS | TABLE | SAMPLES |
|---|---|---|---|---|
| 41319,22 | 16,02 | MEDIAN | 15,98 | 15,97 |
| 41335,25 | 16,03 | PATTERN DEVIATION | 1573,41 | 0,07 |
| 41351,27 | 15,83 | MEAN ABSOLUTE ERROR | 2611,65 | 0,06 |
| 41367,10 | 15,95 | | | |
| 41383,05 | 16,07 | | | |
| 41399,11 | 15,85 | | | |
| 41414,96 | 16,04 | | | |
| 41431,00 | 15,93 | | | |
| 41446,93 | 15,98 | | | |
| 41462,91 | 15,95 | | | |
| 41478,86 | 15,98 | | | |
| 41494,85 | 15,98 | | | |
| 41510,82 | 15,91 | | | |
| 41526,74 | 15,97 | | | |
| 41542,71 | 15,89 | | | |
| 41558,60 | 16,11 | | | |
| 41574,71 | 15,89 | | | |
| 41590,60 | 15,99 | | | |
| 41606,59 | 15,86 | | | |
| 41622,46 | 16,01 | | | |
| 41638,47 | 16,04 | | | |
| 41654,51 | 15,84 | | | |
| 41670,35 | 15,98 | | | |
| 41686,32 | 16,03 | | | |
| 41702,36 | 15,86 | | | |
| 41718,21 | 16,01 | | | |
| 41734,22 | 15,96 | | | |
| 41750,17 | 15,93 | | | |
| 41766,11 | 15,97 | | | |
| 41782,08 | 15,99 | | | |
| 41798,06 | 15,90 | | | |
| 41813,97 | -41813,97 | | | |

**Tabela 2 – Amostragem e Funções Estatísticas Relativas ao Teste com : Motores Mx-28 ID0 e ID1 e Motor Mx-64 ID2 Ganho do Controlador 0. 15, Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 e 0 graus**

A tabela acima refere-se a um intervalo de tempo de processamento do sistema onde o tempo morto entre as amostras permanece em um padrão quase constante, excludente a última amostra,que equivale a um elevado valor aleatório de tempo morto entre amostras no processo de comunicação do controle embarcado do robô LEO2.

A tabela também possui dados estatísticos de mediana,desvio padrão e valor médio absoluto relativos ao fenômeno de jitter de comunicação. Os valores da coluna TABLE são referentes aos valores das funções estatísticas cujos domínios são relativos a todo o período do processo de comunicação durante a dinâmica do teste referido.Enquanto os valores da coluna SAMPLES são referentes aos valores das funções estatísticas cujos domínios são referentes a uma faixa de comunicação livre de picos de tempo morto entre amostras,excludente o valor referente a última amostra.

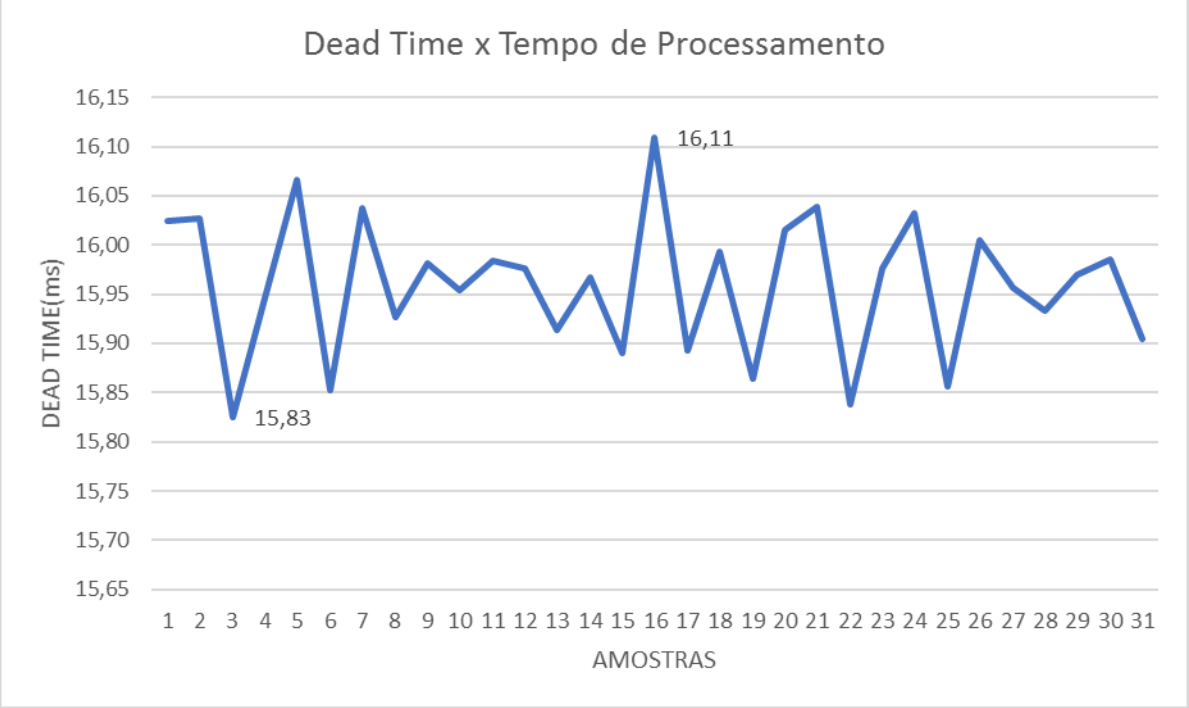


**Figura 45 - Gráfico de Tempo Morto s Amostras (Amostragem) : Motores Mx-28 ID0 e ID1 e Motor Mx-64 ID2 Ganho do Controlador 0.15, Posição Inicial 0 grau & Setpoints sequenciais 90, 180, 270 e 0 graus**

O gráfico acima relaciona o tempo morto do sistema por amostra,conforme a tabela 2,analisando o gráfico,apesar que o tempo morto entre amostras apresente um comportamento quase constante em algumas faixas de amostras na figura 44, percebe-se que este padrão na verdade não é constante,variando aleatóriamente de valor em uma faixa de valores de tempo morto entre aproximadamente 15,80 segundos e 16,12 segundos. Caracterizando o efeito de jitter de comunicação.

Os dados estatísticos mostram:

O valore da função estatística mediana para o intervalo inteiro de amostras do processamento de dados e do valor desta função cujo domínio são as amostras da tabela 2 são quase idênticos,bem próximos uns dos outros,o que mostra que apesar dos picos de tempo morto durante o processamento de dados,a mediana do jitter se mantém quase constante para os dois domínios.

Os valores das funções estatísticas desvio padrão e valor médio absoluto para o intervalo inteiro do processamento de dados,coluna TABLE, e do valor destas funções cujo domínio são referentes ao selecionado na tabela 2,coluna SAMPLES, são bastante diferentes.O valores das funções de desvio padrão e valor médio absoluto cujos domínios são referentes a todo o intervalo de processamento,coluna TABLE, são muito mais elevados do que os respectivos valores do desvio padrão e valor médio absoluto cujos domínios são as amostras da tabela 2 excludente a última amostra.

Este efeito se caracteriza pelos picos aleatórios dos valores de tempo morto quando o domínio é relativo a todas as mostras do processamento de dados,levando a altos valores de desvio padrão e valor médio absoluto.Enquanto os valores de tempo morto quando o domínio é relativo as amostras da tabela 2 ,apesar de variarem aleatóriamente,estes ficam contidos em uma faixa de valores semelhantes,logo os valores das funções estatísticas desvio padrão e valor médio absoluto são relativamente baixos.

Realizado uma conjectura dos efeitos de jitter de comunicação dos testes 1 e 2,ou seja,teste somente com um motor e teste com três motores,temos que:

- Os valores das funções estatísticas mediana tanto para a coluna TABLE quanto para a coluna SAMPLES dos testes 1 e 2 são semelhantes.

- Os valores das funções estatísticas desvio padrão em relação as colunas TABLE é maior para o teste 2 em relação ao teste 1,porém os valores relativos as colunas SAMPLES são semelhantes nos dois testes,apesar do valor do desvio padrão do jitter no teste com três motores ser relativamente maior do que o valor do jitter do teste com 1 motor.

- Os valores das funções estatísticas valor médio absoluto em relação as colunas TABLE é maior para o teste 2 em relação ao teste 1,porém os valores relativos as colunas SAMPLES são semelhantes nos dois testes,apesar do valor do valor médio absoluto do jitter de comunicação no teste com três motores ser relativamente maior do que o valor do jitter do teste com 1 motor.

  Concluimos que a quantidade de motores acoplados no sistema influencia diretamente os valores das funções estatísticas relativas ao fenômeno de jitter de comunicação no processamento de dados.

| | TESTE 1 | | TESTE2 | |
|---|---|---|---|---|
| **STATISTIC FUNCTIONS** | **TABLE** | **SAMPLES** | **TABLE** | **SAMPLES** |
| **MEDIAN** | 15, 98 | 15, 97 | 15,98 | 15,97 |
| **PATTERN DEVIATION** | 696, 4903321 | 0, 064809218 | 1573,41 | 0,07 |
| **MEAN ABSOLUTE VALUE** | 63, 74006411 | 0, 058552113 | 2611,65 | 0,06 |

**Tabela 3 – Tabela Comparativa dos Valores das Funções Estatísticas Relativas ao Fenômeno de Jitter de Comunicação**

# 6. Máquina de Estados

A máquina de estados é definida como um autômato finito determinístico com saídas . Nesta definição, o sistema possui todos os estados definidos, desde seu estado inicial ao seu estado final, e todas as transições possíveis são conhecidas e determinadas por eventos que podem ocorrer no decorrer do processo . Estes eventos são dados provindos do monitoramento do sistema, ou seja, os transdutores e sensores . Estando o sistema em um determinado estado,caracterizado por um determinado conportamento dinâmico do robô LEO2, é verificado via uma rotina computacional se os dados provindos dos transdutores condizem ou não condizem com os dados dos possíveis eventos pré-definidos que podem ocorrer no determinado estado . Caso os dados processados relativos aos transdutores sejam condizentes com algum destes eventos, o autômato transita para um estado determinado relacionado com a ocorrência deste determinado evento. Caso os possíveis eventos para um estado não ocorram, o autômato permanece no estado. Estes eventos pré-definidos são chamados de releases do sistema de máquina de estados.

Em cada estado há uma determinada rotina computacional que designa um comportamento para o autômato, este comportamento identifica o estado, e após verifica a ocorrência ou não de algum release pré-determinado que designe uma transição de estados.

A máquina de estados pode ser de dois tipos, a Máquina de Mealy e a Máquina de Moore. A diferença entre os dois modelos é que na Máquina de Mealy, as saídas do sistema são conhecidas somente durante a transição entre estados devido a algum release, enquanto na Máquina de Moore, as saídas do sistema são definidas no estado, ou seja, cada estado possui uma saída.

## 6.1. Visão Geral

No projeto em questão, o robô LEO2 possui o trabalho de caminhar de forma bípede semelhante a humana sobre um trajeto circular numa dinâmica em duas dimensões conforme a um sistema de máquina de estado configurado em seu sistema de controle embarcado.

Quatro estados e quatro releases foram pré-definidos:

- Estados do sistema:

1. Estado NORMAL

Define que a operação da dinâmica em questão do robô pela trajetória circular está sendo implementada com sucesso, dentro dos parâmetros pré-definidos de angulação das juntas robóticas , os servos motores Dynamixels integrados. Este estado possui uma possível ocorrência de evento, o releaser FORA_DA_FAIXA.

2. Estado ERROR

Define que houve um comportamento de falha na dinâmica em questão do robô LEO2, indicada por algum dado de monitoramento das juntas robóticas que indique que alguma destas estão operando em um ângulo pré-estabelecido como fora da faixa de padrão do caminhar bípede do robô . Logo há a

indicação de uma queda  robô e o torque de todos os motores são desabilitados . O estado possui uma possível ocorrência de evento, o releaser BUTTON_PRESSED.

3. Estado GET_UP

Define um comportamento dinâmico em que o robô LEO2 começa a  realizar a dinâmica de se levantar após ter sofrido uma queda devida a alguma falha de atuação em sua trajetória . Possui uma possível ocorrência de evento, o release ARM_DOWN.

4. Estado RETURNING_ARM

Define um comportamento no qual o robô está no processo dinâmico de se levantar e posicionar suas juntas robóticas todas nas mesmas posições, afim de que este fique rente ao solo, para que possa reiniciar seu trabalho de caminhar sobre a trajetória circular . Dois possíveis eventos podem ocorrer neste estado, o releaser ERROR, caso alguma falha durante a dinâmica do processo de se levantar ocorra e o releaser ARM_UP, indicando que o processo da dinâmica de se levantar do robô LEO2 foi concluida com sucesso.

- Releases do sistema:

1. Release FORA_DA_FAIXA

Este evento é relativo aos estados NORMAL e RETURN_ARM. Pode ocorrer estando o autômato no estado NORMAL, caso a leitura dos transdutores indiquem que alguma junta robótica esteje fora da faixa de angulação pré-definida para o caminhar bípede. Este releaser também pode ocorrer estando o robô no estado RETURN_ARM, quando durante o processo de levantamento do robô, ocorrer algum erro de dinâmica do sistema.

2. Release BUTTON_PRESSED

Este evento é relativo ao estado ERROR. Indica que um determinado botão da placa expansora de canais OpenCM4.85 , entrada digital número 16, foi presionado, indicando a transição para o estado GET_UP.

3. Release ARM_DOWN

Este evento é relativo ao estado GET_UP. Indica que o servo motor referente  braço do robô está na posição pré-determinada como posição baixa. Este releaser indica a transiçao do estado GET_UP para o estado RETURN_ARM.

4. Release ARM_UP

Evento relativo so estado RETURN_ARM. indicando que o comportamento dinâmico do robô se levantar foi concluída com sucesso, ou seja, a posição do servo motor referente como braço do robô está na posição definida com o alta e o robô encontra-se de pé . Este release indica a transição do estado RETURN_ARM para o estado NORMAL.

## 6.2. Implementação

A máquina de estados implementada no sistema de controle embarcado do robô bípede LEO2 foi implementada como uma rotina no ambiente de programação do controlador principal . Já que este elemento do sistema é o que transmite e recebe dados seriais assíncronos para os servos motores Dynamixel e transdutores acoplados as placas de aquisição de dados 3Mxel,conforme o protocolo de comunicação do fabricante,transmitindo e recebendo dados conforme a tabela de controle dos dispositivos de recepção em um sistema de comunicação master-slave.

O estado inicial do sistema foi configurado como estado ERROR, no qual define que o robô está caido, já que alguma junta robótica está operando em alguma angulação fora da faixa pré-definida para a dinâmica .Neste estado, é escrito um comando no endereço do registrador da tabela de controle dos módulos Dynamixel relativo a habilitação do torque de saída do eixo dos servos, TORQUE_ENABLE, para que o torque de todos os servos motores sejam desabilitados . Neste comportamento , condizente com uma falha dinâmica no trabalho de percorrer o trajeto, é verificada se ocorre o releaser BUTTON_PRESSED . Caso ocorra este release, o robô transita para o estado GET_UP, caso não ocorra, o robô fica no estado ERROR . Devido a queima do microprocessador do controlador principal durante os testes dinâmicos do sistema, foi implementada uma função no programa da placa virtual que implementa o papel do botão de entrada digital da placa expansora OpenCM4.85 , rotina que é verificada a cada dez segundos, ou seja, a cada dez segundos o botão é apertado, transitando do estado ERROR para o estado GET_UP.

Estando no estado GET_UP, é realizada uma rotina para habilitar o torque do servo motor relativo ao braço do robô, conforme a escrita no endereço do registrador TORQUE_ENABLE. Neste estado, apenas o motor referente ao braço do robô está habilitado a realizar torque, enquanto os demais estão desabilitados.Neste estado,começa o procedimento dinâmico de levantar o robô,simulando que o braço está descendo afim de que o robô se apoie sobre o solo e comece a se levantar. É transmitido um comando para que este servo motor, do braço, realize a dinâmica para se posicionar em uma posição angular pré-definida como baixo, ou seja, noventa graus . É verificado a ocorrência do releaser ARM_DOWN,que indica que o processo de posicionamento do braço do robô na posição definida como baixa foi realizada com sucesso . Caso este releaser ocorra, o robô transita para o estado RETURN_ARM, enquanto este releaser não ocorre, o robô permanece no estado GET_UP.

Estando no estado RETURN_ARM, é habilitado o torque em todos os servos motores do robô e o ambiente de programação do controlador transmite um dado de ordem para que todos os servos motores se posicionem em zero grau, de forma a levantar o robô em posição base definida como em pé para o robô .Neste comportamento dinâmico,o braço do robô que está rente ao solo começa a se levantar para que o robô fique em pé. É verificado se houve a ocorrência de dois releases FORA_DA_FAIXA e ARM_UP. Caso verifique o release FORA_DA_FAIXA, indicando que durante a dinâmica de colocar o robô em pé houve alguma falha na dinâmica do sistema, o robô transita para o estado ERROR, onde fica até que seja verificado uma rotina de BUTTON_PRESSED que ocorre a cada dez segundos,conforme a função de simulação do botão digital. Caso seja verificado o release ARM_UP, indicando que o servo motor relativo ao braço do robô está posicionado na posição definida como alta, o robô transita para o estado NORMAL e o robô encontra-se na posição para implementar a locomoção bípede. Caso nenhum destes dois releases seja verificado, o robô permanece no estado RETURN_ARM.

Estando no estado NORMAL, têm-se definido que todos os servos motores possuem seus eixos de saída posicionados em ângulos condizentes com a faixa de angulação pré-definida para a dinâmica do caminhar do robô . Neste estado é realizado a rotina computacional em que todos os servos motores do sistema, excluindo o servo motor do braço do sistema estão com torques habilitados e o ambiente de programação do controlador transmite comandos para os servos motores Dynamixel conforme o

protocolo de comunicação do fabricante. Estes comandos por sua vez foram transmitidos ao ambiente do controlador pelo ambiente de programação do computador, conforme o protocolo de comunicação implementado como interface de comunicação entre o computador e o controlador do sistema embarcado.Neste estado,o sistema opera em mode de controle de velocidade em malha fechada. É verificada a ocorrência do release FORA_DA_FAIXA, indicando que houve falha na dinâmica do robô durante a sua trajetória circular, transitando para o estado ERROR, onde por sua vez, espera a ocorrência do release BUTTON_PRESSED. Caso não ocorra o release ERROR, o robô permanece no estado NORMAL e o processo de envio dos dados de comandos para os atuadores é continuado para que seja realizada a tarefa do caminhar bípede conforme os setpoints dos motores estabelecidos.

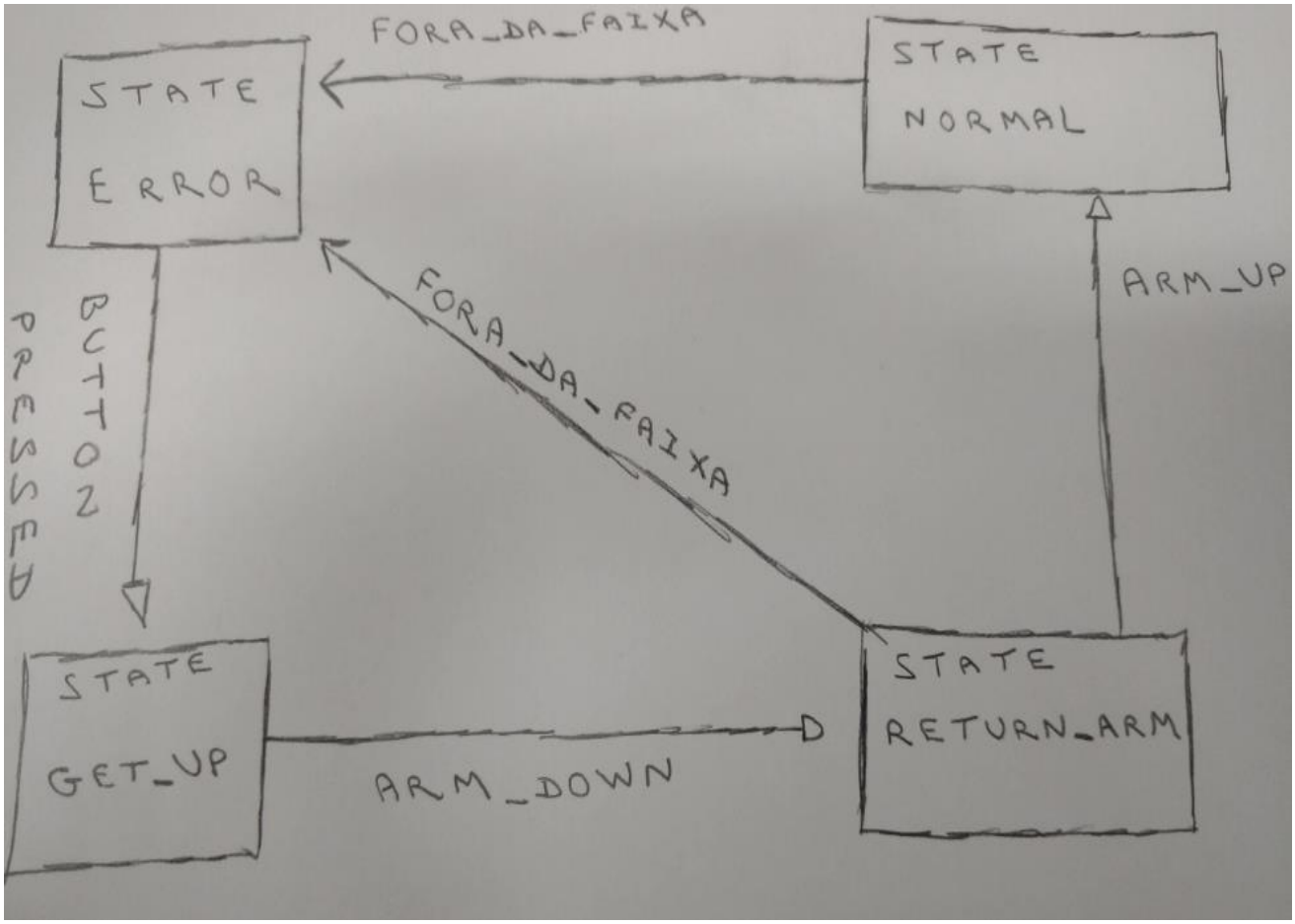


**Figura 46 – Diagrama esquemático da máquina de Estados do Robô LEO2**

| RELEASE STATE | FORA_DA_FAIXA | BUTTON_PRESSED | ARM_DOWN | ARM_UP |
|---|---|---|---|---|
| **ERROR** | X | GET_UP | X | X |
| **GET_UP** | X | X | RETURN_ARM | X |
| **RETURN_ARM** | ERROR | X | X | NORMAL |
| **NORMAL** | ERROR | X | X | X |

**Tabela 4 – Tabela de Transição de Estados**

## 6.3. Testes

A rotina de programação no ambiente da placa cotroladora virtual que implementa a máquina de estados do sistema de controle embarcado do robô bípede LEO2 foi testada com dois motores Mx-28,sendo um referente ao braço ID0 e outro ID1 referente as demais articulações robóticas diretas e o comportamento do sistema foi testado em duas configurações de estados complementares.

Primeiramente foi realizado o teste do sistema de controle embarcado operando como uma máquina de estados contendo dois estados e dois releases.Os estados configurados foram estado ERROR e NORMAL e os respectivos releases foram BUTTON_PRESSED e FORA_DA_FAIXA.Decorrendo conforme a configuração dos estados descrita anteriormente,porém com a diferença entre as transições de que estando no estado ERROR, ao ser precionado o botão 16 da placa de expansão OpenCM4.85exp ,ou seja,o evento que ocorre a cada 10 segundos,conforme a rotina referente ao ambiente de programação da placa virtual,há a transição para o estado NORMAL.

Observa-se que o sistema se inicializa no estado ERROR e após o período de 10 segundos há o ocorrimento do releaser BUTTON_PRESSED e há a transição do estado ERROR para o estado NORMAL. Variando manualmente a posição da alavanca do eixo de saída do motor ID1,colocando-o no estado FORA_DA_FAIXA,após 10 segundos ocorre o releaser BUTTON_PRESSED e este realiza a dinâmica de transitar para o estado NORMAL novamente.

Por fim, foram realizados testes com a máquina de estados configurada conforme descrita nas seções 6.1 e 6.2,ou seja,com quatro estados e quatro releases.

O sistema se inicia no estado ERROR.Após 10 segundos ocorre o releaser BUTTON_PRESSED.O autômato realiza a dinâmica de transição do estado ERROR para o estado GET_UP.Após passado um período,ocorre o releaser ARM_DOWN e o autômato realiza a dinâmica de transição do estado GET_UP para o estado RETURN_ARM.

Após realizar os testes sucessivas vezes não foi possível observar os dois possíveis releases que podem ocorrer no estado RETURN_ARM,ou seja,os releases FORA_DA_FAIXA ou ARM_UP,afim de que gerasse uma transição para ou o estado ERROR ou NORMAL.

Foi verificado manualmente o comportamento dos eixos de saída dos servomotores neste estado.A alavanca de saída do servo motor cuja identidade é ID0 encontrava-se com torque desabilitado,sendo possível mover seu eixo de saída,porém a alavanca de saída do motor de identidade ID1,estava travada e pouco variava a sua leitura de dados referente a posição.

Também foi verificado que normalmente durante os testes,os led de transmissão e recepção de dados,Rx-Tx, da placa de conversão de dados USBTODYNAMIXEL estavam ligados condizentes com a transmissão de dados,porém os leds Rx e Tx da placa expansora OpenCM4.85exp, na qual estão conectados os módulos Dynamixel do sistema,não estavam operantes.Algumas vezes ao passar algum tempo,o led Rx desta placa se mostrava ativo,mostrando que esta placa estava recebendo comandos,porém o led Tx não se encontrava no estado ativo,apesar de esperar um tempo decorrido.

Os leds dos módulos Dynamixel que indicam que os motores estão ligados mostrou-se ativo,concluindo que não houve falha ou danificação dos seus circuitos.

As rotinas computacionais da máquina de estados também foram revisadas e testadas diversas vezes com diversos setpoints e ganhos,porém não se encontrou erros e o comportamento do sistema persistiu.

Pode ser que este evento indique alguma falha de transmissão de dados pela placa expansora de dados OpenCM4.85exp para o computador.

# 7. CONSIDERAÇÕES FINAIS

## 7.1. Implementações Futuras

Durante a implementação do projeto não foi possível obter os dados de leituras dos transdutores conectados a placa 3Mxel,ou seja,os encoderes modelo RE-22 e os transudores de força modelo Flexforce. Este fato é devido a configuração da taxa de transmissão de dados da placa de aquisição e conversão de sinais dos transdutores,3Mxel,estar configurada para 921.000 bps,enquanto a taxa de transmissão de dados do sistema de controle embarcado é configurada como 1Mbps,o que gera uma diferença de oito por cento entre estas taxas.Sugere-se uma futura configuração da taxa de comunicação desta placa de aquisição de dados para 1Mbps,assim,será possível aquisitar os estados das articulações indiretas do robô,como posição e velocidade destas articulações e leituras de tensão provindas dos transdutores de força.Este fato leva a um melhor controle da dinâmica do robô,pois mais dados de monitoramento serão inferidos para a configuração da máquina de estados.Já que na configuração da máquina de estados implementada até o momento de LEO2 há referência somente a dados relativos ao monitoramento das juntas rotóticas diretas de LEO2,ou seja,posição e velocidade dos módulos Dynamixels integrados.

É sugerido o teste do sistema em seu modo de controle de torque,implementado em rotinas computacionais tanto no ambiente de programação do computador quanto do controlador principal porém que não foram testadas e analisadas.

Sugere-se também a instalação física dos resistores de terminação dos cabos de conexão entre o canal de quatro pinos para padrão RS-485 da placa expansora OpenCM4.85exp e o módulo Dynamixel que é conectado a este canal,afim de realizar testes em tempo real e verificar o comportamento do efeito de jitter durante o processo de comunicação da rede de transmissão de dados.Caso se reduza o efeito do jitter,espera-se medidas estatísticas mais otimizadas relativas ao tempo morto do sistema,gerando uma melhor performance da dinâmica do robô durante sua tarefa de caminhar bípede.

Também é sugerido a configuração de uma rotina de controle em malha fechada mais otimizada no ambiente de programação do computador,ou seja,abordar um outro algorítmo de controle mais eficiente ao invés do algorítmo de controle proporcional.

A configuração de uma maior quantidade de estados tende a obter uma melhoria da dinâmica do robô,já que este poderia ficar em estados intermediários,cujas configurações de posição e velocidade das articulações viabilizariam uma otimização no posicionamento do robô LEO2 durante os processos de transição de estados.

# 8. BIBLIOFRAFIA E REFERÊNCIAS

**[1] [Wikipédia]** - Disponível em: <https://pt. wikipedia. org/wiki/Wikip%C3%A9dia:P%C3%A1gina_principal> Acesso em: 29 nov. 2017

**[2] [Robotis]** - Disponível em: <http://www. robotis. us/> Acesso em: 02 dez. 2017

**[3] [Reninshaw]** - Disponível em: <http:// http://www. robotis. us/dynamixel/> Acesso em: 02 dez. 2017

**[4] [Tekscan]** - Disponível em: < https://www. tekscan. com/?utm_source=google&utm_medium=ccomputador&utm_term=tekscan&utm_content=eta1&utm_campaign=corporate> Acesso em: 04 dez. 2017

**[5] [Comparação entre protocolos de comunicação serial]** - Disponível em: <https://www. robocore. net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial. html#uar> Acesso em: 04 dez. 2017

**[6] [Serial Communication]**< https://learn. sparkfun. com/tutorials/serial-communication> Acesso em: 04 dez. 2017

**[7] [Maxim Integrated]** <https://www.maximintegrated.com/en/app-notes/index.mvp/id/763> Acesso em: 04 dez. 2017>

**[8] [National Instruments]** <http://digital. ni. com/public. nsf/allkb/32679C566F4B9700862576A20051FE8F> Acesso em: 04dez. 2017

**[9] [Athos Electronics]** - Disponível em: <http://athoselectronics.com/comunicacao-serial-arduino/> Acesso em: 06 dez. 2017

**[10] [TIA/EIA-485-A**] - Disponível em: <https://www.maximintegrated.com/en/app-notes/index.mvp/id/763> Acesso em: 02 dez. 2017

**[11] [Texas Instruments**] - Disponível em: <http://www.ti.com/lit/an/snla049b/snla049b.pdf> Acesso em: 03 dez. 2017

**[12] [TUdelft**] - Disponível em: <https://www.tudelft.nl/en/3me/organisation/departments/biomechanical-engineering/research/dbl-delft-biorobotics-lab/bipedal-robots/> Acesso em: 28 nov. 2017

**[13] [You tube**] - Disponível em: <https://www.youtube.com/watch?v=SBf5-eF-EIw> Acesso em: 04 dez. 2017

**[14]** - "The design of LEO: A 2D bipedal walking robot for online autonomous Reinforcement Learning". Schuitema et al., 2010

[Figura 8]- Obtido de <http://www. robotis. us/dynamixel/> Acesso em : 01 dez. 2017[Figura 9]- Obtido de <http://www. robotis. us/dynamixel/> Acesso em: 01 dez. 2017

[Figura 10]- Obtido de <http://www. robotis. us/dynamixel/> Acesso em: 01 dez. 2017

[Figura 11]- Obtido de <http://www. robotis. us/dynamixel/> Acesso em: 02 dez. 2017

[Figura 12]- Obtido de < http://www. robotis. us/dynamixel/> Acesso em: 02 dez. 2017

[Figura 13]- Obtido de < http://support. robotis. com/en/product/actuator/dynamixel/mx_series/mx-28at_ar. htm> Acesso 02 dez. 2017

[Figura 14]- Obtido de < http://support. robotis. com/en/product/actuator/dynamixel/mx_series/mx-28at_ar. htm> Acesso 02 dez. 2017

[Figura 15]- Obtido de < http://support. robotis. com/en/product/actuator/dynamixel/mx_series/mx-64at_ar. htm> Acesso 02 dez. 2017

[Figura 16]- Obtido de http://en.robotis.com/model/page.php?co_id=controller> Acesso 02 dez. 2017.

[figura 17]- Obtido de < http://support. robotis. com/en/product/controller/opencm9. 04. htm> Acesso 02 dez. 2017.

[figura 18]- Obtido de <http://support.robotis.com/en/product/controller/opencm_485_exp.htm> Acesso 03 dez. 2017

[Figura 21]- Obtido de http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm> Acesso 06 dez. 2017

[Figura 23]- Obtido de < http://athoselectronics. com/comunicacao-serial-arduino/> Acesso 06 dez. 2017.

[Figura 24]- Obtido de <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi> Acesso 06 dez. 2017.

[Figura 25]- Obtido de https://learn.sparkfun.com/tutorials/serial-communication> Acesso 06 dez. 2017.

[Figura 26]- Obtido de <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi> Acesso 06 dez. 2017.

[Figura 27]- Obtido de < http://mikrotik.tips/simplex-half-duplex-full-duplex/> Acesso 06 dez. 2017.

[Figura 28]- Obtido de < https://www.maximintegrated.com/en/app-notes/index.mvp/id/763> Acesso 07 dez. 2017.

[Figura 29]- Obtido de < https://www.maximintegrated.com/en/app-notes/index.mvp/id/763> Acesso 07 dez. 2017.

[Figura 30]- Obtido de <http://www.ti.com/lit/an/snla049b/snla049b.pdf> Acesso 03 dez. 2017.

[Figura 31]- Obtido de < http://support.robotis.com/en/product/controller/opencm9.04.htm > Acesso 04 dez. 2017.

[Figura 33]- Obtido de < http://support.robotis.com/en/product/controller/opencm9.04.htm > Acesso 04 dez. 2017.