MURILLO CANDIDO DE SOUSA

DESENVOLVIMENTO DE PROGRAMAS DE MONITORAMENTO E COMUNICAÇÃO PARA IMPLEMENTAÇÃO DE SENSORES NUMA UNIDADE DE REPARO POR ATRITO

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA MECÂNICA

2018

MURILLO CANDIDO DE SOUSA

DESENVOLVIMENTO DE PROGRAMAS DE MONITORAMENTO E COMUNICAÇÃO PARA IMPLEMENTAÇÃO DE SENSORES NUMA UNIDADE DE REPARO POR ATRITO

Monografia apresentada ao Curso de Graduação em Engenharia Mecatrônica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **ENGENHEIRO MECATRÔNICO.**

Área de Concentração:

Instrumentação e automação industrial.

Orientadora: Profa. Dr.-Ing. Vera Lúcia D. S. Franco

UBERLÂNDIA – MG

2018

AGRADECIMENTOS

Agradeço primeiramente a meus familiares, meus pais Deuzeli e Tatiane e meu irmão Mateus Gabriel, os quais formam a minha base e sempre me incentivaram a estudar, mesmo com todas as dificuldades.

À Universidade Federal de Uberlândia pelo oferecimento de um ensino de qualidade.

À minha orientadora, Professora Vera Lucia D. Sousa Franco, por acreditar em mim, me dar todo o suporte necessário nessa jornada até minha graduação e me fornecer através deste Projeto uma aprendizagem prática.

Ao engenheiro Denis Soares de Freitas, pelo auxílio, pelos conhecimentos passados e paciência durante a execução deste Projeto.

A todos os amigos e companheiros da Universidade Federal de Uberlândia, em especial a equipe do Laboratório de Tecnologia em Atrito e Desgaste por serem sempre muito solícitos, em especial às pessoas que estiveram comigo nesses dias.

Às pessoas mais próximas que direta ou indiretamente fizeram parte da minha formação, em especial a Eduardo Vitorino.

À Petrobras pelo apoio financeiro viabilizando assim o desenvolvimento do projeto.

Resumo

O setor industrial de energia e transportes é a base para a infraestrutura de uma nação, por isso busca-se a utilização de aplicações que evitem custos adicionais devido a defeitos e falhas. Nesse contexto é que se insere o uso da técnica de reparo de trincas por atrito em estruturas *offshore*, em contrapartida ao método tradicional de reparo por solda a arco elétrico que pode sofrer fragilização por hidrogênio em ambientes submersos. A técnica de Processamento de Pinos por Atrito (PPA) resume-se em realizar um furo na região trincada e o preenchimento do mesmo com o uso de um pino, submetido a condições de velocidade de rotação e força compressiva contra o furo, gerando uma união metalúrgica entre as partes. Um convênio entre a Petrobras e a Universidade Federal de Uberlândia (UFU) permitiu a criação no Laboratório de Tecnologia em Atrito e Desgaste (LTAD) de uma estrutura para operações em campo capaz de desenvolver essa técnica de manutenção de estruturas offshore, chamada de Unidade de Processamento de Pinos por Atrito – UPPA X. A UPPA X consiste numa unidade automatizada comandada por um computador dedicado e controlada por um Controlador Lógico Programável (CLP) Bosch Rexroth L25®. Esta monografia descreve a implementação da célula de carga HBM U2B® e o sensor de temperatura infravermelho Raytek MI, dispositivos que possibilitam a realização de testes e monitoramento da Unidade. Para detecção automática, comunicação e configuração dessa instrumentação foram desenvolvidas rotinas baseada em LabView® e também a partir do CLP.

Palavras-chave: Reparo por Atrito, Unidade de Processamento de Pinos por Atrito, Controle, Instrumentação, Automação

Abstract

The industrial sector of energy and transport is the basis for the infrastructure of a nation, so it is sought the use of applications that avoid additional costs due to defects and failures. In this context, the use of solid-state welding technique such as friction welding in offshore structures is inserted, in contrast to the traditional method of fusion welding repair that can become fragile by hydrogen in submerged environments. The Friction Hydro Pillar Processing (FHPP) consists of drilling a substrate in the cracked region and filling it with a rod submitted to certain conditions of rotation speed and axial force into the cavity generating a metallurgical union between the pieces. An agreement between the company Petrobras and the Federal University of Uberlandia - Brazil has enabled the creation of a Friction Hydro Pillar Processing Machine for field operations able to develop this technique of maintenance of offshore structures in the Laboratory of Technology at Friction and Wear (LTAD). It consists of an automated unit commanded by a dedicated computer and controlled by a Programmable Logic Controller (PLC) Bosch Rexroth L25. This monograph describes the implementation of the HBM U2B® load cell and the Raytek MI infrared temperature sensor, devices that enable the tests performing and the machine monitoring. Routines based on LabView® and also were from the PLC developed looking for automatic detection, communication and settings of this instrumentation.

Keywords: *Friction Repair, Friction Hydro Pillar Processing Machine, Control, Instrumentation, Automation.*

LISTA DE FIGURAS

LISTA DE TABELA

LISTA DE ABREVIAÇÕES

ASCII	*American Standard Code for Information Interchange*
BPS	Bits por segundo
CD	*Carrier Detect*
CLP	Controlador Lógico Programável
CTS	*Clear To Send*
DCE	*Data Communications Equipment*
DSR	*Data Set Ready*
DTE	*Data Terminal Equipment*
DTR	*Data Terminal Ready*
EIA	*Electronics Industry Alliance*
FHPP	*Friction Hydro Pillar Processing*
FPSO	*Floating Production Storage and Offloading*
GND	Terra
LD	*Ladder*
LTAD	Laboratório de Tecnologia em Atrito e Desgaste
PCI	*Peripheral Component Interconnect*
RS-232	*Recommended Standard* - 232
RS-485	*Recommended Standard* - 485
RTS	*Request To Send*
ST	Texto Estruturado
UART	*Universal Asynchronous Receiver and Transmitter*

UPPAX	Unidade de Processamento de Pinos por Atrito X
ZAC	Zona afetada pelo calor

SUMÁRIO

CAPÍTULO I

INTRODUÇÃO

A extração e o transporte de óleo e gás em mar aberto são feitos com a utilização de plataformas que servem como base de instalação dos equipamentos de extração, conhecidas como plataformas *offshore* (LUCZYNSKI, 2002). Essas instalações possuem, além de altos riscos, elevados custos de operação, ainda mais quando se trata em aplicações em águas profundas (FREITAS, 2014). Em média, a manutenção *offshore* é vinte vezes mais cara que uma manutenção em estruturas terrestres, conhecidas como *onshore* (CAIXETA, 2011).

As estruturas *offshore* estão sujeitas ao constante surgimento de trincas de fadiga em uniões soldadas em decorrência dos movimentos das ondas, as quais introduzem carregamentos e tensões significativas nas estruturas (LOTSBERG; LANDET, 2005). Nas paredes de um tanque de armazenamento de petróleo em uma plataforma semi-submersível (FPSO – *Floating Production Storage and Offloading*) da Petrobras foi encontrada uma trinca, que é mostrada na Fig. 1.1, resultante de fadiga de baixo ciclo (SOUZA, 2006).

Figura 1.1 - Tanque trincado de armazenamento de um FPSO da PETROBRAS (SOUZA, 2006).

Comumente, utilizam-se técnicas de soldagem a arco elétrico no reparo de estruturas, porém estas técnicas estão sujeitas a fragilização devido a dissolução de hidrogênio na zona de fusão. Em ambientes submersos há a dissociação de vapor d'água quando submetidos a esses arcos elétricos, criando ali uma atmosfera rica em hidrogênio (SILVA, 1999).

O processo de reparo por atrito (ou FHPP – *Friction Hydro Pillar Processing*) se desenvolve como uma técnica alternativa na busca por práticas mais seguras e versáteis, e que ainda minimizem (ou até mesmo anulem) as necessidades de paradas de produção nas FPSOs. Esse processo que ocorre abaixo da temperatura de fusão (estado sólido) reduz problemas críticos como trincas, porosidades, adsorção de gases e contaminação (SALAMA; LOTSBERG, 2004).

A prática de reparo por atrito, ilustrada na Fig. 1.2, fundamenta-se no preenchimento de um furo de geometria definida (cônica ou cilíndrica) por meio de uma introdução coaxial de um pino consumível, estando este submetido a uma velocidade de rotação e esforços de compressão contra a cavidade do furo (PIRES, 2007).

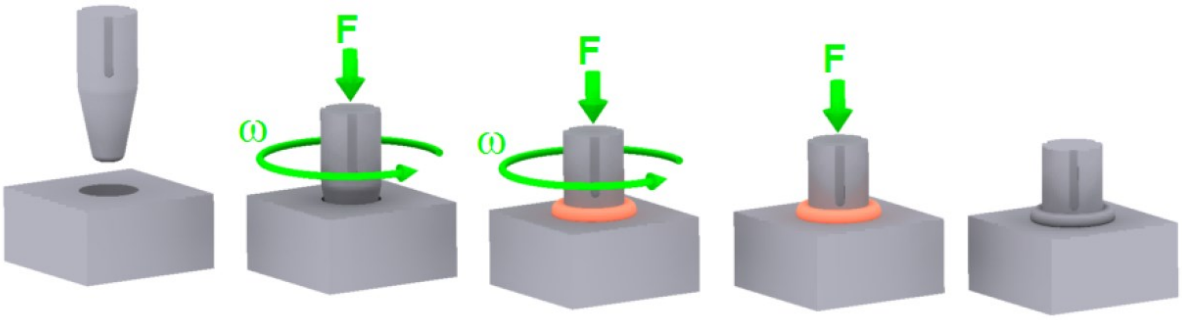


Figura 1.2 – Esquema ilustrativo do processo FHPP (extraído de FREITAS, 2014).

O Laboratório de Tecnologia em Atrito e Desgaste (LTAD) da Universidade Federal de Uberlândia (UFU) explora a utilização dessa técnica através de Unidades de Processamento de Pinos por Atrito (UPPA) que possuem como objetivo a verificação de diferentes faixas de operações. Desde então já foram construídas 5 versões da Unidade (UPPA1, UPPA2, UPPA3, UPPA4 e UPPAX), sendo a primeira UPPA1, esquematizada na Figura 1.3, e a última UPPAX, a ser utilizada no presente trabalho.

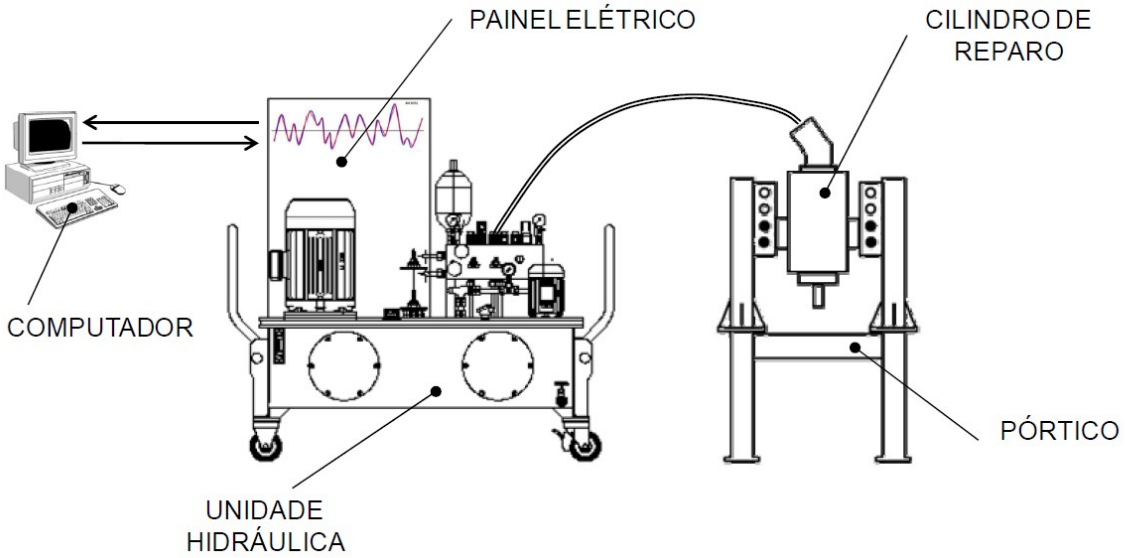


Figura 1.3 - Esquema ilustrativo da Unidade de Processamento de Pinos por Atrito 1 (UPPA1) (HWANG, 2010).

A última versão (UPPAX) utilizada no presente trabalho possui todo o sistema de controle implementado a partir de um CLP (Controlador Lógico Programável)

IndraControl L25®. Essa implementação de controle só é possível através de uma instrumentação para atuar corretamente nos ensaios de FHPP. Assim, necessitou-se conhecer melhor os sensores utilizados e seus protocolos de comunicação envolvidos.

Nesse contexto de comunicação entre sensores e atuadores é que se inserem padrões de transferência de dados. O padrão RS-485 é utilizado neste trabalho criando uma rede industrial de comunicação que estabelece conexão mais robusta entre sensores e o módulo do CLP para transmissão de dados seriais, R-IB IL RS 485/422 PRO(-2MBD)-PAC® Bosch Rexroth.

A criação de rotinas para realizar a comunicação desse módulo do CLP foi proposta com o objetivo de implementar: uma célula de carga que utiliza de um condicionador de sinal num padrão de comunicação serial RS-485 e; um sensor de temperatura infravermelho que por meio de um circuito eletrônico transmite sinal no padrão de comunicação RS-232.

1.1 Objetivos

O objetivo principal deste trabalho é a implementação e operacionalização da célula de carga HBM U2B® para verificação da força aplicada por um cilindro hidráulico e comparar com o valor que é medido indiretamente por sensores de pressão instalados nas linhas ligadas às câmaras do cilindro. Como a temperatura do material processado também é uma grandeza importante para as propriedades mecânicas da união resultante, é também objetivo do presente trabalho a implementação na lógica de controle do CLP da comunicação com o sensor de temperatura infravermelho RAYMID10LT® Raytek MI Sensor.

1.2 Justificativa

A necessidade de instrumentação e automação adequadas para dar suporte aos ensaios de FHPP é que motivou a inserção de uma rotina no CLP atual da unidade UPPAX, para que mostrasse a força aplicada pelo cilindro hidráulico. A quantificação do erro de medição por meio do uso de sensores de pressão que é afetada por perdas de carga no sistema hidráulico, é a principal função da célula de carga implementada.

Alguns ensaios de FHPP ocorrem com pré-aquecimento do material e, nestes ensaios, é necessário verificar o aporte térmico do material processado. Para se mensurar esse aporte térmico é necessário conhecer, dentre outras grandezas, a temperatura atingida com o pré-aquecimento. O sensor infravermelho tem o papel de medir essa temperatura na superfície do material. A automatização desta medição de temperatura no CLP, via comunicação serial, evita erros de uma leitura manual.

CAPÍTULO II

FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata de aspectos do sistema UPPAX como: sistema hidráulico e instrumentação. Também são abordados conceitos de protocolos de comunicação serial RS-232 e RS-485 que interagem no processo transmitindo dados dos sensores utilizados para a UPPAX e a codificação ASCII. Por fim são abordadas linguagens de programação utilizadas no CLP Rexroth IndraControl L25®: *Ladder* e Texto Estruturado (ST- *Structured Text*).

2.1 Unidade de Processamento de Pinos por Atrito X (UPPAX)

O processamento de pinos por atrito, conhecido internacionalmente como *Friction Hydro Pillar Processing* (FHPP), trata-se de uma técnica na qual o movimento relativo entre duas partes em estado sólido submetidos a esforços compressivos gera calor através do atrito ocasionado, remetendo o material destas partes a um regime plástico na superfície de contato. Após a execução do processo, as superfícies formam uma união metalúrgica entre elas (AWS, 1991).

A rotação é utilizada como método para a obtenção do atrito e calor entre as superfícies. A Figura 2.1 esquematiza o processo.

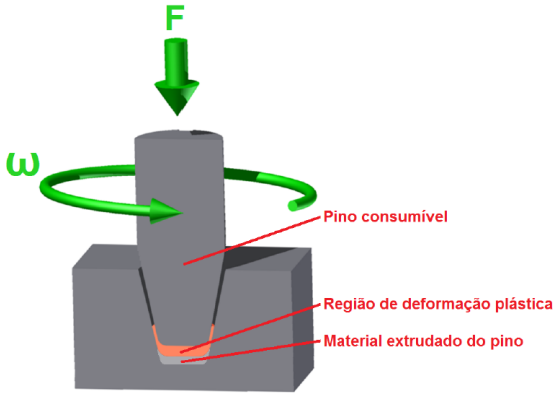


Figura 2.1 – Processo de pinos por atrito esquematizado (FREITAS, 2014).

Alguns parâmetros são importantes durante o processo e sua análise ajuda a otimizar os resultados obtidos. Esses parâmetros são a velocidade de rotação, força axial, taxa de queima, tempo de aquecimento, velocidade de frenagem e força de forjamento (HWANG, 2010).

A velocidade de rotação é um parâmetro obtido pela rotação do próprio eixo e que embora seja muito importante ao processo, não apresenta alterações significativas na qualidade da solda para uma grande faixa de variação desta rotação (AWS, 1991). Portanto utiliza-se uma velocidade obtida como ótima para cada conjunto de material e aplicação (HWANG, 2010).

A força axial tem influência direta na espessura e em características microestruturais da zona afetada pelo calor (ZAC). Este parâmetro também é responsável pelo controle da variação de temperatura na região da solda, pela potência requerida do equipamento e pela taxa de queima (HWANG, 2010).

A estipulação da velocidade de consumo de peças durante o processo de soldagem é conhecida como a taxa de queima. Este parâmetro, que determina o início e o fim do ciclo de solda, é significativamente influenciado pela combinação entre força axial e velocidade de rotação e também tem grande importância na qualidade da união (PIRES, 2007). A relação existente entre os parâmetros do processamento de pinos por atrito está representada na Fig. 2.2, onde também são apontadas as suas influências em propriedades mecânicas, como dureza na ZAC (zona afetada pelo calor) e limite de escoamento, do material resultante.

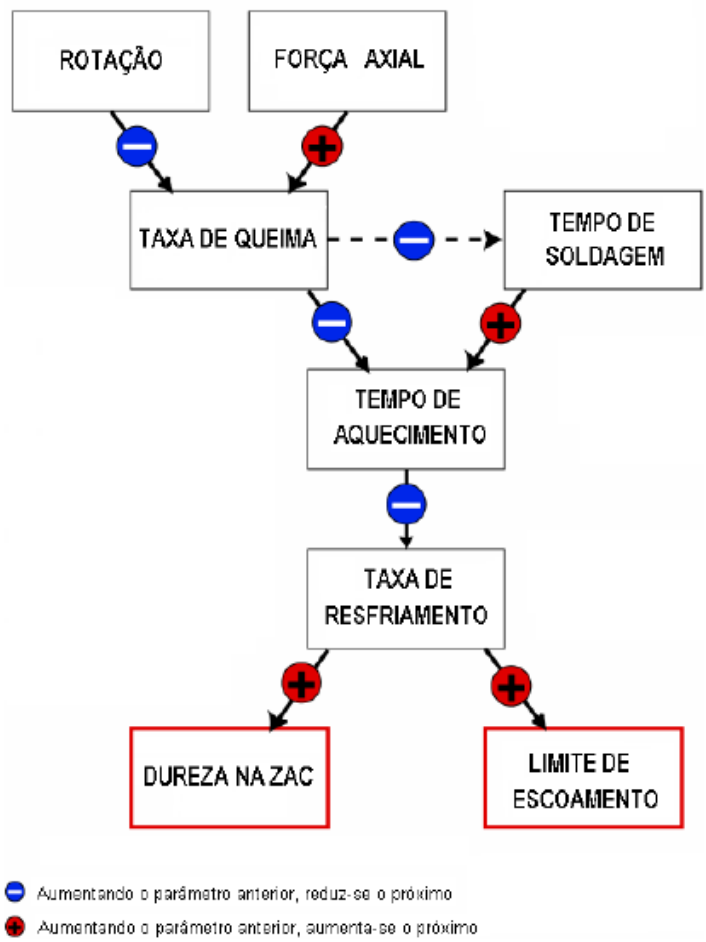


Figura 2.2 – Influência dos parâmetros na soldagem por atrito (MEYER, 2004)

Os parâmetros força axial e rotação atuam devido a um sistema hidráulico existente. Este sistema hidráulico é abordado a seguir.

2.2 Sistema Hidráulico

O sistema hidráulico do UPPA X é composto basicamente por 3 elementos, conforme ilustrados na Fig. 2.3: Cilindro de Reparo, Unidade Hidráulica e Bloco de Válvulas.

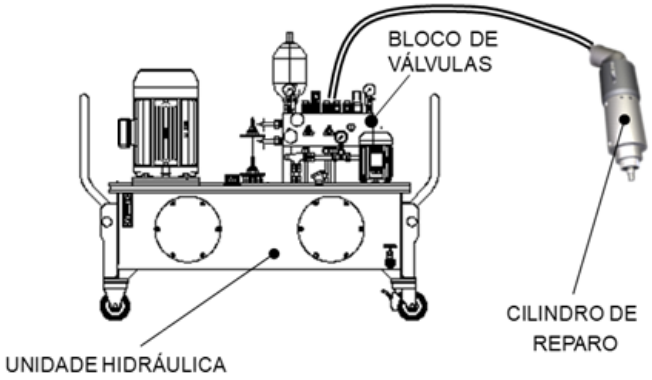


Figura 2.3 – Sistema hidráulico de uma Unidade de Processamento de Pinos por Atrito (editado de HWANG, 2010).

Cilindro de Reparo: unidade responsável pela aplicação, conjunta ou não, dos esforços axiais e da rotação no pino processado durante os processos de reparo. Esta unidade é composta por um cilindro hidráulico de haste vazada vinculado a um motor hidráulico (FORMOSO, 2012)

Unidade Hidráulica: encarregada de prover a energia a ser utilizada pelo Cilindro de Reparo no processo. Três conjuntos moto-bomba presentes nesta Unidade oferecem as condições necessárias ao Cilindro de Reparo: um primeiro conjunto com pressão e vazão fixas propicia rotação e torque ao eixo; um segundo conjunto também com pressão e vazão fixas fornece força axial ao eixo e possibilita seu deslocamento axial; e um terceiro conjunto se encarrega de filtrar e reconduzir o óleo (FORMOSO, 2012).

Bloco de Válvulas: este bloco é responsável pela regulagem do fluxo e da pressão obtida na Unidade Hidráulica. Válvulas proporcionais servo controladas e sensores de monitoramento de pressão na entrada e saída no cilindro de reparo compõem o bloco (HWANG, 2010).

A Figura 2.4 mostra a interação dos elementos que configuram o sistema hidráulico da UPPA.

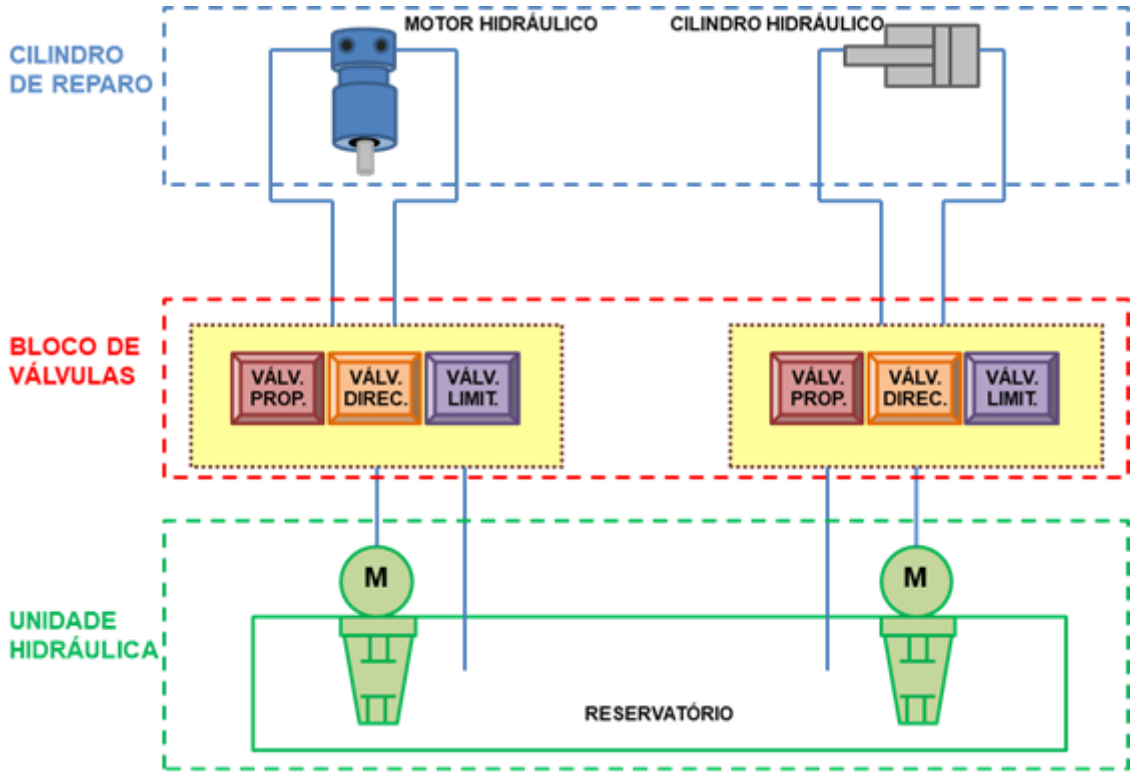


Figura 2.4 – Sistema Hidráulico da UPPA X (FORMOSO, 2012).

2.3 Instrumentação

A ciência que desenvolve e aplica técnicas de medição, indicação, registro e controle de processos de fabricação é chamada de instrumentação (PAVANI, 2011). As grandezas físicas envolvidas nesses processos de fabricação podem ser deslocamento, pressão, fluxo, força, nível, temperatura, tensão e corrente elétrica.

O presente trabalho tem como objetivo de estudo a medida da força compressiva, por uma célula de carga (HBM U2B® 50kN), e a medida da temperatura, por meio um sensor infravermelho (Raytek RAYMID10LT®).

2.3.1 Célula de Carga

O transdutor que transforma a grandeza física força em sinal elétrico, utilizado em balanças, pesagem industrial, aplicações de automação e controle em processos industriais é chamado de célula de carga. Seu funcionamento ocorre devido a uma deformação sofrida proveniente de uma força aplicada externamente (LADEIRA, 2011). O sensor utilizado para converter essa deformação em sinal elétrico é o extensômetro (*Strain Gauge*).

O extensômetro, como mostrado na Fig. 2.5, parte do princípio que ao sofrer deformação os metais também experimentam alteração em outras propriedades físicas, no caso a resistência elétrica (ANDOLFATO; CAMACHO; BRITO, 2004). A lei de Hooke (1678), mostrada na Equação (2.1), relaciona a tensão (σ) no extensômetro com um produto entre a deformação (ε) do mesmo e o seu módulo de elasticidade (E). Variações da resistência elétrica do extensômetro são diretamente proporcionais a essa deformação sofrida, enquanto a tensão é também proporcional à força aplicada possibilitando, portanto, uma relação entre essas grandezas físicas.

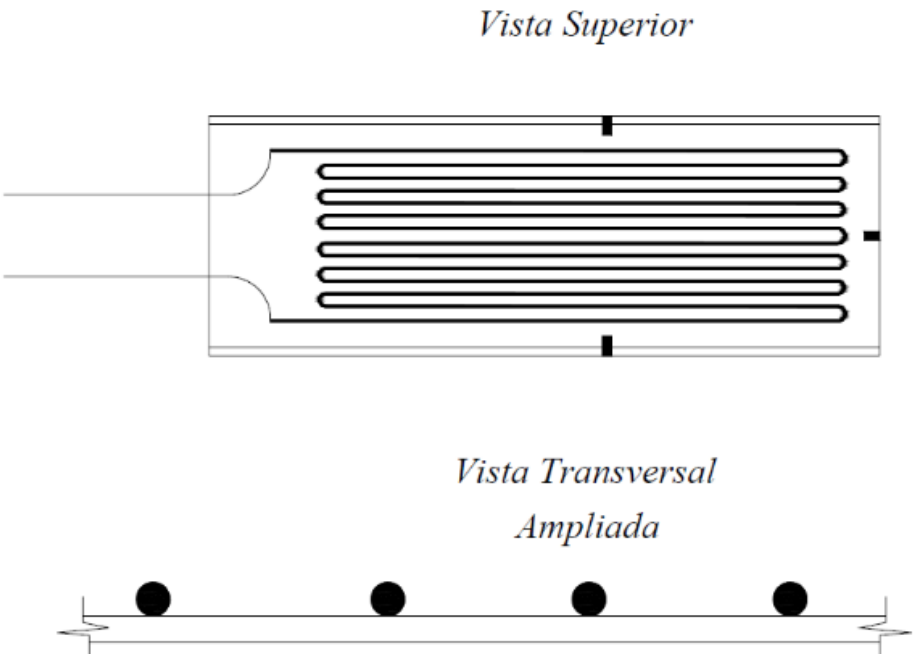


Figura 2.5 – Extensômetro de resistência a fio (ADOLFATO; CAMACHO; BRITO, 2004).

$$\sigma = E \times \varepsilon \qquad (2.1)$$

Estes extensômetros como o da Fig. 2.5 necessitam ser aplicados em circuitos elétricos especiais, de modo que as deformações sofridas gerem uma variação da resistência mensurada e, consequentemente, uma alteração equivalente da tensão elétrica (ADOLFATO; CAMACHO; BRITO, 2004). Os referidos circuitos elétricos são conhecidos como Ponte de *Wheatstone* (*Wheatstone's Bridge*).

Criado em 1843 pelo físico britânico Sir Charles Wheatstone, o circuito de ponte, como o da Fig. 2.6, para medir resistências elétricas é considerado o circuito apropriado para verificar e mensurar as variações observadas nos extensômetros (<http://www.celuladecarga.com.br>, 2018).

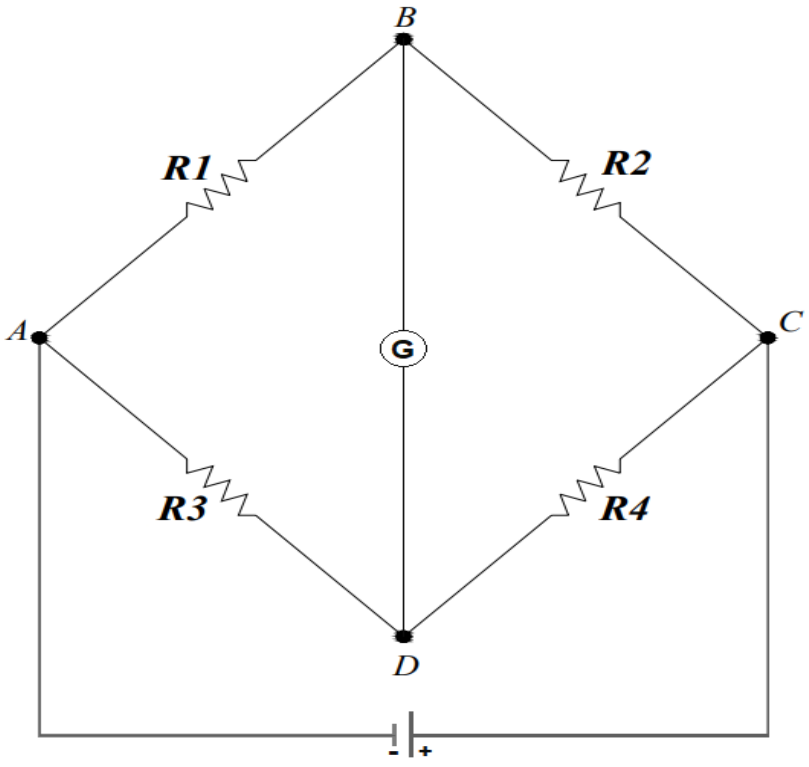


Figura 2.6 – Ponte de *Wheatstone* com galvanômetro central (alterado de ADOLFATO; CAMACHO; BRITO, 2004).

À topologia de ponte, então, são conectados os extensômetros nas posições das resistências e para qualquer força suportada pela célula de carga, há uma deformação correspondente que gera uma variação de resistência na ponte, resultando assim num sinal elétrico. Este sinal elétrico observado é medido no galvanômetro central da ponte de *Wheatstone* e tratado para que possa ser amplificado, filtrado, amostrado e/ou convertido ao padrão de comunicação (ou protocolo) adequado para a leitura do usuário, conforme ilustra a Fig. 2.7.

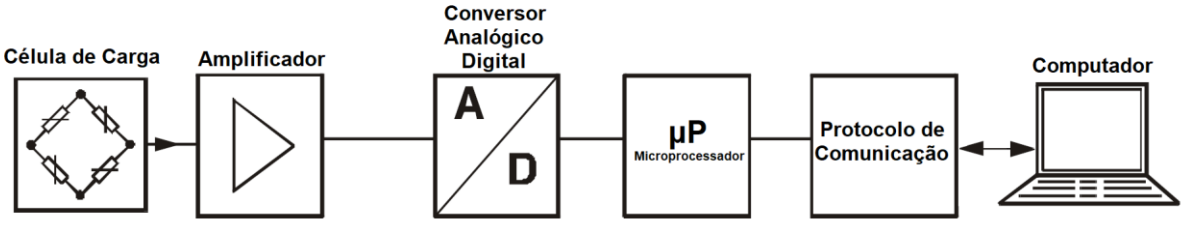


Figura 2.7 – Condicionamento de sinal de medição da célula de carga.

2.3.2 Sensor de Temperatura Infravermelho

A temperatura é uma grandeza física definida como abstrata e está relacionada ao grau de agitação de moléculas de um material, que varia e se relaciona, então, com a energia interna dessas moléculas. Enquanto a maioria dos sensores utilizados para a medição de temperatura baseiam-se num contato direto com o material, há sensores que utilizam da radiação emitida pelo corpo para realizar a inferência da temperatura. Esses sensores são conhecidos como pirométricos e possuem vantagens como eliminar problemas de estabilidade, realizar medições não destrutivas e evitar falhas de isolamento que assolam as medições de temperaturas muito elevadas (DALLY; RILEY; MCCONNELL, 1993).

Estes sensores baseiam-se no princípio que de acordo com o grau de agitação das moléculas (temperatura) do material que compõe um corpo, haverá emissão de energia na forma de radiação eletromagnética. A lei de Boltzmann euncia que essa energia (S) é igual à quarta potência da temperatura (T) multiplicada de uma constante, chamada de constante de Boltzmann (σ), como a Equação (2.2) mostra (YOUNG; FREEDMAN, 2015). A radiação infravermelha situa-se na área de

comprimentos de onda do espectro magnético, compreendida entre 0,78μm e 1000μm, e são preferidos por conterem a maior emissividade significativa para temperaturas até 700ºC (DALLY; RILEY; MCCONNELL, 1993).

$$S = \sigma \times T^4 \tag{2.2}$$

O princípio de funcionamento desses sensores, exposto na Fig. 2.8, é a concentração da radiação infravermelha emitida por uma determinada área do alvo (corpo observado) a um ponto específico do sensor, através do uso de lentes colimadoras e um espelho côncavo para a focalização. A energia nesse ponto é proporcional a temperatura média observada na área do alvo e é capaz de produzir um sinal elétrico com o auxílio de um transdutor, instalado no ponto para onde a radiação é focalizada.

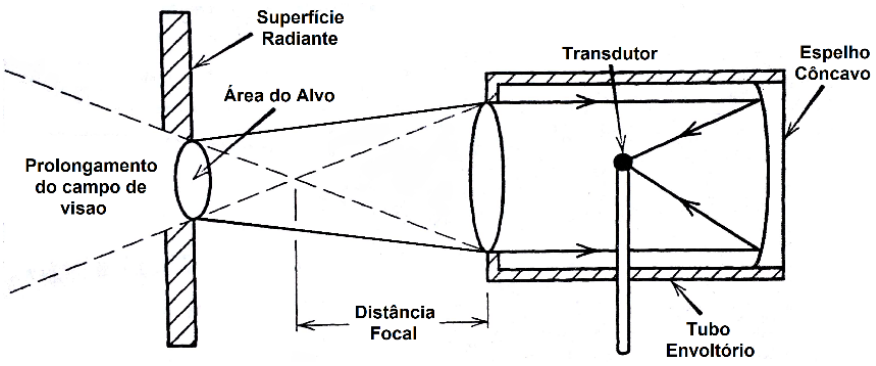


Figura 2.8 – Princípio de funcionamento de um sensor pirométrico (modificado de DALLY; RILEY; MCCONNELL, 1993).

Duas características são denominadas críticas para o funcionamento do sensor: foco da lente e emissividade do material. O foco da lente é que vai determinar a que distância do alvo o sensor deverá ficar para se medir a temperatura da melhor área útil do objeto observado. A emissividade do material, no qual a temperatura é medida, é fundamental para que se faça um ajuste na medição a fim de reparar a distorção do valor medido, pois as teorias de radiação eletromagnéticas são baseadas

na irradiação do corpo negro (ideal). A situação real dos objetos medidos é de uma emissividade aproximadamente igual ou menor que a ideal, portanto é desejável que este parâmetro seja previamente conhecido.

O sinal obtido no transdutor é então tratado (amplificado, filtrado e/ou amostrado) e convertido ao padrão de comunicação RS-232, para que possa se comunicar com uma unidade de processamento (computador e/ou CLP).

2.4 Comunicação Serial

A comunicação serial trata-se do tipo de comunicação que acontece numa linha de transmissão, onde os bits são enviados um após o outro (*bit-serial*). Os dispositivos eletrônicos processam dados em microprocessadores no modo paralelo, por isso, como na Fig. 2.9, um transmissor realiza a conversão paralelo-serial, enquanto que o receptor realiza a conversão serial-paralelo. Devido às possíveis altas taxas de transmissão (*baud rates*), às reduções de custos e às facilidades de instalação, possibilitou-se uma maior aceitação e amplo uso deste tipo de comunicação.

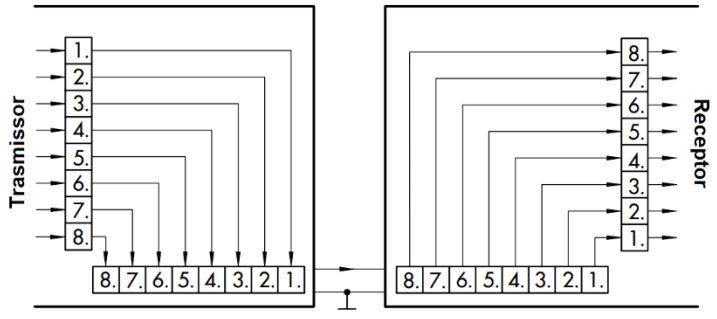


Figura 2.9 – Esquema simples de dois fios para transmissão de dados serial.

A comunicação serial pode se caracterizar pela direção do fluxo de dados (*data flow*), quantidade de dispositivos se comunicando na rede e pela velocidade de transmissão. Quanto ao fluxo de dados, ilustrado na Fig. 2.10, tem-se: *simplex*

(transmissão unidirecional), *half duplex* (uma transmissão por vez) e *full duplex* (transmissão bidirecional).

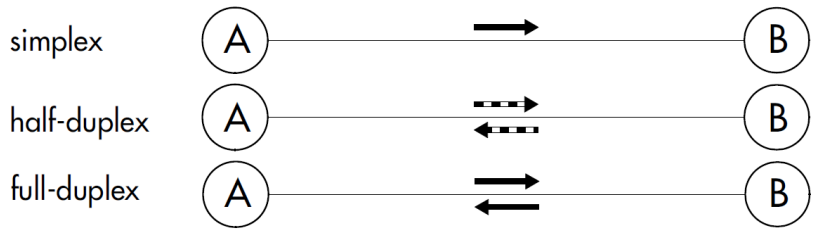


Figura 2.10 – Fluxo de dados da comunicação serial.

A transferência de dados pode ser composta de apenas dois dispositivos comunicando entre si, chamada de conexão ponto a ponto, ou vários dispositivos, denominada rede de comunicação. A Figura 2.11 expõe a conexão ponto a ponto com duas linhas de transmissão independentes e numa ligação cruzada: o transmissor no primeiro dispositivo se conecta ao receptor do segundo e vice-versa. A Figura 2.12 mostra a rede de comunicação com um meio de transmissão num fio único por onde sinais são transmitidos e recebidos.

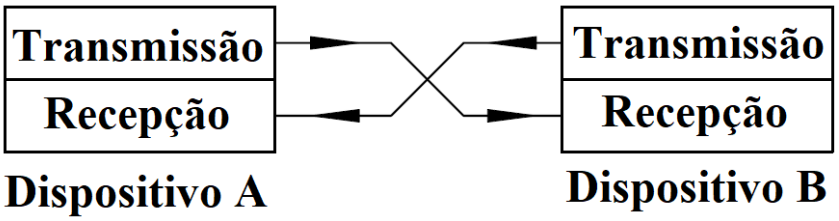


Figura 2.11 – Conexão ponto-a-ponto entre dispositivos A e B.

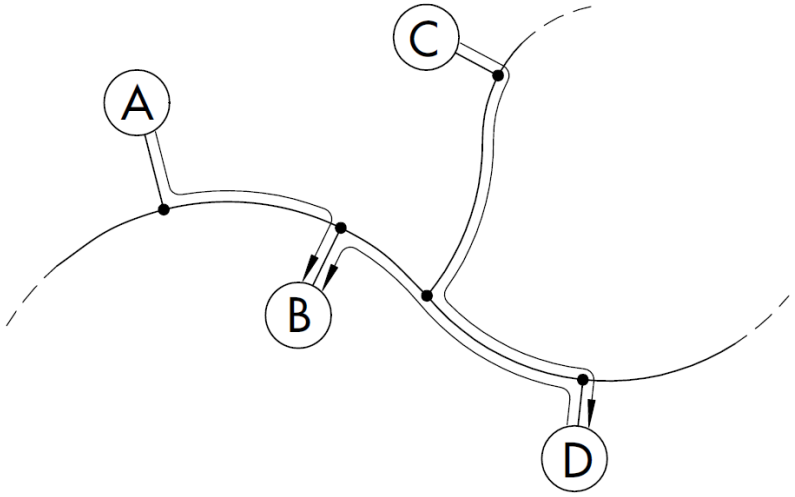


Figura 2.12 – Rede de transmissão com os dispositivos A, B, C e D.

Um critério essencial para a determinação da capacidade de transmissão de uma linha de transmissão de dados é a taxa de transmissão de dados, ou seja, a velocidade com que os dados podem ser transmitidos. Essa taxa de transmissão é caracterizada pela quantidade de bits transmitidos por segundo e é medida em bits por segundo (BPS). Em casos de altas taxas de transmissão as unidades de medida podem ser quilobit por segundo (kbit/s) e megabit por segundo (Mbit/s). Como os dados na transmissão serial são enviados por meio de uma comutação de tensão na linha de transmissão, essa taxa de comutação, chamada de *Baud Rate*, é que se torna o fator importante para esta comunicação.

A comunicação serial pode ser ainda caracterizada quanto a técnica de transmissão utilizada entre os dispositivos receptor e transmissor: síncrona ou assíncrona. No caso de uma transmissão síncrona, um sinal de *clock* precisa também ser transmitido através de outra conexão isolada ou pode ser derivado do sinal de dados.

Na transmissão assíncrona, nenhum sinal de *clock* é transmitido, então mesmo se usando dispositivos na mesma frequência, uma pequena diferença pode atrapalhar o funcionamento em sincronia. Busca-se reduzir essa falha sincronizando o receptor com a frequência do transmissor em intervalos que devem ser os mais curtos possíveis. A sincronização ocorre no início de cada caractere marcado com um bit adicional de início e parada. Um caractere é denominado UART (*Universal*

Asynchronous Receiver and Transmitter) quando é usado para esse propósito, conforme ilustrado na Fig. 2.13.

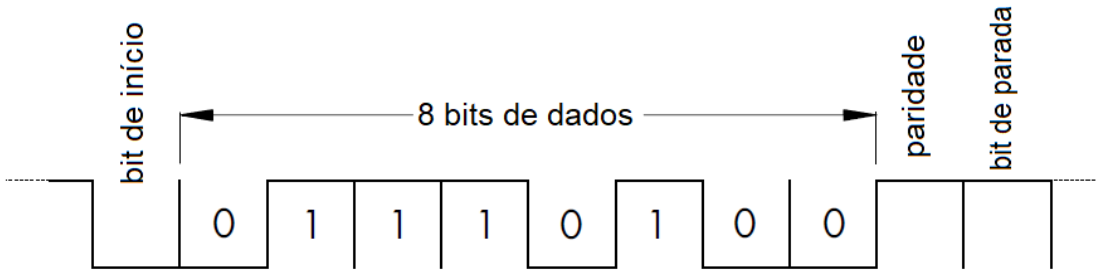


Figura 2.13 – Transmissão assíncrona utilizando o caractere UART.

Na primeira borda do sinal do bit de início o receptor sincroniza seu *clock* interno com os dados de recepção. Os bits a seguir são amostrados no meio do tempo de bit. Após os sete ou oito bits de dados, adiciona-se um bit de paridade para detecção de erro e, em seguida, um ou dois bits de parada para marcar o final. A mensagem só é aceita quando o bit de paridade e a polaridade do bit de parada estiverem em conformidade com os padrões de formato.

A Electronics Industry Alliance (EIA) desenvolveu os três grandes padrões de comunicação serial recomendados (*Recommended Standard* - RS): RS-232, RS-422 e RS-485 (SILVA; PEREIRA; BUCHMANN, 2013). Eles se tornaram conhecidos e amplamente utilizados em dispositivos e sensores devido a sua simplicidade e confiabilidade (TEIXEIRA, 2009). O presente trabalho se dedica a explicar e utilizar dos padrões RS-232 e RS-485.

2.4.1 Comunicação Serial RS-232

Embora antigo, o RS-232, também conhecido como EIA-232C, é muito utilizado por ser projetado para manipular comunicações distantes em até 15 m, com taxas transmissivas até 115200 bps e apenas entre dois dispositivos. A transmissão ocorre em palavras que dependendo da configuração podem ter de 5 a 9 bits configurando um único caractere.

As conexões podem ser de vários tipos, porém o padrão comumente usado em periféricos é o DB-9 (9 pinos), nas disposições DTE (macho) e DCE (fêmea), expressado na Fig 2.14. Cada pino tem uma função, especificada na Tab. 2.1, contudo para se estabelecer uma conexão RS-232 de duas vias existem 3 sinais essenciais: TXD, pino 3, que é a via transmissora de sinal; RXD, pino 2, que é a via receptora de sinal e; GND, pino 5, que é o sinal de aterramento que referencia e adequa os sinais entre dois dispositivos comunicando sob este protocolo (AXELSON, 2000).

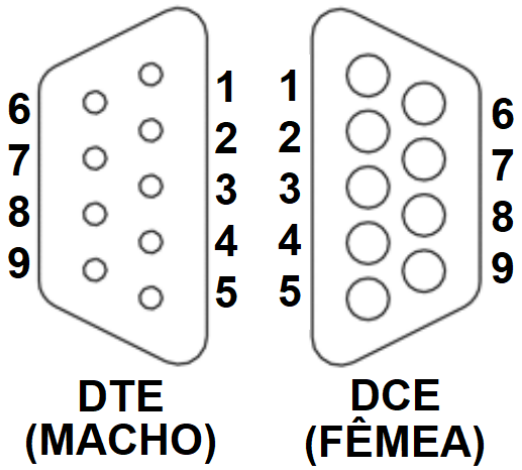


Figura 2.14 – Demarcação de pinos no conector DB-9 (modificado de AXELSON, 2000).

Tabela 2.1 – Especificação dos pinos RS-232 usados na interface com conector DB-9 (adaptado de AXELSON, 2000).

Pino	Sinal	Fonte do sinal	Tipo de sinal	Descrição
1	CD	DCE	Controle	Detector de portador
2	RxD	DCE	Dados	Dados recebidos
3	TxD	DTE	Dados	Dados enviados
4	DTR	DTE	Controle	Terminal de dados pronto
5	GND	-	-	Sinal de terra
6	DSR	DCE	Controle	Conjunto de dados pronto
7	RTS	DTE	Controle	Requisição de envio

8	CTS	DCE	Controle	Livre para enviar
9	RI	DCE	Controle	Indicador de toque

Algumas aplicações industriais possuem como requisito atingir distâncias maiores e como o protocolo RS-232 tem uma limitação de 15 metros, combinam-se então padrões de comunicação para se obter maiores distâncias possíveis entre dois dispositivos comunicantes. O protocolo RS-485, descrito a seguir, pode ser usado desta forma para se conseguir essa maior distância, com o auxílio de um conversor bidirecional desses dois protocolos (AXELSON, 2000).

2.4.2 Comunicação Serial RS-485

O padrão recomendado RS-485, também conhecido como EIA-485, foi projetado para oferecer maior flexibilidade por estabelecer comunicação numa rede com mais de 2 participantes (até 32 participantes são possíveis), como apresentado na Fig. 2.15, dar garantia de qualidade do transmitido numa distância de até 1200 metros e taxas de transmissão de até 12.000 kbps. Para atenuar efeitos de eco de sinal, recomenda-se o uso de uma resistência de 120 Ohms entre os terminais do que deve ser considerado o último dispositivo (ou o mais distante) da rede (PERRIN, 1999).

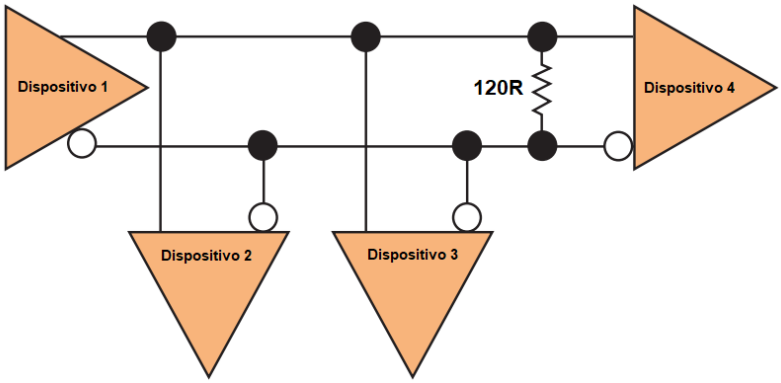


Figura 2.15 – Rede de dispositivos conectados via RS-485 com resistor terminal (transposto de PERRIN, 1999).

As conexões neste padrão, diferentemente do RS-232, podem ser de 4 fios (*full-duplex*) ou 2 fios (*half-duplex*) e a comunicação é feita por meio da diferença de tensão entre vias, ou seja, as vias não são dedicadas a transmissão ou a recepção. O protocolo garante também um balanço de tensão (tensões de mesmo valor, mas opostas em relação ao sinal) entre as vias, o que permite uma proteção contra ruídos, eliminados através da diferenciação de sinal, e possibilita o uso em maiores distâncias e maiores taxas de transmissão.

As vias numa rede RS-485 são geralmente denominadas de A e B e cada diferença de tensão é interpretada segundo o modo de transmissão utilizado. O presente trabalho utiliza-se da codificação ASCII como modo de transmissão, o qual é descrito na seção a seguir.

2.4.3 Codificação ASCII

O código ASCII (*American Standard Code for Information Interchange*) foi inicialmente, em 1963, definido como um código americano padronizado para troca de informações, por contemplar 128 caracteres (7 bits) usados na língua inglesa. Porém, com a adoção desse código por outras nacionalidades, foi necessário eleger mais caracteres, ampliando-se então para 8 bits o que disponibilizou mais 128 caracteres específicos de outras línguas, como por exemplo as vogais acentuadas, recorrentes na Língua Portuguesa (NUNES, 2016). A partir dessa generalização o código deixou de ser algo americano e se tornou universal.

As correspondências da codificação ASCII com os valores binários, decimal e hexadecimal estão inseridas no Anexo A deste trabalho. Essa correspondência possibilita grupos de bits que trafegam através da comunicação serial se converterem em comandos. Esses comandos são enviados e recebidos através de uma lógica de programação utilizada no CLP (Controlador Lógico Programável).

2.5 Controlador Lógico Programável - CLP

Os chamados Controladores Lógico Programáveis (CLPs) são dispositivos que oferecem facilidade de programação e reprogramação, confiabilidade para ambientes industriais, possibilidade de expansão em módulos adjacentes (mais entradas e saídas) e integração de dados de processos do mesmo (FRANCHI, 2008). A estrutura de um CLP é dividida então numa Unidade Central de Processamento (CPU) e interfaces de entrada e saída, como exibido na Fig. 2.16.

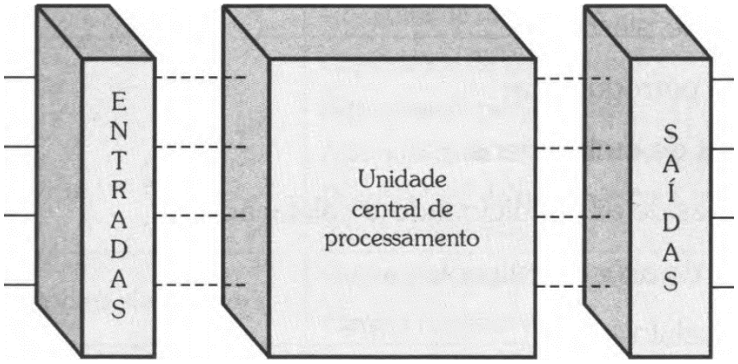


Figura 2.16 – Estrutura simplificada de um CLP (modificado de FRANCHI, 2008).

O CLP Rexroth IndraControl L25®, mostrado na Fig. 2.17, é utilizado como sistema central da unidade UPPAX e realiza o controle da unidade, combinando os benefícios de um dispositivo compacto com um sistema padrão de entradas e saídas. Este sistema fornece ainda interfaces de comunicação Ethernet e pode ser expandido em dois módulos via barramento PCI e *Fieldbus*.

Figura 2.17 – CLP Rexroth IndraControl L25® (extraído de <http://www.hidrosistemas.ind.br >, 2018).

Algumas linguagens de programação para CLPs são definidas pela norma IEC 61131-3 e podem ser implementadas através do uso do software IndraLogic®. No presente trabalho, após testes práticos, decidiu-se pela utilização de uma linguagem gráfica, o *Ladder* (LD – *Ladder Diagram*), e uma linguagem textual, o denominado Texto Estruturado (ST – *Structured Text*).

2.5.1 Linguagem de Programação Ladder

A linguagem *Ladder* (LD) é uma linguagem gráfica baseada em diagramas de contatos elétricos ou relés. Os contatos representam as entradas, que podem ser normalmente abertos ou fechados e possibilitam a energização ou não de atuadores (saídas), de acordo com a lógica implementada. A Figura 2.18 apresenta um exemplo simples com os componentes principais dessa linguagem.

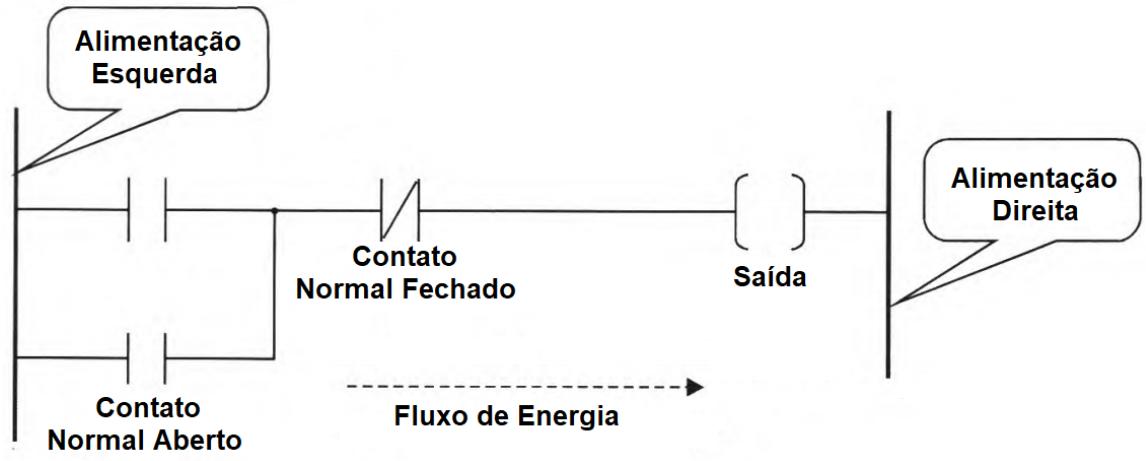


Figura 2.18 – Estrutura típica em linguagem *Ladder* (transcrito com alterações de FRANCHI, 2008).

A interface do software IndraLogic®, demonstrada na Fig. 2.19, possui outras funcionalidades que auxiliam na programação, como temporizadores e blocos de sub-rotinas (ou subprogramas).

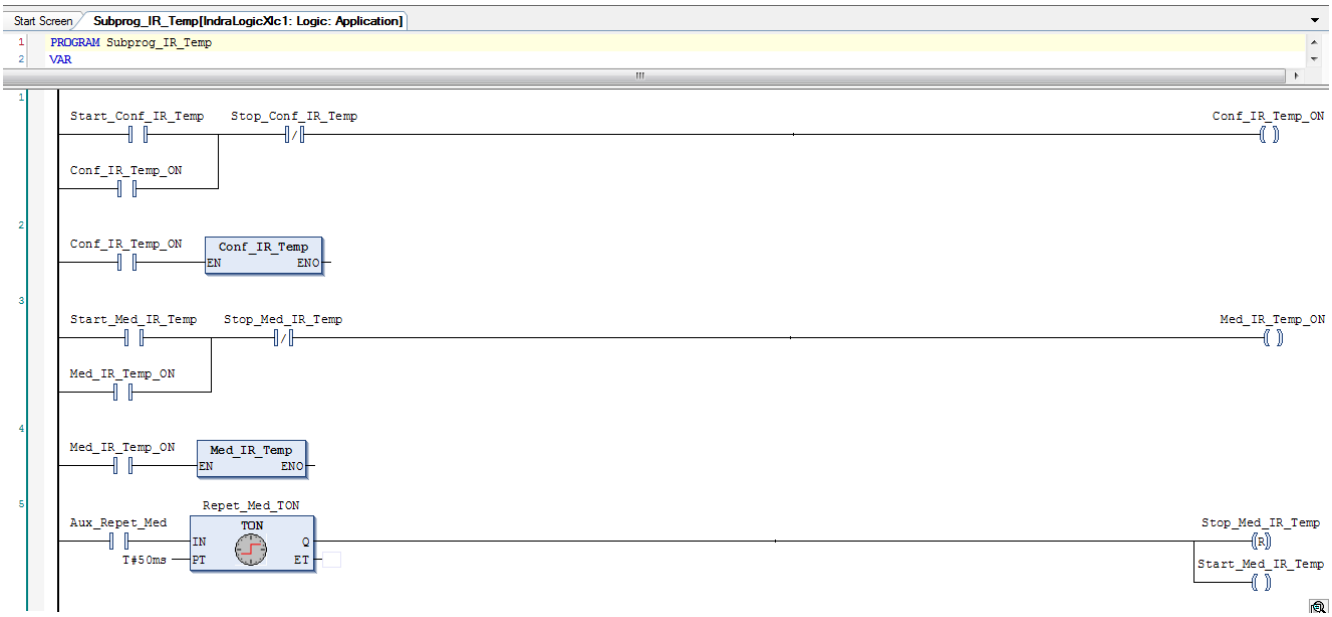


Figura 2.19 – Parte de um programa em linguagem Ladder no software IndraLogic®.

2.5.2 Linguagem de Programação Texto Estruturado

O Texto Estruturado (ST) é uma linguagem textual muito alinhada com as linguagens de programação atuais, que contêm instruções condicionais (IF, CASE), estruturas de iteração (FOR, WHILE, REPEAT), mas também facilita uma programação de comportamento sequencial (FRANCHI, 2008). Essas características é que a tornam a mais recomendada em execuções complexas. A Figura 2.20 apresenta uma parte de uma sub-rotina desta linguagem na interface do IndraLogic®, com instruções de temporização e escrita na porta física do módulo RS-485 do CLP.

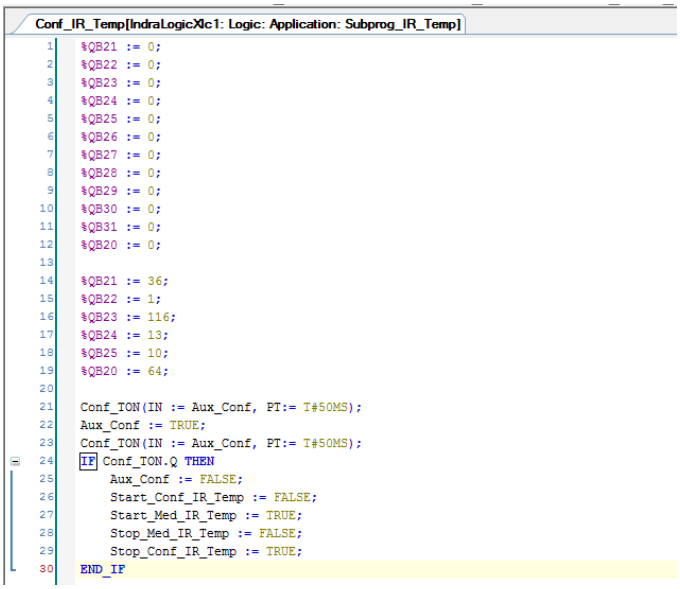


Figura 2.20 – Parte de uma sub-rotina em Texto Estruturado, na interface IndraLogic®.

CAPÍTULO III

DESENVOLVIMENTO PRÁTICO

Este capítulo se destina a esclarecer sobre procedimentos práticos adotados no projeto de implementação dos sensores e foi dividido em duas etapas: implementação da célula de carga e; implementação do sensor de temperatura infravermelho.

3.1 Implementação da Célula de Carga

A célula de carga é um sensor de força e sua implementação é feita inicialmente pela concretização da conexão dos componentes físicos (*hardware*) e em seguida desenvolvendo-se a aplicação (*software*).

3.1.1 Hardware para Medição de Força

O sensor de força HBM U2B® (Fig. 3.1) utiliza de outros dois equipamentos para amplificar, tratar e converter o sinal obtido no sensor para uma interface serial. Estes componentes são um amplificador (HBM AD103C®), mostrado na Fig. 3.2, e um condicionador de sinal (HBM AED9101D®), representado na Fig. 3.3 com os seus

componentes. Juntos esses dispositivos possibilitam uma conexão serial RS-485 com o módulo do R-IB IL RS 485/422 PRO-PAC® (Fig. 3.4) produzido pela empresa Bosch Rexroth®, que é responsável pela interface serial do CLP da unidade UPPAX.

Figura 3.1 – Célula de Carga U2B® 50kN produzida pela HBM.

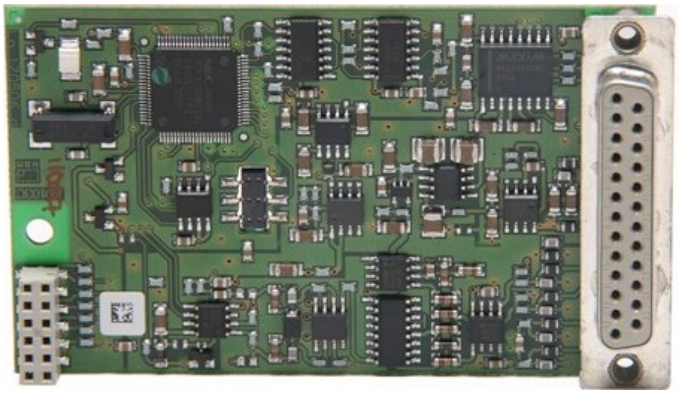


Figura 3.2 – Amplificador AD103C® produzido pela HBM.

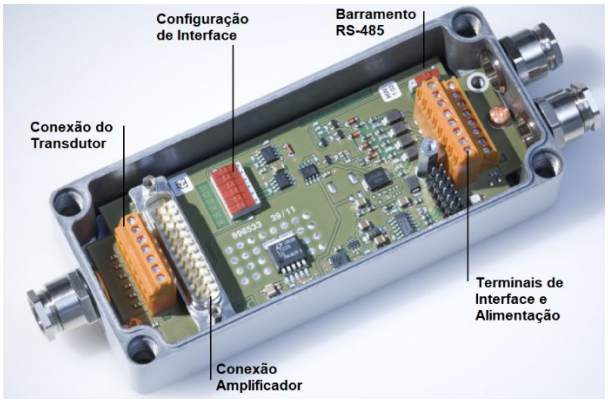


Figura 3.3 – Condicionador AED9101D® produzido pela HBM e seus componentes.

Figura 3.4 – Módulo Serial R-IB IL RS 485/422 PRO-PAC® produzido pela Bosch Rexroth.

Os terminais de interface e alimentação do dispositivo AED9101D®, expostos na Fig. 3.5, são conectados na interface do módulo serial e na alimentação (24V) do CLP. Para efetuar o contato entre esses equipamentos foi projetado um cabo com conexão de 15 vias (DB-15), sendo a via referente ao CLP um conector do tipo macho e a via referente ao condicionador de sinal um conector do tipo fêmea, representado na Fig. 3.6 com seus respectivos sinais na Tab. 3.1.

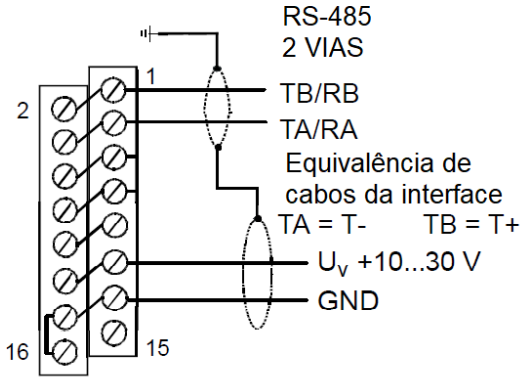


Figura 3.5 – Detalhamento do terminal de interface e alimentação do condicionador de sinal AED9101D®.

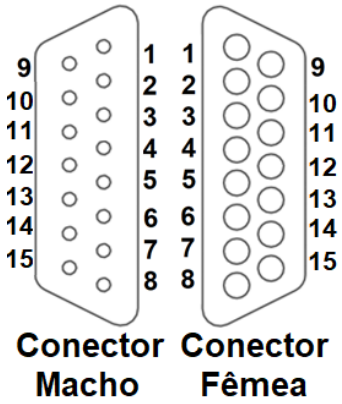


Figura 3.6 – Conector de 15 pinos e sua sequência de pinos.

Tabela 3.1 – Correspondência de sinais no conector DB-15 projetado.

PINO	SINAL
1	TB/RB/T+
2	-
3	-
4	TA/RA/T-
5	-
6	GND
7	0V
8	-
9	24V
10	24V
11	24V
12	-
13	-
14	0V
15	0V

Estabelecida a conexão entre as partes, é necessário então realizar uma aplicação que concretize a implementação da célula de carga à unidade UPPAX.

3.1.2 Software para Medição de Força

Durante o desenvolvimento de um *software* para a célula de carga deve se preocupar primeiramente com uma lista de comandos e ajustes que podem ser feitos ao condicionador de sinais AED9101D®. O Anexo B contém os principais comandos desse condicionador, utilizados na aplicação elaborada utilizando a codificação ASCII.

A aplicação tem o objetivo de estabelecer a comunicação com o condicionador de sinal, selecionar novos parâmetros para uma melhor comunicação e realizar a medição da força axial atuante na célula de carga. Além disso, o programa deve conter rotinas que evitem ou reparem possíveis erros do equipamento ou dos operadores. O programa, demonstrado no Apêndice A, foi desenvolvido na plataforma IndraLogic®, distribuído pelo fabricante Bosch Rexroth, e foram utilizadas as linguagens *Ladder* e Texto Estruturado para a implementação de rotinas, sub-rotinas e funções que o compõem.

O sistema deve operar de modo semelhante à topologia mestre-escravo, onde o CLP faz o papel de mestre, enviando comandos, e o condicionador de sinais responde a estes comandos, atuando como um escravo. A Figura 3.7 mostra o algoritmo de funcionamento deste *software* em forma de fluxograma.

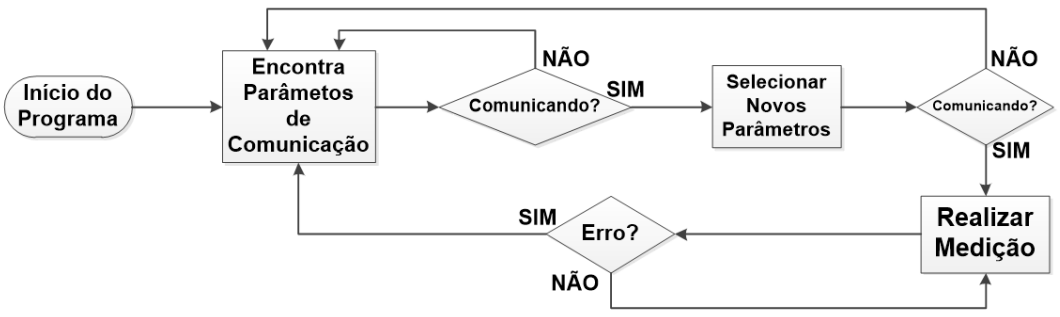


Figura 3.7 – Fluxograma lógico do software para a medição da força na célula de carga.

A ideia básica do programa é a existência de dois estados: *comunicando* e; *não-comunicando*. Sempre que o módulo condicionador de sinal está se comunicando com o CLP, a informação segue seu fluxo normal, até a medição, que é feita continuamente. Caso haja algum erro de comunicação (*não-comunicando*), o sistema retorna do início e tenta reencontrar os parâmetros para restabelecer comunicação e voltar a realizar as medições.

3.2 Implementação do Sensor de Temperatura Infravermelho

O sensor de temperatura infravermelho utilizado no projeto é o modelo RAYMID10LT®, fabricado pela empresa RAYTEK. A implementação desse sensor se realizou em duas etapas: a de montagem e conexão de componentes físicos (*hardware*) e; o desenvolvimento de uma programação adequada (*software*).

3.2.1 Hardware para Medição de Temperatura

O sensor infravermelho, demonstrado na Fig. 3.8, mede a quantidade de energia emitida por um objeto e converte essa energia em um sinal de temperatura e vem acompanhado de um circuito eletrônico, esquematizado na Fig. 3.9, que pode converter esse sinal de temperatura em sinal de Termopares (K e J), tensão (0-5V), de corrente (0-20 mA ou 4-20 mA) ou para interface serial (RS-232). A Figura 3.9 revela ainda as entradas da interface serial, indicadas por RxD, TxD e GND, e a entrada de alimentação, indicada por +12...24V e GND.

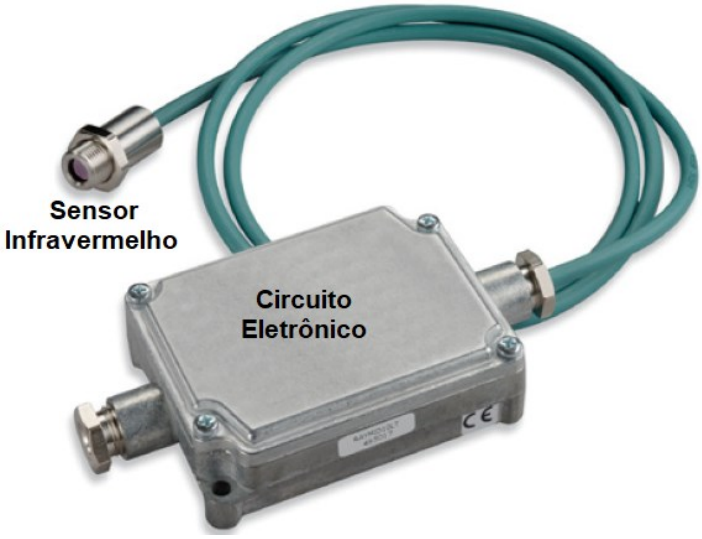


Figura 3.8 – Sensor de temperatura infravermelho RAYMID10LT® com circuito eletrônico conversor de sinal.

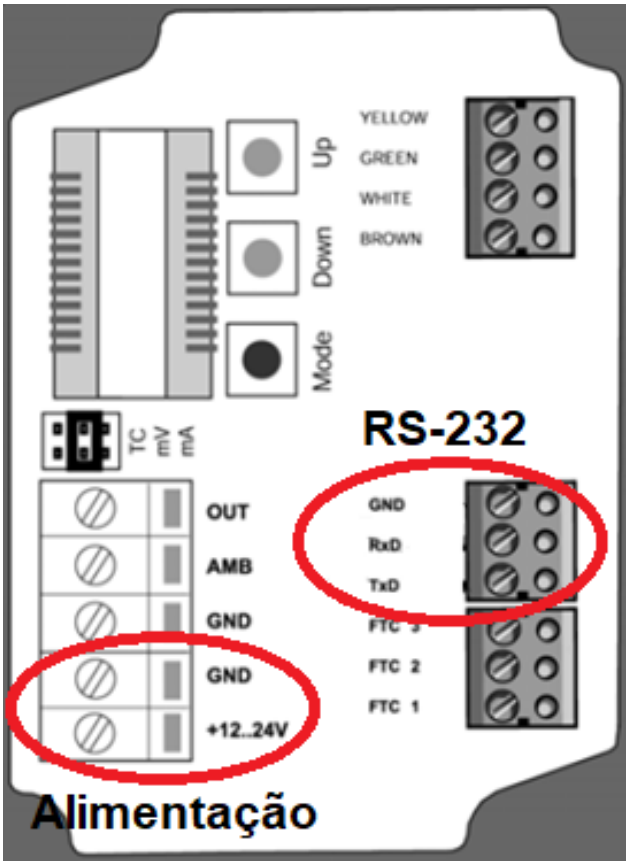


Figura 3.9 – Esquema do circuito eletrônico do sensor de temperatura infravermelho RAYMID10LT®.

Como esse dispositivo só opera pela interface RS-232, foi necessário utilizar um conversor bidirecional do padrão RS-232 para o RS-485, para que fosse possível usar o módulo serial do CLP e também para que fosse aproveitada a conexão projetada (DB-15) para a instalação da célula de carga, com a mesma configuração apontada na Fig. 3.6. Portanto, foi adquirido o conversor E232-485-2®, exibido na Fig. 3.10, do fabricante COMM5.

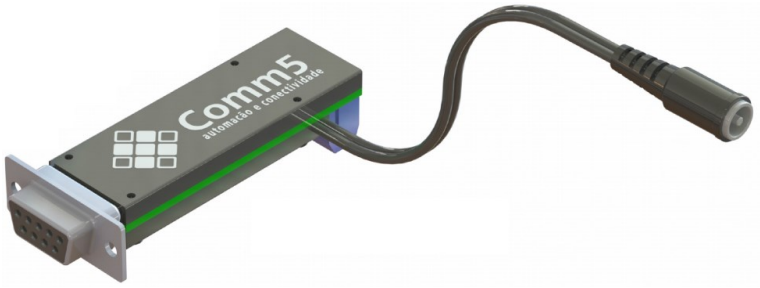


Figura 3.10 – Conversor bidirecional de RS-232 para RS-485 E232-485-2®, do fabricante COMM5.

Por meio do uso desse conversor foi possível então conectar o módulo serial do CLP com o sensor de temperatura. A partir disso, foi necessário criar uma aplicação para estabelecer essa comunicação.

3.2.2 Software para Medição de Temperatura

Diferentemente do condicionador de sinais da célula de carga, o circuito eletrônico do sensor infravermelho não permite alteração dos parâmetros de comunicação. A lógica de funcionamento deste circuito só permite receber uma requisição de medição para então responder com o sinal de temperatura na interface serial.

O programa se baseia em selecionar os parâmetros de configuração da comunicação (protocolo RS-485, *Baud Rate* e paridade) no módulo do CLP e requerer a temperatura atual medida no sensor. O Apêndice B deste trabalho traz a programação em *Ladder* e Texto Estruturado, feita na plataforma IndraLogic®.

CAPÍTULO IV

RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos da implementação da célula de carga na unidade UPPAX e a implementação do sensor de temperatura infravermelho.

4.1 Funcionalidade do *Software* para Medição de Força

O *software* desenvolvido para as medições da força realizou a comunicação entre o conjunto, formado pela célula de carga, o condicionador de sinal, e o CLP. Além disso, por meio do *software* foi possível monitorar a força aplicada na célula de carga. A Figura 4.1 apresenta uma leitura aleatória para verificar a funcionalidade do *software* desenvolvido.

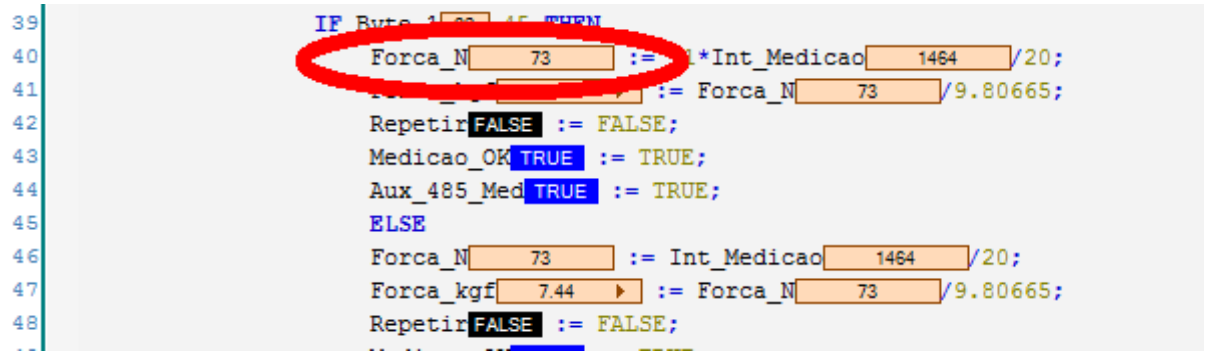


Figura 4.1 – Leitura da força realizada pelo *software* da célula de carga.

4.2 Funcionalidade do *Software* para Medição de Temperatura

A funcionalidade do *software* pode ser verificada na Fig. 4.2, que mostra a temperatura obtida no programa desenvolvido. Pode se observar que a leitura obtida é bem próxima da leitura no indicador do circuito eletrônico do sensor infravermelho, apresentado na Fig. 4.3.

Figura 4.2 – Verificação da temperatura monitorada pelo software desenvolvido.

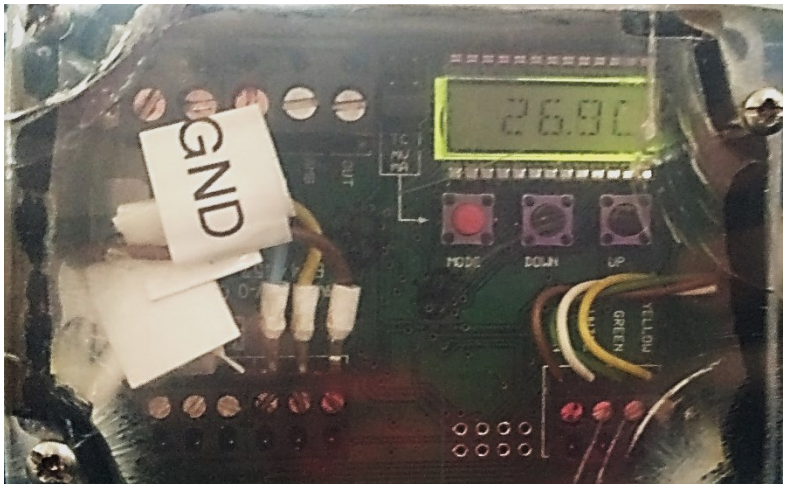


Figura 4.3 – Temperatura indicada no circuito eletrônico do sensor infravermelho.

CAPÍTULO V

CONCLUSÕES

Os programas desenvolvidos foram capazes de realizar a comunicação através do padrão industrial de comunicação RS-485, utilizando para tanto o módulo R-IB IL RS 485/422 PRO-PAC® do CLP Rexroth IndraControl L25®. Este padrão mostrou-se mais vantajoso que o RS-232 devido as maiores distâncias que pode assumir entre os dispositivos numa rede industrial, composta de um elemento central de controle (CLP) e um sensor enviando informações sobre o que é medido.

A automatização das medições acompanhada da utilização de uma programação, através da plataforma IndraLogic®, capaz de prever possíveis erros dos operadores foi importante para a construção da plataforma de integração dos sensores da unidade UPPAX. Isso agrega confiabilidade e segurança ao programa, requisitos fundamentais à instalação de novos equipamentos.

Os *softwares* desenvolvidos realizam também o monitoramento da força e da temperatura do material de ensaio. Os equipamentos (*hardwares*) que condicionam o sinal da célula de carga HBM U2B® e do sensor de temperatura infravermelho RAYMID10LT® foram devidamente instalados e implementados à lógica de controle do CLP.

CAPÍTULO VI

TRABALHOS FUTUROS

A avaliação do erro de medição da força é proposta como trabalho futuro, onde as medidas da célula de carga são tomadas como referência e, então, as medidas dos sensores de pressão são verificadas.

A verificação do efeito da velocidade de deslocamento do cilindro hidráulico da UPPAX na perda de carga é outra atividade a ser desenvolvida posteriormente. Com isso, propõe-se uma análise das consequências dessa velocidade na medição de força pelos sensores de pressão. Para isso, verifica-se a força medida pela célula de carga e pelos sensores de pressão com o cilindro se deslocando em diferentes velocidades.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDOLFATO, R. P.; CAMACHO, J. S.; BRITO, G. A. Extensometria Básica. 2004. Núcleo de Ensino e Pesquisa da Alvenaria Estrutural, Universidade Estadual Paulista "Júlio de Mesquita Filho", Ilha Solteira.

AXELSON, J. Serial Port Complete. Madison: Lakeside Research. 2000. 321p.

AWS - AMERICAN WELDING SOCIETY. Welding Handbook. 8. ed. Miami, 1991. v. 2, (0-87171-354-3).

CAIXETA, L. A. Otimização de Parâmetros de Processamento de Pinos por Atrito em Unidade com Capacidade de 245 kN. 2011. 122 p. Dissertação de Mestrado, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG.

DALLY, J.; RILEY, W.; McConnell, K. Instrumentation for Engineering Measurements. 2nd ed. Canadá: Jonh Wiley & Sons, Inc. 1993.

FORMOSO, C. M., Projeto e Desenvolvimento de um Sistema de Instrumentação e Controle para uma Unidade de Processamento de Pinos por Atrito Portátil. 2012. 88p. Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia-MG.

FRANCHI, C. M.; de Camargo, V. L. Controladores Lógico Programáveis - Sistemas Discretos. Editora Érica Ltda, 1. ed. São Paulo, 2008.

FREITAS, D. S. de. Controle de Força e Rotação de uma Unidade de Reparo por Atrito usando Controlador PID e Inteligência Artificial. 2014. 191 p. Dissertação de Mestrado,

Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG.

HWANG, H. F., Desenvolvimento, Projeto, Construção e Teste de um Cilindro de Reparo por Atrito Portátil. 2010. Dissertação de Mestrado, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG.

LADEIRA, G. M. V. Instrumentação de um tribômetro para ensaios de deslizamento em dutos flexíveis. 2011. 102p. Dissertação de Mestrado, Universidade Federal de Uberlândia, Uberlândia, MG.

LOTSBERG, I.; LANDET, E. Fatigue Capacity of Side Longitudinals in Floating Structures. Marine Structures. v. 18, n. 1, p. 25-42. jan. 2005.

LUCZYNSKI, E. Os Condicionantes para o Abandono das Plataformas Offshore após o Encerramento da Produção. 2002. 220 p. Tese de Doutorado, Programa Interunidades de Pós-Graduação em Energia, Universidade de São Paulo, São Paulo, SP.

MEYER, A. Friction Hydro Pillar Processing. 2004. 123 p. Dr.-Ing. Thesis an der Technischen Universität Braunschweig, Hamburg.

NUNES, J. A codificação binária da informação: códigos alfanuméricos. «Correio dos Açores: regional/opinião». 2016. p. 21.

PAVANI, S. A. Instrumentação Básica / Sérgio Pavani. – 3. ed. – Santa Maria: Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, Curso Técnico em Automação Industrial, 2011. 98 p.: Il. ; 21 cm.

PERRIN, B. The Art and Science of RS-485. Circuit Cellar Magazine, Jul. 1999.

PIRES, R. R. Efeitos da Geometria, da Força Axial e da Rotação no Reparo por Atrito. 2007. 136 p. Dissertação de Mestrado, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG.

SALAMA, M. M.; LOTSBERG, I., OMAE-FPSO'04-0054 Fatigue Performance of Repaired FPSO Details using Stitch Friction Welding Process. OMAE Speciality Symposium on FPSO Integrity, Houston, USA. Proceedings of OMAE-FPSO, p. 1-9, 2004.

SILVA, B. S.; PEREIRA, M. C.; BUCHMANN, R. M. Padrões de Comunicação Serial Clássicos: RS-232, RS-422 e RS-485. 2013. Departamento de Engenharia Eletrônica e de Computação, Escola Politécnica, Universidade Federal do Rio de Janeiro.

SILVA, P. S. C. P. Comportamento Mecânico e Fraturas de Componentes e Estruturas Metálicas. UFPR, p. 171-173, 1999.

SOUZA, R. J. Desenvolvimento, Projeto e Construção de um Equipamento de Reparo de Trincas por Atrito. 2006. 42 p. Dissertação de Mestrado, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG.

TEIXEIRA, L. L. Medidor de Energia Eletrônico. 2009. 68 p. Projeto de Diplomação, Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre.

YOUNG, H. D.; FREEDMAN, R. A. Física III: Eletromagnetismo. 14ª ed. São Paulo: Pearson, 2015.

ANEXO A

Tabela codificação ASCII

Tabela Codificação ASCII

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
0	0	0	Null – NUL	0%
1	1	1	Start of Heading – SOH	1%
2	2	10	Start of Text – STX	2%
3	3	11	End of Text – ETX	3%
4	4	100	End of Transmission – EOT	4%
5	5	101	Enquiry – ENQ	5%
6	6	110	Acknowledge – ACK	6%
7	7	111	Bell, rings terminal bell – BEL	7%
8	8	1000	BackSpace – BS	8%
9	9	1001	Horizontal Tab – HT	9%
10	0A	1010	Line Feed – LF	%0A
11	0B	1011	Vertical Tab – VT	%0B
12	0C	1100	Form Feed – FF	%0C
13	0D	1101	Enter – CR	%0D
14	0E	1110	Shift-Out – SO	%0E
15	0F	1111	Shift-In – SI	%0F
16	10	10000	Data Link Escape – DLE	10%
17	11	10001	Device Control 1 – D1	11%
18	12	10010	Device Control 2 – D2	12%
19	13	10011	Device Control 3 – D3	13%
20	14	10100	Device Control 4 – D4	14%
21	15	10101	Negative Acknowledge – NAK	15%
22	16	10110	Synchronous idle – SYN	16%
23	17	10111	End Transmission Block – ETB	17%
24	18	11000	Cancel line – CAN	18%
25	19	11001	End of Medium – EM	19%
26	1A	11010	Substitute – SUB	%1A
27	1B	11011	Escape – ESC	%1B
28	1C	11100	File Separator – FS	%1C
29	1D	11101	Group Separator – GS	%1D
30	1E	11110	Record Separator – RS	%1E
31	1F	11111	Unit Separator – US	%1F
32	20	100000	Space – SPC	+
33	21	100001	!	21%
34	22	100010	"	22%
35	23	100011	#	23%
36	24	100100	$	24%
37	25	100101	%	25%
38	26	100110	&	26%
39	27	100111	'	27%
40	28	101000	(	28%

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
41	29	101001	)	29%
42	2A	101010	*	%2A
43	2B	101011	+	%2B
44	2C	101100	,	%2C
45	2D	101101	-	-
46	2E	101110	.	.
47	2F	101111	/	%2F
48	30	110000	0	0
49	31	110001	1	1
50	32	110010	2	2
51	33	110011	3	3
52	34	110100	4	4
53	35	110101	5	5
54	36	110110	6	6
55	37	110111	7	7
56	38	111000	8	8
57	39	111001	9	9
58	3A	111010	:	%3A
59	3B	111011	;	%3B
60	3C	111100	<	%3C
61	3D	111101	=	%3D
62	3E	111110	>	%3E
63	3F	111111	?	%3F
64	40	1000000	@	40%
65	41	1000001	A	A
66	42	1000010	B	B
67	43	1000011	C	C
68	44	1000100	D	D
69	45	1000101	E	E
70	46	1000110	F	F
71	47	1000111	G	G
72	48	1001000	H	H
73	49	1001001	I	I
74	4A	1001010	J	J
75	4B	1001011	K	K
76	4C	1001100	L	L
77	4D	1001101	M	M
78	4E	1001110	N	N
79	4F	1001111	O	O
80	50	1010000	P	P
81	51	1010001	Q	Q
82	52	1010010	R	R
83	53	1010011	S	S
84	54	1010100	T	T
85	55	1010101	U	U

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
86	56	1010110	V	V
87	57	1010111	W	W
88	58	1011000	X	X
89	59	1011001	Y	Y
90	5A	1011010	Z	Z
91	5B	1011011	[	%5B
92	5C	1011100	\	%5C
93	5D	1011101	]	%5D
94	5E	1011110	^	%5E
95	5F	1011111	_	_
96	60	1100000	`	60%
97	61	1100001	a	a
98	62	1100010	b	b
99	63	1100011	c	c
100	64	1100100	d	d
101	65	1100101	e	e
102	66	1100110	f	f
103	67	1100111	g	g
104	68	1101000	h	h
105	69	1101001	i	i
106	6A	1101010	j	j
107	6B	1101011	k	k
108	6C	1101100	l	l
109	6D	1101101	m	m
110	6E	1101110	n	n
111	6F	1101111	o	o
112	70	1110000	p	p
113	71	1110001	q	q
114	72	1110010	r	r
115	73	1110011	s	s
116	74	1110100	t	t
117	75	1110101	u	u
118	76	1110110	v	v
119	77	1110111	w	w
120	78	1111000	x	x
121	79	1111001	y	y
122	7A	1111010	z	z
123	7B	1111011	{	%7B
124	7C	1111100	|	%7C
125	7D	1111101	}	%7D
126	7E	1111110	~	%7E
127	7F	1111111	•	%7F
128	80	10000000	€	80%
129	81	10000001	•	81%
130	82	10000010	‚	82%

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
131	83	10000011	ƒ	83%
132	84	10000100	„	84%
133	85	10000101	…	85%
134	86	10000110	†	86%
135	87	10000111	‡	87%
136	88	10001000	ˆ	88%
137	89	10001001	‰	89%
138	8A	10001010	Š	%8A
139	8B	10001011	‹	%8B
140	8C	10001100	Œ	%8C
141	8D	10001101	•	%8D
142	8E	10001110	Ž	%8E
143	8F	10001111	•	%8F
144	90	10010000	•	90%
145	91	10010001	‘	91%
146	92	10010010	’	92%
147	93	10010011	“	93%
148	94	10010100	”	94%
149	95	10010101	•	95%
150	96	10010110	–	96%
151	97	10010111	—	97%
152	98	10011000	˜	98%
153	99	10011001	™	99%
154	9A	10011010	š	%9A
155	9B	10011011	›	%9B
156	9C	10011100	œ	%9C
157	9D	10011101	•	%9D
158	9E	10011110	ž	%9E
159	9F	10011111	Ÿ	%9F
160	A0	10100000		%A0
161	A1	10100001	¡	%A1
162	A2	10100010	¢	%A2
163	A3	10100011	£	%A3
164	A4	10100100	¤	%A4
165	A5	10100101	¥	%A5
166	A6	10100110	¦	%A6
167	A7	10100111	§	%A7
168	A8	10101000	¨	%A8
169	A9	10101001	©	%A9
170	AA	10101010	ª	%AA
171	AB	10101011	«	%AB
172	AC	10101100	¬	%AC
173	AD	10101101		%AD
174	AE	10101110	®	%AE
175	AF	10101111	¯	%AF

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
176	B0	10110000	°	%B0
177	B1	10110001	±	%B1
178	B2	10110010	²	%B2
179	B3	10110011	³	%B3
180	B4	10110100	´	%B4
181	B5	10110101	µ	%B5
182	B6	10110110	¶	%B6
183	B7	10110111	·	%B7
184	B8	10111000	¸	%B8
185	B9	10111001	¹	%B9
186	BA	10111010	º	%BA
187	BB	10111011	»	%BB
188	BC	10111100	¼	%BC
189	BD	10111101	½	%BD
190	BE	10111110	¾	%BE
191	BF	10111111	¿	%BF
192	C0	11000000	À	%C0
193	C1	11000001	Á	%C1
194	C2	11000010	Â	%C2
195	C3	11000011	Ã	%C3
196	C4	11000100	Ä	%C4
197	C5	11000101	Å	%C5
198	C6	11000110	Æ	%C6
199	C7	11000111	Ç	%C7
200	C8	11001000	È	%C8
201	C9	11001001	É	%C9
202	CA	11001010	Ê	%CA
203	CB	11001011	Ë	%CB
204	CC	11001100	Ì	%CC
205	CD	11001101	Í	%CD
206	CE	11001110	Î	%CE
207	CF	11001111	Ï	%CF
208	D0	11010000	Ð	%D0
209	D1	11010001	Ñ	%D1
210	D2	11010010	Ò	%D2
211	D3	11010011	Ó	%D3
212	D4	11010100	Ô	%D4
213	D5	11010101	Õ	%D5
214	D6	11010110	Ö	%D6
215	D7	11010111	×	%D7
216	D8	11011000	Ø	%D8
217	D9	11011001	Ù	%D9
218	DA	11011010	Ú	%DA
219	DB	11011011	Û	%DB
220	DC	11011100	Ü	%DC

Decimal:	Hexadecimal:	Binário:	Caractere:	URL Encode:
221	DD	11011101	Ý	%DD
222	DE	11011110	Þ	%DE
223	DF	11011111	ß	%DF
224	E0	11100000	à	%E0
225	E1	11100001	á	%E1
226	E2	11100010	â	%E2
227	E3	11100011	ã	%E3
228	E4	11100100	ä	%E4
229	E5	11100101	å	%E5
230	E6	11100110	æ	%E6
231	E7	11100111	ç	%E7
232	E8	11101000	è	%E8
233	E9	11101001	é	%E9
234	EA	11101010	ê	%EA
235	EB	11101011	ë	%EB
236	EC	11101100	ì	%EC
237	ED	11101101	í	%ED
238	EE	11101110	î	%EE
239	EF	11101111	ï	%EF
240	F0	11110000	ð	%F0
241	F1	11110001	ñ	%F1
242	F2	11110010	ò	%F2
243	F3	11110011	ó	%F3
244	F4	11110100	ô	%F4
245	F5	11110101	õ	%F5
246	F6	11110110	ö	%F6
247	F7	11110111	÷	%F7
248	F8	11111000	ø	%F8
249	F9	11111001	ù	%F9
250	FA	11111010	ú	%FA
251	FB	11111011	û	%FB
252	FC	11111100	ü	%FC
253	FD	11111101	ý	%FD
254	FE	11111110	þ	%FE
255	FF	11111111	ÿ	%FF

ANEXO B

Principais Comandos do Condicionador de Sinais HBM AED9101D®

BDR

Baud Rate
(Baud rate)

Baud rates:	1200, 2400, 4800, 9600, 19200, 38400 Baud
Facrory setting:	9600 Baud and Even Parity Bit
Response time:	<10ms
Parameters:	2
Password protection:	no
Parameter protection:	with command **TDD1**

Query: **BDR <Baud rate>,<Parity>**

Entry of the required baud rate as decimal number.

The following baud rates are possible:

1200, 2400, 4800, 9600, 19200, 38400 bauds

Entry of the parity required:

0= without parity bit
1=with parity bit even

Important note

The answer is given in the new setting (baud rate, parity). Communication is no longer possible initially after a changed baud rate. The computer also has to be changed over to the new selected baud rate setting. To maintain the new baud rate, it must be stored in the EEPROM with the command **TDD1**. This procedure serves also as safeguard that no baud rates can be set in the AED which the remote station does not support. If the new entered baud rate is not stored, the AED reports after a reset or power On again in the previously valid baud rate.

Query: **BDR?;**

Effect: Output of the set baud rate, Identification for parity

Example: ***BDR?; 9600,1 CRLF*** *corresponds to 9600 bauds, Even Parity*

COF

Configurate Output Format
(Output format for the measured value)

Range: 0...255
Factory setting: 9
Response time: <10ms
Parameters: 1
Password protection: no
Parameter protection: with command **TDD 1**

Query: **COF(0...255);**

Entry of the output format for measured value command **MSV?**

The possible formats and the decimal number to be entered for them are listed in the following table. The measured value output refers to the set nominal value of the AED (see command **NOV**).

Output at max. capacity	NOV> 0	NOV= 0
2 Byte binary	NOV value	20000
4 Byte binary	NOV value	5120000
ASCII	NOV value	1000000

For the 2-bytes binary output, the **NOV** value must be $\leq$ 30000, otherwise the measured value is output with overflow or underflow ($7fff_H$ or 8000_H). With NOV30000, the overload range is only approx. 2700 digits.

Query: **COF?;**

Effect: Output of the selected output format as three-digit decimal number from 0...255

SZA

Sensor Zero Adjust
(Factory characteristic zeropoint)

Range:	0...1,599999e6
Factory setting:	Calibration to 0mV/V
Response time:	<15ms...4.2s
Parameters:	1/0
Password protection:	yes
Parameter protection:	after entry of **SFA**

For an entry signal of 0mV/V, the internal measured value is assigned to the output value 0 Digit.

Accepting an applied signal with SZA (response time<4.2s):

1. Connect the transducer electronics to a calibration standard.
2. Set the calibration standard to 0mV/V staggering.
3. Accept the applied signal by means of the command **SZA**.

   The applied signal is measured and filed in the memory, but accounted for only after measuring or entering the **SFA** value.

Manual input of the zero point via SZA (response time<15ms):

1. Use the command **SZA<zero value>** to enter the zero point.

The entered value is stored, but accounted for only after measuring or entering the parameter for **SFA**.

Query: **SZA?**; (Response time <15ms)

Effect: The value used in the AED for calculating the factory characteristic is output with ±7 digits (e.g. - 0000345crlf).

SFA

Sensor Fullscale Adjust
(Factory characteristic end value)

Range:	0...1,599999e6
Factory setting:	Calibration to 2mV/V
Response time:	<15ms...4.2s
Parameters:	1/0
Password protection:	yes
Data backup:	on entry

Accepting an applied signal with SFA (response time<4.2s):

1. Connect the transducer electronics to a calibration standard.
2. Set the calibration standard to 2mV/V staggering.
3. Accept the applied signal by means of the command **SFA**.

   The applied signal is measured and filed in the memory, and offset against the previously measured or entered **SZA** value.

Manual input of the zero point via SZA (reaction time<15ms):

1. Use the command SFA<nominal value> to enter the measured value for 2mV/V.

   The entered value is stored, and offset against the previously measured or entered **SZA** value.

Query: **SFA?**; (Response time<10ms)

Effect: The value used in the AED for calculating the factory characteristic is output with ±7 digits (e.g. - 0950246crlf).

NOV

Nominal Value
(Resolution of the user characteristic)

Range:	0...1,599999e6
Factory setting:	0 (=switched off)
Response time:	<10ms
Parameters:	1
Password protection:	yes
Parameter protection:	with command **TDD1**

Input: **NOV<value>;**

Query: **NOV?**; (Response time <10ms)

Effect: The value stored in the AED is output in 7 digits complete with sign (e.g. 0000345crlf).

The **NOV** value is used for scaling the output value during measured value output. At **NOV**=0 this output scaling is deactivated. The ASCII measured value output is scaled to 1000000 at the factory. If a measured value output of 2000 digit at nominal load is required, then use this command to set the nominal value **NOV2000**. The input parameters or tara values are not changed by this scaling.

Meas. value output format at nom. load	NOV=0	NOV>0
2 Byte binary	20000	NOV value
4 Byte binary	5120000	NOV value
ASCII	1000000	NOV value

With the 2 byte binary type of output, the **NOV** value must be $\leq$ 30000. Otherwise the measured value will be output complete with overflow or underflow ($7fff_H$ or 8000_H ; H: Hexadecimal). With **NOV30000**, the overload range is only approx. 2700 digits.

MSV

Measured Signal Value
(Output measured values)

Range:	Integer	±32767
	Long Integer	±8388607
	ASCII	±1000000
Factory setting:	ASCII	
Response time:	for **FMD0**: < 2^{ICR} + 1,67ms + 1,67ms	
	for **FMD1**: < 2^{ICR} + **ASF**(1...9) + 1,67ms + 1,67ms, with ICR = Measuring rate	
Parameters:	1	
Password protection:	no	
Parameter protection:	no data to be protected	

Query: **MSV?(0);** (not in 2-wire mode)

Effect: Continuous output of measured values until output is stopped by means of the **STP** command.

Query: **MSV?(1...65535);**

Effect: Outputs the stated number of measured values.

The measured value will be output in ASCII or binary format (see command **COF**).

RES

Restart
(Device start)

Range: —
Factory setting: —
Response time: <3s
Parameters: —
Password protection: no
Parameter protection: no data to be protected

The command **RES** produces a warm start. This command generates no answer. All parameters are set as they were stored with the last **TDD** command, i.e. EEPROM values are taken over into the RAM.

ESR

Event Status Register
(Output of error messages)

Query: **ESR?;**

Effect: This function outputs the error messages, defined according to the IEC standard, as a 3-digit decimal. The occurring errors are linked by "Or".

Error message	Error
000	No error
004	Not In use
008	Device Dependent Error (hardware error, e.g. EEPROM error)
016	Execution Error (parameter entry error)
032	Comand Error (command does not exist)

Example:

024 → Hardware- and parameter error

After **RES**, power On or reading the error status, the contents of the register is deleted.

APÊNDICE A

Programa da Célula de Carga

Rotina Principal (LD)

Primeira Sub-rotina de Teste de Comunicação (ST)

```
%QB21 := 0;
%QB22 := 0;
%QB23 := 0;
%QB24 := 0;
%QB25 := 0;
%QB26 := 0;
%QB27 := 0;
%QB28 := 0;
%QB29 := 0;
%QB30 := 0;
%QB31 := 0;
%QB20 := 0;

%QB21 := 36;
%QB22 := 2;
%QB23 := 114;
%QB24 := 13;
%QB25 := 10;
%QB20 := 64;

%QB21 := 69;
%QB22 := 83;
%QB23 := 82;
%QB24 := 63;
%QB25 := 10;
%QB20 := 21;

Delay0(IN := Aux_Delay0, PT:= T#300MS);
Delay1(IN := Aux_Delay1, PT:= T#300MS);
Aux_Delay0 := TRUE;
Delay0(IN := Aux_Delay0, PT:= T#300MS);
IF Delay0.Q THEN
    %QB20 := 48;
    Aux_Delay1 := TRUE;
    Delay1(IN := Aux_Delay1, PT:= T#300MS);
    Var_Iniciar := FALSE;
    IF Delay1.Q=TRUE THEN
        Aux_Delay0 := FALSE;
        Aux_Delay1 := FALSE;
        Read_Byte_0 := %IB40;
        Read_Byte_1 := %IB41;
        Read_Byte_2 := %IB42;
        Read_Byte_3 := %IB43;
        Read_Byte_4 := %IB44;
        Read_Byte_5 := %IB45;
        Read_Byte_6 := %IB46;
        Read_Byte_7 := %IB47;
        Read_Byte_8 := %IB48;
        Read_Byte_9 := %IB49;
        Read_Byte_10 := %IB50;
        Read_Byte_11 := %IB51;
        IF Read_Byte_0>=48 AND Read_Byte_0<=59 THEN
            Com2 := FALSE;
            Comunicacao_OK := TRUE;
        END_IF
    END_IF
END_IF
```

Segunda Sub-rotina de Teste de Comunicação (ST)

```
Repetir := FALSE;
Medicao_OK := FALSE;
%QB21 := 69;
%QB22 := 83;
%QB23 := 82;
%QB24 := 63;
%QB25 := 10;
%QB20 := 21;
Delay0(IN := Aux_Delay0, PT:= T#300MS);
Delay1(IN := Aux_Delay1, PT:= T#300MS);
Aux_Delay0 := TRUE;
Delay0(IN := Aux_Delay0, PT:= T#300MS);
IF Delay0.Q=TRUE THEN
    %QB20 := 48;
    Aux_Delay1 := TRUE;
    Delay1(IN := Aux_Delay1, PT:= T#300MS);
    Comunicacao_OK := FALSE;
    IF Delay1.Q=TRUE THEN
        Aux_Delay0 := FALSE;
        Aux_Delay1 := FALSE;
        Read_Byte_0 := %IB40;
        Read_Byte_1 := %IB41;
        Read_Byte_2 := %IB42;
        Read_Byte_3 := %IB43;
        Read_Byte_4 := %IB44;
        Read_Byte_5 := %IB45;
        Read_Byte_6 := %IB46;
        Read_Byte_7 := %IB47;
        Read_Byte_8 := %IB48;
        Read_Byte_9 := %IB49;
        Read_Byte_10 := %IB50;
        Read_Byte_11 := %IB51;
        IF Read_Byte_0 = 53 THEN
            IF Read_Byte_1 = 48 AND Read_Byte_2 = 48 AND Read_Byte_3 = 48 THEN
                Com_OK_2 := TRUE;
                Com2 := TRUE;
                ELSE
                Var_iniciar := TRUE;
                Com_OK_2 := FALSE;
                Com2 := TRUE;
            END_IF
            ELSE
            Var_iniciar := TRUE;
            Com_OK_2 := FALSE;
            Com2 := TRUE;
        END_IF
    END_IF
END_IF
```

Primeira Sub-rotina de Configuração da Comunicação (ST)

```
%QB21 := 83;
%QB22 := 80;
%QB23 := 87;
%QB24 := 34;
%QB25 := 65;
%QB26 := 69;
%QB27 := 68;
%QB28 := 34;
%QB29 := 10;
%QB30 := 0;
%QB31 := 0;
%QB20 := 25;
Conf_Delay0(IN := Aux_Conf0, PT:= T#50MS);
Aux_Conf0 := TRUE;
Conf_Delay0(IN := Aux_Conf0, PT:= T#50MS);
IF Conf_Delay0.Q THEN
    %QB21 := 84;
    %QB22 := 68;
    %QB23 := 68;
    %QB24 := 48;
    %QB25 := 10;
    %QB26 := 0;
    %QB27 := 0;
    %QB28 := 0;
    %QB29 := 0;
    %QB30 := 0;
    %QB31 := 0;
    %QB20 := 21;
    Conf_Delay1(IN := Aux_Conf1, PT:= T#2500MS);
    Aux_Conf1 := TRUE;
    Conf_Delay1(IN := Aux_Conf1, PT:= T#2500MS);
    IF Conf_Delay1.Q THEN
        Aux_Conf0 := FALSE;
        Aux_Conf1 := FALSE;
        Com_OK_2 := FALSE;
        Conf_OK := TRUE;
        Com2 := FALSE;
        Aux_485_Conf := FALSE;
    END_IF
END_IF
```

Segunda Sub-rotina de Configuração da Comunicação (ST)

```
%QB21 := 78;
%QB22 := 79;
%QB23 := 86;
%QB24 := 49;
%QB25 := 48;
%QB26 := 48;
%QB27 := 48;
%QB28 := 48;
%QB29 := 48;
%QB30 := 48;
%QB31 := 10;
%QB20 := 27;

Conf_Delay0(IN := Aux_Conf0, PT:= T#50MS);
Aux_Conf0 := TRUE;
Conf_Delay0(IN := Aux_Conf0, PT:= T#50MS);
IF Conf_Delay0.Q THEN

    %QB21 := 83;
    %QB22 := 90;
    %QB23 := 65;
    %QB24 := 48;
    %QB25 := 10;
    %QB26 := 0;
    %QB27 := 0;
    %QB28 := 0;
    %QB29 := 0;
    %QB30 := 0;
    %QB31 := 0;
    %QB20 := 21;
    Conf_Delay1(IN := Aux_Conf1, PT:= T#50MS);
    Aux_Conf1 := TRUE;
    Conf_Delay1(IN := Aux_Conf1, PT:= T#50MS);
    IF Conf_Delay1.Q THEN

        %QB21 := 83;
        %QB22 := 70;
        %QB23 := 65;
        %QB24 := 57;
        %QB25 := 57;
        %QB26 := 55;
        %QB27 := 57;
        %QB28 := 53;
        %QB29 := 52;
        %QB30 := 10;
        %QB31 := 0;
        %QB20 := 26;
        Conf_Delay2(IN := Aux_Conf2, PT:= T#50MS);
        Aux_Conf2 := TRUE;
        Conf_Delay2(IN := Aux_Conf2, PT:= T#50MS);
        IF Conf_Delay2.Q THEN
            %QB21 := 67;
            %QB22 := 79;
            %QB23 := 70;
            %QB24 := 51;
            %QB25 := 10;
            %QB26 := 0;
            %QB27 := 0;
            %QB28 := 0;
            %QB29 := 0;
            %QB30 := 0;
            %QB31 := 0;
            %QB20 := 21;
            Conf_Delay3(IN := Aux_Conf3, PT:= T#50MS);
            Aux_Conf3 := TRUE;
            Conf_Delay3(IN := Aux_Conf3, PT:= T#50MS);
            IF Conf_Delay3.Q THEN
                Aux_Conf0 := FALSE;
                Aux_Conf1 := FALSE;
                Aux_Conf2 := FALSE;
                Aux_Conf3 := FALSE;
                Conf_OK := FALSE;
                Aux_485_Med := FALSE;
                Aux_485_Conf := TRUE;
            END_IF
        END_IF
    END_IF
END_IF
```

Sub-rotina de Medição (ST)

```
%QB21 := 77;
%QB22 := 83;
%QB23 := 86;
%QB24 := 63;
%QB25 := 10;
%QB20 := 21;
Leit_Delay0(IN:=Aux_Leitura0, PT:=T#50MS);
Aux_Leitura0 := TRUE;
Leit_Delay0(IN:=Aux_Leitura0, PT:=T#50MS);
IF Leit_Delay0.Q THEN
    %QB20 := 48;
    Leit_Delay1(IN:=Aux_Leitura1, PT:=T#50MS);
    Aux_Leitura1 := TRUE;
    Leit_Delay1(IN:=Aux_Leitura1, PT:=T#50MS);
    Aux_485_Conf := FALSE;
    IF Leit_Delay1.Q THEN
        Aux_Leitura0 := FALSE;
        Aux_Leitura1 := FALSE;
        Aux_485_Conf := FALSE;
        Byte_0 := %IB40;
        Byte_1 := %IB41;
        Byte_2 := %IB42;
        Byte_3 := %IB43;
        Byte_4 := %IB44;
        Byte_5 := %IB45;
        Byte_6 := %IB46;
        Byte_7 := %IB47;
        Byte_8 := %IB48;
        Byte_9 := %IB49;
        Byte_10 := %IB50;
        Byte_11 := %IB51;
        IF Byte_0>48 AND Byte_0<=59 THEN
            IF Byte_1=0 AND Byte_2=0 AND Byte_3=0 AND Byte_4=0 AND Byte_5=0 AND Byte_6=0 AND Byte_7=0 AND Byte_8=0 AND Byte_9=0 AND Byte_10=0 AND Byte_11=0 THEN
                Repetir := TRUE;
                Medicao_OK := FALSE;
                Aux_485_Med := TRUE;
                ELSE
                Leitura (Byte0:=Byte_8, Byte1:=Byte_7, Byte2:=Byte_6, Byte3:=Byte_5, Byte4:=Byte_4, Byte5:=Byte_3, Byte6:=Byte_2, Int_Num=>Int_Medicao);
                IF Byte_1=45 THEN
                    Forca_N := -1*Int_Medicao/20;
                    Forca_kgf :
                    Repetir := FALSE;
                    Medicao_OK := TRUE;
                    Aux_485_Med := TRUE;
                    ELSE
                    Forca_N := Int_Medicao/20;
                    Forca_kgf := Forca_N/9.80665;
                    Repetir := FALSE;
                    Medicao_OK := TRUE;
                    Aux_485_Med := TRUE;
                END_IF

            END_IF
            ELSE
            Medicao_OK := FALSE;
            Repetir := TRUE;
            Aux_485_Med := TRUE;
        END_IF
    END_IF
END_IF
```

Sub-rotina de Re*set* dos Parâmetros (ST)

```
Repetir := FALSE;
Medicao_OK := FALSE;
Aux_485_Med := FALSE;
Aux_485_Conf := TRUE;
```

Sub-rotina de Retorno ao Início do Programa (ST)

```
Comunicacao_OK := TRUE;
```

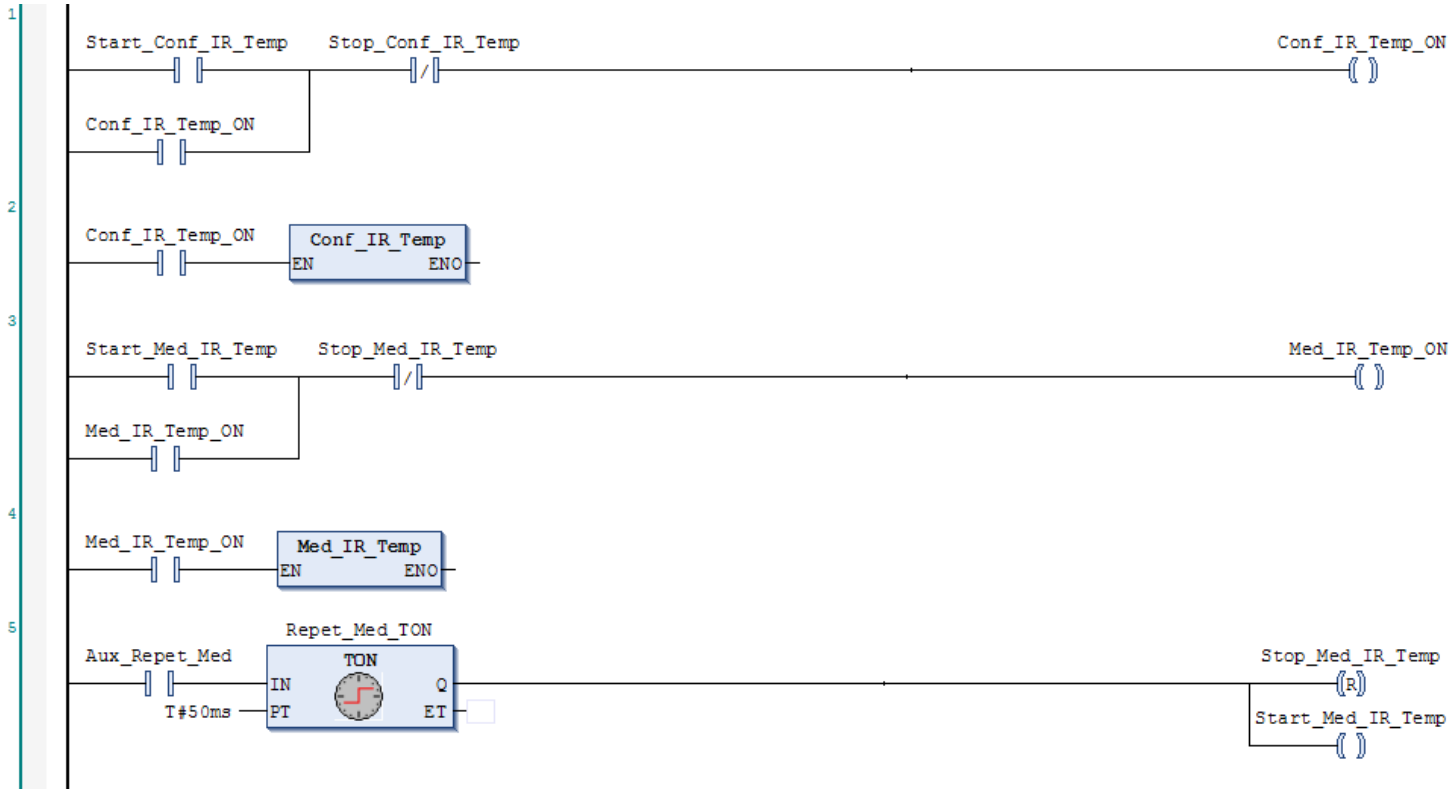
Função de Conversão de 7 *Bytes* em um Número Inteiro (ST)

```
CASE Byte0 OF
    48: Num0 := 0;
    49: Num0 := 1;
    50: Num0 := 2;
    51: Num0 := 3;
    52: Num0 := 4;
    53: Num0 := 5;
    54: Num0 := 6;
    55: Num0 := 7;
    56: Num0 := 8;
    57: Num0 := 9;
END_CASE
CASE Byte1 OF
    48: Num1 := 0;
    49: Num1 := 1;
    50: Num1 := 2;
    51: Num1 := 3;
    52: Num1 := 4;
    53: Num1 := 5;
    54: Num1 := 6;
    55: Num1 := 7;
    56: Num1 := 8;
    57: Num1 := 9;
END_CASE
CASE Byte2 OF
    48: Num2 := 0;
    49: Num2 := 1;
    50: Num2 := 2;
    51: Num2 := 3;
    52: Num2 := 4;
    53: Num2 := 5;
    54: Num2 := 6;
    55: Num2 := 7;
    56: Num2 := 8;
    57: Num2 := 9;
END_CASE
CASE Byte3 OF
    48: Num3 := 0;
    49: Num3 := 1;
    50: Num3 := 2;
    51: Num3 := 3;
    52: Num3 := 4;
    53: Num3 := 5;
    54: Num3 := 6;
    55: Num3 := 7;
    56: Num3 := 8;
    57: Num3 := 9;
END_CASE
CASE Byte4 OF
    48: Num4 := 0;
    49: Num4 := 1;
    50: Num4 := 2;
    51: Num4 := 3;
    52: Num4 := 4;
    53: Num4 := 5;
    54: Num4 := 6;
    55: Num4 := 7;
    56: Num4 := 8;
    57: Num4 := 9;
END_CASE
CASE Byte5 OF
    48: Num5 := 0;
    49: Num5 := 1;
    50: Num5 := 2;
    51: Num5 := 3;
    52: Num5 := 4;
    53: Num5 := 5;
    54: Num5 := 6;
    55: Num5 := 7;
    56: Num5 := 8;
    57: Num5 := 9;
END_CASE
CASE Byte6 OF
    48: Num6 := 0;
    49: Num6 := 1;
    50: Num6 := 2;
    51: Num6 := 3;
    52: Num6 := 4;
    53: Num6 := 5;
    54: Num6 := 6;
    55: Num6 := 7;
    56: Num6 := 8;
    57: Num6 := 9;
END_CASE
Int_Num := Num0 + 10*Num1 + 100*Num2 + 1000*Num3 + 10000*Num4 + 100000*Num5 + 1000000*Num6;
```

APÊNDICE B

Programa do Sensor de Temperatura Infravermelho

Rotina Principal (LD)



Sub-rotina de Configuração (ST)

```
%QB21 := 0;
%QB22 := 0;
%QB23 := 0;
%QB24 := 0;
%QB25 := 0;
%QB26 := 0;
%QB27 := 0;
%QB28 := 0;
%QB29 := 0;
%QB30 := 0;
%QB31 := 0;
%QB20 := 0;

%QB21 := 36;
%QB22 := 1;
%QB23 := 116;
%QB24 := 13;
%QB25 := 10;
%QB20 := 64;

Conf_TON(IN := Aux_Conf, PT:= T#50MS);
Aux_Conf := TRUE;
Conf_TON(IN := Aux_Conf, PT:= T#50MS);
IF Conf_TON.Q THEN
    Aux_Conf := FALSE;
    Start_Conf_IR_Temp := FALSE;
    Start_Med_IR_Temp := TRUE;
    Stop_Med_IR_Temp := FALSE;
    Stop_Conf_IR_Temp := TRUE;
END IF
```

Sub-rotina de Medição (ST)

```
Aux_Repet_Med := FALSE;
%QB21 := 63;
%QB22 := 84;
%QB23 := 13;
%QB20 := 19;

Med_TON_1(IN:=Aux_Med_1, PT:=T#50MS);
Aux_Med_1 := TRUE;
Med_TON_1(IN:=Aux_Med_1, PT:=T#50MS);
IF Med_TON_1.Q THEN
    %QB20 := 48;
    Med_TON_2(IN:=Aux_Med_2, PT:=T#50MS);
    Aux_Med_2 := TRUE;
    Med_TON_2(IN:=Aux_Med_2, PT:=T#50MS);
    IF Med_TON_2.Q THEN
        IF %IB40=56 THEN
            Byte_1 := %IB41;
            Byte_2 := %IB42;
            Byte_3 := %IB43;
            Byte_4 := %IB44;
            Byte_5 := %IB45;
            Byte_6 := %IB46;
            Byte_7 := %IB47;
            Byte_8 := %IB48;
            Byte_9 := %IB49;
            Byte_10 := %IB50;

            Leitura_Temp (Byte0:=Byte_3, Byte1:=Byte_4, Byte2:=Byte_5, Byte3:=Byte_6, Byte4:=Byte_8, Int_Num=>Int_Temperatura);

            Start_Med_IR_Temp := FALSE;
            Aux_Med_1 := FALSE;
            Aux_Med_2 := FALSE;
            Aux_Repet_Med := TRUE;
            Stop_Med_IR_Temp := TRUE;
            ELSE
            Start_Med_IR_Temp := FALSE;
            Aux_Med_1 := FALSE;
            Aux_Med_2 := FALSE;
            Start_Conf_IR_Temp :=TRUE;
            Stop_Conf_IR_Temp := FALSE;
            Stop_Med_IR_Temp := TRUE;
        END_IF
    END_IF
END_IF
```

Função de Conversão de 7 *Bytes* em um Número Inteiro (ST)

```
CASE Byte0 OF
    48: Num0 := 0;
    49: Num0 := 1;
    50: Num0 := 2;
    51: Num0 := 3;
    52: Num0 := 4;
    53: Num0 := 5;
    54: Num0 := 6;
    55: Num0 := 7;
    56: Num0 := 8;
    57: Num0 := 9;
END_CASE
CASE Byte1 OF
    48: Num1 := 0;
    49: Num1 := 1;
    50: Num1 := 2;
    51: Num1 := 3;
    52: Num1 := 4;
    53: Num1 := 5;
    54: Num1 := 6;
    55: Num1 := 7;
    56: Num1 := 8;
    57: Num1 := 9;
END_CASE
CASE Byte2 OF
    48: Num2 := 0;
    49: Num2 := 1;
    50: Num2 := 2;
    51: Num2 := 3;
    52: Num2 := 4;
    53: Num2 := 5;
    54: Num2 := 6;
    55: Num2 := 7;
    56: Num2 := 8;
    57: Num2 := 9;
END_CASE
CASE Byte3 OF
    48: Num3 := 0;
    49: Num3 := 1;
    50: Num3 := 2;
    51: Num3 := 3;
    52: Num3 := 4;
    53: Num3 := 5;
    54: Num3 := 6;
    55: Num3 := 7;
    56: Num3 := 8;
    57: Num3 := 9;
END_CASE
CASE Byte4 OF
    48: Num4 := 0;
    49: Num4 := 1;
    50: Num4 := 2;
    51: Num4 := 3;
    52: Num4 := 4;
    53: Num4 := 5;
    54: Num4 := 6;
    55: Num4 := 7;
    56: Num4 := 8;
    57: Num4 := 9;
END_CASE
CASE Byte5 OF
    48: Num5 := 0;
    49: Num5 := 1;
    50: Num5 := 2;
    51: Num5 := 3;
    52: Num5 := 4;
    53: Num5 := 5;
    54: Num5 := 6;
    55: Num5 := 7;
    56: Num5 := 8;
    57: Num5 := 9;
END_CASE
CASE Byte6 OF
    48: Num6 := 0;
    49: Num6 := 1;
    50: Num6 := 2;
    51: Num6 := 3;
    52: Num6 := 4;
    53: Num6 := 5;
    54: Num6 := 6;
    55: Num6 := 7;
    56: Num6 := 8;
    57: Num6 := 9;
END_CASE
Int_Num := Num0 + 10*Num1 + 100*Num2 + 1000*Num3 + 10000*Num4 + 100000*Num5 + 1000000*Num6;
```