UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

GUILHERME MINHOLI GALANA

SISTEMA EMBARCADO DE AQUISIÇÃO E GESTÃO DA INFORMAÇÃO PARA CONTROLE OPERACIONAL DE UMA USINA SUCROALCOOLEIRA

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2017

GUILHERME MINHOLI GALANA

SISTEMA EMBARCADO DE AQUISIÇÃO E GESTÃO DA INFORMAÇÃO PARA CONTROLE OPERACIONAL DE UMA USINA SUCROALCOOLEIRA

Trabalho de conclusão de curso, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELEN - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador: Prof. Dr. Roberto Ribeiro Neli

CAMPO MOURÃO

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

SISTEMA EMBARCADO DE AQUISIÇÃO E GESTÃO DA INFORMAÇÃO PARA CONTROLE OPERACIONAL DE UMA USINA SUCROALCOOLEIRA

por

GUILHERME MINHOLI GALANA

Trabalho de Conclusão de Curso apresentado no dia 27 de novembro de 2017 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho _____ (aprovado, aprovado com restrições ou reprovado).

_____
Prof. André Luiz Regis Monteiro
(UTFPR)

_____
Prof. Lucas Ricken Garcia
(UTFPR)

_____
Prof. Roberto Ribeiro Neli
(UTFPR)
Orientador

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

RESUMO

Este trabalho estudou o desenvolvimento de uma ferramenta que aulixa o gestor a tomar decisões para gerenciar e planejar a logística de funcionamento de uma usina sucroalcooleira. Para isto foi desenvolvido um sistema de aquisição de informação em pontos remotos espalhados geograficamente e compartilhados através de uma rede. Para cada ponto do processo produtivo foi utilizado um sistema embarcado em conjunto a sensores pertinentes ao processo monitorado. Como são processos distintos em cada etapa produtiva, há uma variedade de sensores como, por exemplo, ultrassonico, umidade do solo, temperatura, luminosidade, radio frequência, vibração, entre outros. O sistema embarcado utilizado foi a Raspberry-Pi, que possui processamento robusto e vários modos de conectividade. Para a rede foram utilizados protocolos industriais e conexão via cabo.

Palavras chave: **Raspberry-PI, Logística, Rede, Sensores**

ABSTRACT

This work aims to develop a tool that empowers the manager to make decisions to manage and plan the logistics of running a sugar and ethanol plant. For this, an information acquisition system was developed at remote points spread geographically and shared through a network. For each point in the production process, a system was used in conjunction with sensors that are relevant to the monitored process. As they are distinct processes in each productive stage, there are a variety of sensors such as ultrasonic, soil moisture, temperature, luminosity, radio frequency, vibration, among others. The embedded system used was Raspberry-Pi, which features robust processing and multiple modes of connectivity. For the network were used industrial protocols and cable connection.

Keywords: **Raspberry-Pi, Logistics, Networking, Sensors**

SUMÁRIO

1 INTRODUÇÃO

As usinas sucroalcooleiras têm grande representatividade no setor de agronegócio brasileiro. Segundo o quarto levantamento de 2016 da Companhia Nacional de Abastecimento (CONAB), a produção de cana-de-açúcar atingiu 665,6 milhões de toneladas na safra 2015/16; gerando 33,49 milhões de toneladas de açúcar e 30,5 bilhões de litros de etanol, principais subprodutos da cana (CONAB, 2016).

Segundo dados da União das Indústrias de Cana-de-açúcar (UNICA), o setor gerou 1,2 milhões de empregos diretos e um Produto Interno Bruto (PIB) de 48 bilhões de dólares. A produção nacional representa 25% da produção mundial de açúcar e 20% de etanol (ÚNICA, 2017).

Assim como diversos segmentos industriais, o setor de agronegócio segue a tendência de desenvolvimento tecnológico para melhorar a produção e elevar a competitividade. Eid (1996) fez um estudo de caso em várias usinas do seguimento, a fim de constatar os benefícios do processo de modernização. As usinas que investiram em tecnologia, demonstraram ter um desempenho superior em produtividade de matéria prima, melhor uso de equipamentos mecanizados, melhor gestão da produção automatizada e, consequentemente, menor custo de produção em relação às outras.

Silva (2009) desenvolveu um planejamento multiobjetivo, que faz a análise de vários fatores para escolher o melhor modo de operação para vários estágios de uma usina sucroalcooleira em Minas Gerais. Ele comprovou a eficácia do planejamento e tomada de decisões, melhorando a eficiência e alocação de recursos, em relação à safra do ano anterior, gerando um balanço financeiro ainda melhor.

Portanto, para melhor desempenho produtivo, o gestor deve avaliar várias alternativas, baseando-se em dados estatísticos, informações climáticas e dos processos, status de funcionamento e análise de desempenho de cada estágio da produção. O retorno monetário e a margem de lucro são consequências da eficiência do sistema como um todo. Percebe-se então a importância da aquisição e acessibilidade da informação, além da comunicação entre processos para formular as melhores estratégias de operação.

O’Brien (2004) define Sistema de Informação (S.I.) como um conjunto de hardware, software, rede de comunicação, armazenamento e disposição da informação em uma organização.

Weill e Ross (2006) apresentam a importância e influência da Tecnologia de Informação (TI) no desempenho empresarial. Dizem que o crescimento da TI é elemento fundamental na eficiência, inovação, crescimento e integração dos negócios de uma empresa.

Este trabalho visa desenvolver um sistema para realizar a aquisição de informação e monitorar o fluxo produtivo de uma usina sucroalcooleira, desde a produção da matéria prima, até a estocagem do produto final. Desta forma, o gestor pode otimizar o controle, desempenho e uso dos recursos e, consequentemente, elevar a eficiência operacional da usina reduzindo os custos de produção.

1.1 JUSTIFICATIVA

O segmento de cana-de-açúcar no Brasil movimenta bilhões de dólares anualmente, além de milhões de empregos diretamente. O desenvolvimento tecnológico industrial deve ser integrado pelo setor para obter melhor rentabilidade e competitividade no cenário mundial.

Os estados de São Paulo e Paraná se destacam em produtividade de toneladas de cana por alqueire, em relação à outras regiões do país por dois principais motivos: clima, fundamental para qualquer cultura agrária, e a tecnologia utilizada.

O desenvolvimento de um sistema para gerenciamento e coleta de informação aplicado na cultura da cana-de-açúcar pode gerar um grande volume de dados sobre o solo, clima, qualidade da matéria prima e dos subprodutos, além do monitoramento dos equipamentos mecanizados da produção. O sistema propicia uma redução da frota de caminhões, tratores, colhedoras, caçambas de irrigação, tempo parado da linha de produção e volume maior de matéria prima, pois com a aquisição e gestão da informação é possível planejar o uso do equipamento de forma compartilhada e prever possíveis problemas mecânicos que desativariam a produção momentaneamente.

Os dados influenciam no modo de operação da usina, consequentemente, otimizam e reduzem os custos operacionais, elevando a eficiência nas lavouras e linha produtiva. Com diagnósticos em tempo real e mais precisos, a eficiência e interação entre os meios produtivos podem se ajustar aos melhores quadros operacionais.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um sistema capaz de coletar dados de vários estágios da produção de uma usina sucroalcooleira através de sensores, compartilhá-los dentro da rede e dispor estas informações à um gestor.

1.2.2 Objetivo específico

- Subdividir as etapas de funcionamento de uma usina sucroalcooleira, onde cada etapa tem suas responsabilidades.
- Desenvolver algoritmos que façam a interação com sensores e rede de comunicação e implementá-los em sistema embarcado.
- Cada sistema embarcado assume o papel de estação onde é responsável por gerenciar um conjunto de sensores/atuadores, dispor as informações na rede e atuar conforme instruções recebidas.
- Desenvolver uma rede fechada onde a comunicação ocorra entre as estações por meio de protocolos industriais.
- Desenvolver uma estação que gerencie e atue nas outras, além de traduzir as informações dispostas na rede.

2 REVISÃO BIBLIOGRÁFICA

2.1 SENSORES

Sensores transformam condições do ambiente em sinais interpretáveis por um sistema eletricoeletrônico. Os sensores são compostos por transdutores, que fazem o papel de transformar uma grandeza física - temperatura, umidade, pressão, entre outros-, em sinal elétrico interpretável por um circuito (THOMAZINI, ALBUQUERQUE, 2005).

2.1.1 Sensor de distância ultrassônico

O sensor usa o princípio de propagação de ondas sonoras em uma frequência acima da faixa audível. Normalmente o sensor é composto por um emissor e um receptor. O emissor pode ser magnetostritivo ou piezoelétrico, que tem o papel de transformar um sinal elétrico em onda sonora. A onda então se propaga pelo meio até encontrar um obstáculo, uma pequena porção do sinal é absorvida e o resto é refletido e recebido pelo receptor do sensor (THOMAZINI, ALBUQUERQUE, 2005).

Se marcar o tempo entre a emissão e a recepção do sinal, é possível obter-se a distância através da equação da velocidade média:

$$V_m = \frac{\Delta S}{\Delta T} \qquad [1]$$

Sendo:

- V_m é Velocidade média.
- ΔS é Variação do espaço.
- ΔT é a variação do tempo.

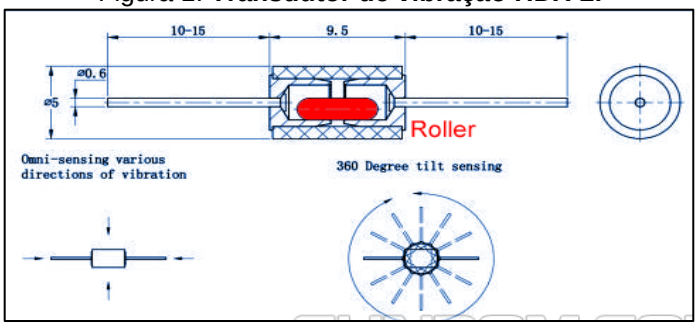
Adotando Vm como a velocidade do som no ar, 340 m/s, o tempo obtido com base no funcionamento do sensor é considerada a ida até o obstáculo e a volta da

onda refletida; portanto para obter a distância até o objeto é utilizado metade do tempo.

2.1.2 Sensor de vibração

Pela definição do dicionário, vibração pode ser definida como um movimento aleatório de um corpo em relação ao seu centro de massa, equilíbrio ou de um ponto de referência (AURÉLIO, 2017). Os dispositivos eletrônicos sensíveis a vibração mecânica, transformam este movimento em sinal elétrico interpretável, por exemplo o sensor HDX-2, que usa a vibração para desativar a condução de corrente elétrica entre seus terminais. A Figura 1 mostra a composição do transdutor.

Figura 1. **Transdutor de vibração HDX-2.**

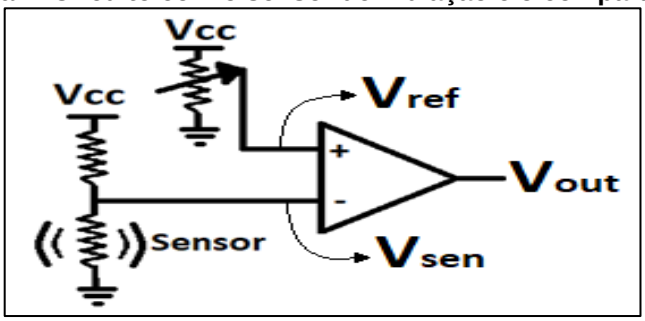


Fonte: (BESTEP, 2017)

O dispositivo é formado por dois terminais condutores separados por um espaço não condutor. Há também um componente chamado *Roller,* que faz a conexão entre os dois terminais. Quando estático, a conexão é bem-sucedida e há condução de corrente elétrica entre os dois terminais. Uma vibração em qualquer orientação do eixo das coordenadas x, y e z - a ponte é desconectada e provoca uma descontinuidade da condução de corrente entre os terminais. A condução descontinua sob uma tensão constante provoca uma variação de resistência (BESTEP, 2017).

Portanto o transdutor pode ser interpretado como uma resistência variada, que quando ligá-lo à um comparador de tensão, é possível calibrar o sensor para acionar uma saída a partir de uma tensão de referência. Na Figura 2, é possível ver um esquema do comparador de tensão.

Figura 2. **Circuito com o sensor de vibração e o comparador.**



Fonte: Autoria Própria

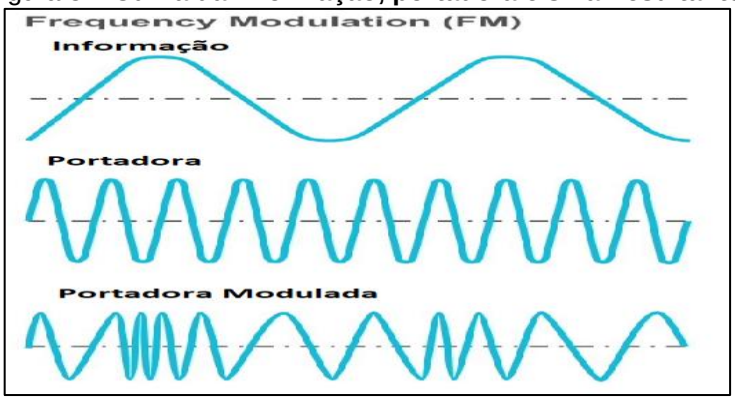
Nesse tipo de circuito, quando a tensão do sensor (Vsen) é maior que a tensão de referência (Vref), tem-se VCC na saída (Vout), caso contrário, GND. É possível definir a tensão de referência regulada por um potenciômetro e desta forma calibrar a sensibilidade do sensor (BOYLESTAD, 2004).

2.1.3 Receptor de dados modulados em frequência.

A transmissão de informação sem fio é muito usada em sistemas que compartilham dados, neste tipo de comunicação os dados são enviados ao destino por meio de ondas eletromagnéticas através do ar.

Os transmissores de Radiofrequência modulam a frequência instantânea de uma onda senoidal, denominada portadora, para conter a informação a ser transmitida. A Figura 3 mostra as ondas de informação, portadora e o sinal resultante (RADIO ACADEMY, 2017).

Figura 3 – **Curva da informação, portadora e sinal resultante.**



Fonte: (RADIO ACADEMY, 2017) - Adaptada.

Através da figura, é possível perceber a influência da informação na frequência do sinal de transmissão. A frequência da portadora deve estar dentro do espectro de radiofrequência e respeitar as normas definidas pela ANATEL - Agencia Nacional de Telecomunicações.

Portanto o receptor de sinais modulados em frequência gera a portadora para conseguir ser sensível a variações de frequência instantânea do sinal recebido e conseguir extrair a informação (RADIO ACADEMY, 2017).

2.1.4 Sensor de Temperatura NTC

O NTC (*Negative Temperature Coenficient*), é um termistor, ou seja, um dispositivo elétrico que varia a sua resistência de acordo com a temperatura em seu corpo. O NTC tem coeficiente negativo, que se refere a queda de resistência com o aumento da temperatura. Esse tipo de comportamento é devido aos compostos químicos utilizados em sua fabricação, óxidos metálicos tais como o Cromo, Níquel, Cobre, Ferro, Manganês e titânio (SOLOMAN, 2009).

A Figura 4 apresenta a variância da resistência de um NTC 10 K de grau em grau, dada suas especificações técnicas fornecidas pelo fabricante (EPCOS, 2006).

Figura 4 – **Tabela com Resistencia e temperatura de um NTC 10k**

R (25℃) : 10KΩ　　B(25℃/50℃):3950

T (℃)	R (kΩ)	T (℃)	R (KΩ)	T (℃)	R (kΩ)	T (℃)	R (kΩ)
-25	133.500	15	15.7511	55	2.96331	95	.781670
-24	125.672	16	15.0306	56	2.85569	96	.758701
-23	118.350	17	14.3472	57	2.75256	97	.736519
-22	111.498	18	13.6987	58	2.65369	98	.715094
-21	105.084	19	13.0833	59	2.55890	99	.694397
-20	99.0773	20	12.4990	60	2.46799	100	.674400
-19	93.4469	21	11.9441	61	2.38080	101	.655075
-18	88.1750	22	11.4169	62	2.29714	102	.636398
-17	83.2296	23	10.9161	63	2.21686	103	.618345
-16	78.5909	24	10.4400	64	2.13980	104	.600890
-15	74.2384	25	10.0000	65	2.06583	105	.584013
-14	70.1527	26	9.55693	66	1.99480	106	.567690
-13	66.3162	27	9.14743	67	1.92658	107	.551902
-12	62.7122	28	8.75777	68	1.86105	108	.536629
-11	59.3254	29	8.38690	69	1.79809	109	.521852
-10	56.1416	30	8.03380	70	1.73758	110	.507552
-9	53.1475	31	7.69753	71	1.67942	111	.493712
-8	50.3307	32	7.37721	72	1.62351	112	.480316
-7	47.6799	33	7.07200	73	1.56975	113	.467346
-6	45.1842	34	6.78110	74	1.51804	114	.454788
-5	42.8339	35	6.50378	75	1.47300	115	.442627
-4	40.6197	36	6.23934	76	1.42045	116	.430848
-3	38.5330	37	5.98711	77	1.37439	117	.419438
-2	36.5656	38	5.74646	78	1.33007	118	.408384
-1	34.7103	39	5.51680	79	1.28740	119	.397674
0	32.9600	40	5.29758	80	1.24632	120	.387294
1	31.3081	41	5.08828	81	1.20675	121	.377233
2	29.7487	42	4.88838	82	1.16864	122	.367481
3	28.2760	43	4.69743	83	1.13193	123	.358026
4	26.8848	44	4.51498	84	1.09655	124	.348859
5	25.5702	45	4.34060	85	1.06246	125	.339968
6	24.3274	46	4.17391	86	1.02960		
7	23.1523	47	4.01452	87	.997924		
8	22.0407	48	3.86207	88	.967376		
9	20.9889	49	3.71624	89	.937916		
10	19.9934	50	3.58800	90	.909498		
11	19.0509	51	3.44314	91	.882083		
12	18.1582	52	3.31529	92	.855630		
13	17.3124	53	3.19287	93	.830101		
14	16.5109	54	3.07563	94	.805459		

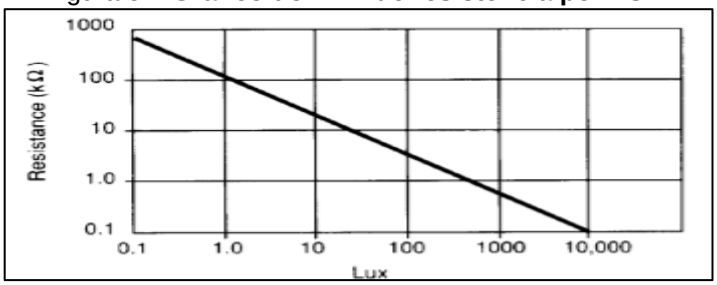
Fonte: (EPCOS, 2006)

2.1.5 Sensor de luminosidade LDR

Os LDR’s (*Light Dependent Resistor*) são dispositivos que tem sua resistência atrelada a quantidade de luz que incide sobre ele. É um componente do tipo semicondutor que absorve a luz e a transforma em calor. Este aumento de temperatura faz com que mais elétrons migrem para banda de condução, ou seja, a resistência elétrica do material reduz (THOMAZINI, ALBUQUERQUE, 2005).

A figura a seguir apresenta o gráfico da relação entre resistência e a quantidade de luz incidente medida em LUX.

Figura 5 – **Gráfico do LDR de resistência por LUX.**



Fonte: (LPRS, 2012)

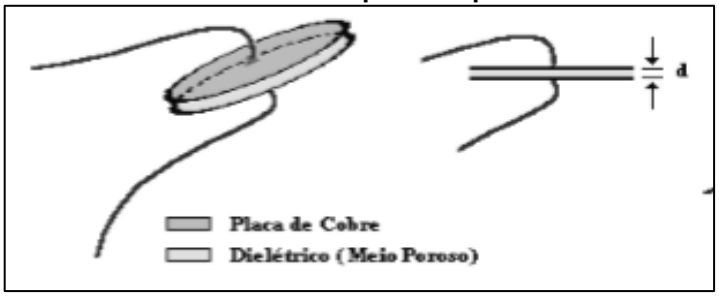
Um Lux corresponde a incidência perpendicular de um lúmen em uma superfície de 1 metro quadrado padronizado pelo sistema internacional de medidas. Pelo gráfico é possível ver que quanto menos LUX, mais resistente se torna o componente (INMETRO, 2012).

2.1.6 Sensor de umidade de solo

Uma técnica para medir a umidade do solo é o princípio da variação da capacitância. Alguns sensores capacitivos são constituídos de placas condutoras paralelas, que são separadas por um material isolante ou dielétrico e alimentadas com potenciais elétricos opostos. A capacidade desses componentes armazenarem energia é conhecida como capacitância. Variando apenas o material dielétrico, há uma variação de capacitância (HALLIDAY, 2002).

A Figura 6 mostra a disposição das placas separadas por um material dielétrico de espessura “d”.

Figura 6 – **Placas condutoras separadas por material dielétrico.**



Fonte: (BORIM; PINTO, SD).

Portanto para medir a umidade do solo, o sensor dispõe de placas condutoras paralelas alimentadas com cargas opostas e usa o solo como material dielétrico. A quantidade de água presente na terra, altera o valor do material dielétrico e consequentemente a capacitância (BORIM; PINTO, SD). A Figura 7 mostra o sensor EC-5 fabricado pela Decagon.

Figura 7 – **Sensor de umidade de solo capacitivo EC-5**

Fonte: (DECAGON, 2017)

A vantagem de um sensor capacitivo para esta aplicação, é que o sensor tem um bom desempenho independente da composição química ou granulometria do solo. Com a tensão em cima do sensor capacitivo é possível usar o circuito comparador de tensão para também controlar a sensibilidade e o valor da tensão de saída.

2.1.7 Conversor Analógico/Digital

Um sinal analógico é um sinal contínuo que varia com o tempo e pode assumir infinitos valores. Já os sinais digitais têm uma representação limitada a dois valores, alto/baixo ou ligado/desligado (LATHI, 2007).

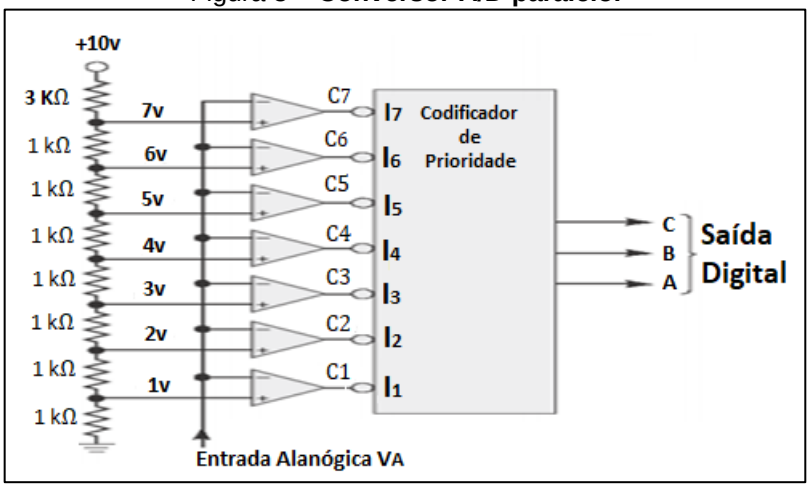
Sistemas computacionais utilizam a forma digital para expressar valores, portanto, para tratar uma informação analógica proveniente de uma grandeza física, é necessário fazer a conversão do sinal.

Um conversor então deve pegar uma amostra do sinal analógico de tempo contínuo para fazer uma representação da amplitude neste instante. Este processo

de quantizar o sinal em amostra é denominado de amostragem ou discretização – transformar um sinal contínuo em discreto no tempo (LATHI, 2007).

É feita uma relação entre o sinal amostrado instantaneamente e uma tensão de referência. O resultado desta comparação é expressa em um conjunto de número digitais. A Figura 8 mostra um conversor Analógico/Digital (A/D) paralelo para entender o funcionamento.

Figura 8 – **Conversor A/D paralelo.**



Fonte: (TOCCI; WIDMER; MOSS, 2007), adaptado.

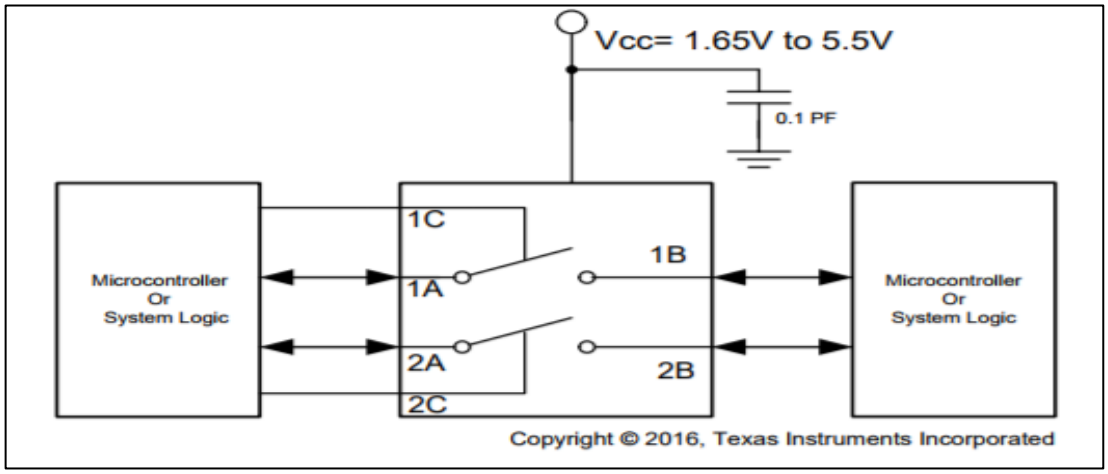
O sinal de referência da alimentação ligado à uma associação de resistores provoca diferentes níveis de tensão em cada estágio de comparação - do C7 ao C1 na figura. Um sinal analógico amostrado no barramento “Entrada analógica Va” da figura, é comparado com os vários níveis de tensão. Caso maior ou igual ao nível comparado, o sinal de saída deste estagio de comparação é representado digitalmente como verdadeiro ou alto. No conversor da figura, a saída está representada em 3 bits paralelos (TOCCI; WIDMER; MOSS, 2007).

2.2 PORTAS LÓGICAS

Portas logicas são dispositivos que trabalham o sinal de maneira racional, relacionando uma ou mais entradas para obter uma única saída. A porta *NOT* tem como função inverter logicamente o sinal de entrada. Digitalmente falando, seria transformar o “zero” da entrada em “um” na saída (TOCCI; WIDMER; MOSS, 2007).

O circuito integrado *Switch* é uma chave lógica, no qual com a aplicação de um sinal de controle, o CI permite que uma entrada seja conectada à uma saída de forma direta. A Figura 9 mostra a lógica interna de um *switch* de dois canais da *Texas Instruments* (TEXAS INSTRUMENTS, 2016).

Figura 9 – **Switch de dois canais com controle independente.**



Fonte: (TEXAS INSTRUMENTS, 2016).

2.3 *RASPBERRY PI*

O *Raspberry Pi* é um computador de tamanho reduzido e de baixo custo, capaz de controlar processos, conectar-se com diversas fontes de comunicação e com robustez em processamento de dados. Foi criado pela Fundação *Raspberry Pi* que nasceu em 2003, na faculdade de *Cambridge* - Inglaterra, com intuito de promover educação e programação de baixo custo (RASPBERRYPI, 2017).

Existem vários modelos que se diferem pela conectividade, tamanho e capacidade/velocidade de processamento – números de núcleos, *clock* do processador, memória RAM. O Quadro 1 apresenta dois modelos e suas características de *hardware.*

A vantagem desse sistema é dispor de um Sistema Operacional (SO) embarcado. O SO é um conjunto de programas que gerencia os recursos de *hardware* e promove uma interface de interação entre usuário e máquina. Compiladores são recursos valiosos do SO e têm a função de traduzir códigos escritos em alto nível pelo programador, para linguagem de máquina executável (TANENBAUM, 2016).

Quadro 1 – **Características de Hardware dos modelos de Raspberry.**

	RASPBERRY PI 3 MODELO B+	**RASPBERRY PI 2 MODELO B**
CPU	1.2 GHZ quad-core ARM CORTEX-A53 – Broadcom	700 MHZ ARM11 – Broadcom
GPU	Videocore IV - 4 núcleos	Videocore IV- 4 núcleos
Memória RAM	1 GB	512 MB
Cartão de Memória	Micro-SD	SD card
Conectividade	4x USB, HDMI, Ethernet, 3.5 mm áudio jack	2x USB, HDMI, Ethernet, 3.5 mm áudio jack
Conectores	Interface de Câmera (CSI), GPIO, SPI, I2C, JTAG, Bluetooth, WI-FI	Interface de Câmera (CSI), GPIO, SPI, I2C, JTAG
Preço*	U$ 35.00	U$ 25.00
*** Cotado em outubro de 2017.**		

Fonte: (RASPBERRYPI, 2017).

2.4 COMUNICAÇÃO SERIAL

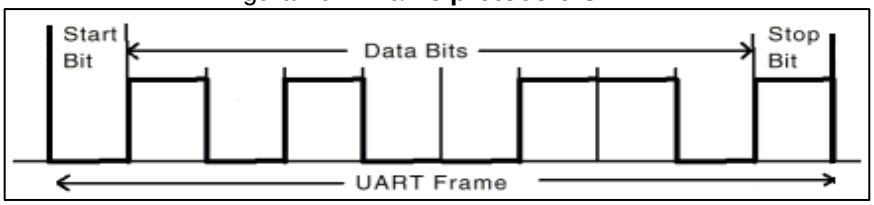
Na comunicação serial os bits de informação são enviados um a um em sequência por um canal de comunicação. Existem dois métodos de implementação da comunicação: síncrono - com conexão de *clock* entre os elementos que estão em comunicação -, e assíncrono sem o compartilhamento do *clock* (BOCCATO, 2017).

O UART (*Universal Asynchronous Receiver/Transmiter*) é um conjunto de regras que definem como o *frame* deve ser transmitido. Os parâmetros de *BaudRate* devem ser configurados, eles definem o número de bits enviados em 0,0001 s. São eles (BOCCATO, 2017):

- Paridade: 1 ou 2 bits com o objetivo de verificar erros no pacote.
- *Start e Stop:* Que marcam o início e o fim da mensagem.
- Dados*:* o tamanho da mensagem.

A Figura 10 mostra um *frame* orientado pelo protocolo UART, com um start e stop bit e 8 bits de mensagem.

Figura 10 – **Frame protocolo UART**



Fonte: (BOCCATO, 2017), adaptada.

Na conexão física existem vários métodos que viabilizam a comunicação pelo protocolo UART, a conexão *Transistor-Transistor Logic* (TTL), o protocolo RS-232, RS-485, entre outros. Estes protocolos definem o comportamento dos sinais elétricos que contém a informação. A logica TTL, é conexão direta de portas lógicas e/ou transistores. O protocolo RS-232 define o nível de tensão para o nível logico 1 entre -3 a -25 v e nível baixo entre +3 a +25. Além de valores lógicos invertidos, o padrão oferece mais robustez a interferências externas (ROCHA, 2013).

O protocolo RS-485 utiliza cabeamento par-trançado, que são fios entrelaçados com a finalidade de anular o ruído externo. Isto acontece porque os fios que transmitem os dados estão sempre com potenciais elétricos contrários, sendo assim o campo magnético tem orientação contrária e se atraem. A força de atração do campo tende ser mais forte que a interferência de campos magnéticos externos.

Além da UART, outro tipo de comunicação serial é a *Inter-Integrated Circuit (*I²C). Este sistema é do tipo síncrono e utiliza duas linhas bidirecionais, uma para o *clock (SCL)* e outra para dados (SDA). A velocidade da troca de informação está entre 100 a 10 Kbit/s. Este modelo de comunicação é do tipo mestre-escravo, onde o mestre inicia todas trocas de informações. Há um conjunto de instruções pré-definidas onde o mestre diz que deseja enviar ou receber dados. O mestre deve então mandar o endereço do escravo, a função e os dados (VERONESI, 2006).

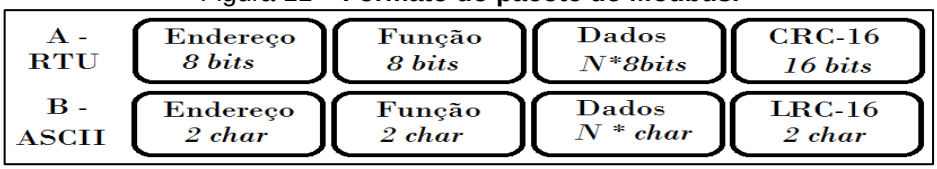
2.5 PROTOCOLO *MODBUS*

O protocolo *Modbus* foi criado pela *Modicon*, em 1979, para padronizar a troca de dados em controladores lógicos programáveis (PLC, *Programmable Logic Controllers*). Ele é do tipo livre, simples e robusto, que são os motivos de ser amplamente empregado em diversos equipamentos industriais e dispositivos que se comuniquem (GUARESE, 2011).

O protocolo é do modelo Mestre/Escravo e a forma de comunicação é serial, a qual pode ser transmitida em duas possíveis orientações: RTU (*Remote Terminal Unit*) e ASCII (*American Standard Code for Information Interchange).* No modo RTU, os dados são ordenados em formato binário de 8 bits ou representados por 2 números hexadecimais (2 bytes). No Modelo ASCII, o formato é de 7 bits, que forma letras, números e até acentos, tornando os dados legíveis. Neste caso, o tráfego fica mais intenso, pois as mensagens ficam maiores (MODBUS, 2017).

Portanto, a comunicação é organizada em um pacote ou envelope. Para entender a organização, definem-se campos destinados a cada informação pertinente à troca de dados. A Figura 11 mostra a disposição do pacote organizado em campos e identifica o tamanho em cada orientação do protocolo.

Figura 11 **– Formato do pacote do Modbus.**



Fonte: Autoria Própria.

Portanto, no primeiro campo deve ser inserido o endereço do escravo que o mestre deseja interagir. No modo RTU, o tamanho é de 8 bits ou dois números hexa, no ASCII são 2 *chars,* onde cada *char* tem tamanho de 8 bits, possibilitando formar dois caracteres do formato ASCII.

O campo da função define qual a interação será realizada: leitura, escrita, entre outros. Os valores do campo são pré-definidos pelo protocolo, por exemplo: se o campo contém o número binário 00000101, corresponde a escrita de um único bit. Os valores dos comandos são comuns aos dois modelos, RTU e ASCII.

O campo de dados é a área útil, onde a informação é disponibilizada. A escala é condicionada e limitada a cada dispositivo e sua tecnologia. Mas os valores se aproximam de até 256 bytes.

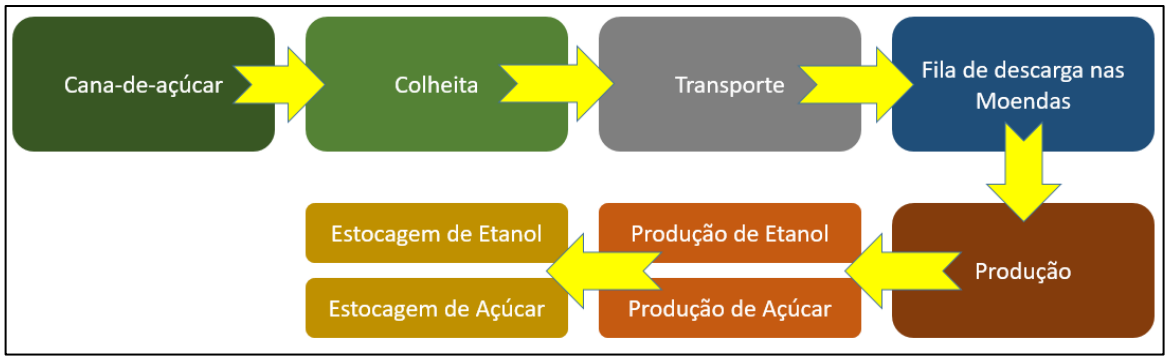
O último campo serve para verificação de validade do pacote formado, denominado CRC-16 (*Cyclic Redundancy Check*), de 16 bits, ou LRC (*Longitudinal Redundancy Check*) para ASCII. Ou seja, o algoritmo gera um valor de 16 bits a partir dos outros campos combinados. Ao receber o pacote, é possível verificar se a mensagem está correta ou corrompida com base no número gerado (MODBUSTOOLS, 2017).

3. METODOLOGIA

O sistema desenvolvido por este trabalho visa uma forma de aquisição de informação dos processos produtivos, e não a intenção de controlar as moendas, válvulas, motores e outros equipamentos mecanizados da linha de produção.

Fazendo a análise da produção de subprodutos da cana, como etanol e o açúcar, e desconsiderando os processos químicos ou físicos envolvidos, é possível montar o fluxo de produção e identificar pontos que requerem melhor controle para otimizar e reduzir os custos operacionais. A Figura 12 mostra o fluxograma da obtenção da matéria prima à estocagem do produto final para logística do sistema.

Figura 12 – **Fluxograma da logística do sistema.**



Fonte: Autoria Própria

Há várias fazendas que compõe o abastecimento da usina, e saber como está o desenvolvimento da cana possibilita ao gestor fazer o planejamento de ordem de colheita e determinar o deslocamento apenas do maquinário necessário.

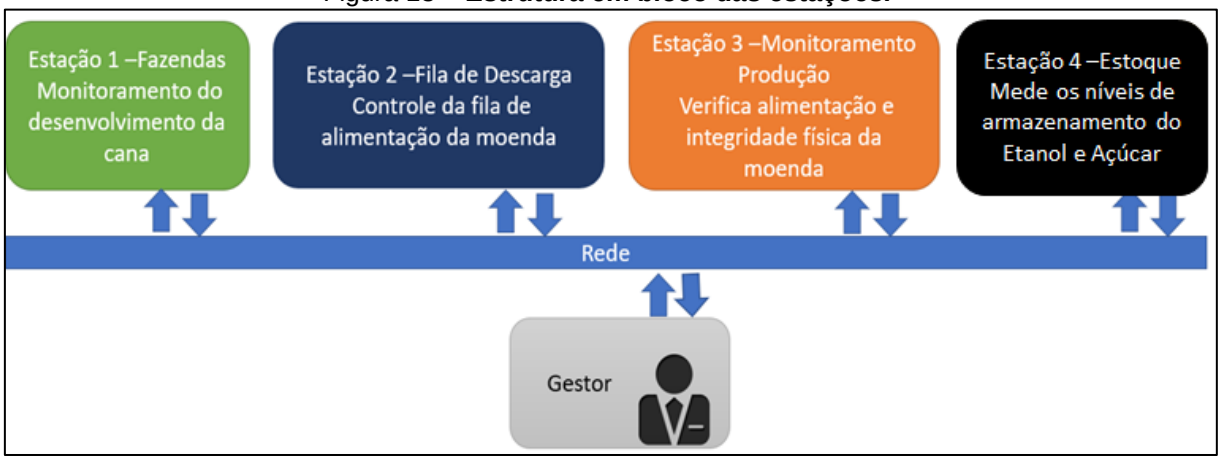
O controle da descarga é importante para vida útil e integridade física da moenda, além de manter a alimentação próxima ao nível excelente e reduzir a ociosidade na fila de descarga dos caminhões.

Monitorar componentes importantes da produção mecanizada pode gerar prognósticos mais precisos e agendamento de manutenção preventiva, sendo assim, reduz falhas mecânicas de grande gravidade, que aumenta o tempo parado de reparo.

O estoque dos produtos finais são parâmetros que possibilitam o gestor fazer avaliação da eficácia da produção da sua usina, são parâmetros para rentabilidade da matéria prima e demanda de abastecimento do mercado – sabendo o volume de saída dos produtos.

Portanto estudou-se o desenvolvimento de 4 estações que gerenciam um conjunto de sensores para monitorar o funcionamento/desenvolvimento das etapas de produção de álcool e açúcar na usina e dispor estes dados em uma rede fechada. O propósito do sistema é disponibilizar estas informações à um gestor para ele otimizar a logística e o quadro operacional da sua empresa. A Figura 13 mostra a estrutura em blocos das estações.

Figura 13 **– Estrutura em bloco das estações.**



Fonte: Autoria Própria.

Cada estação é um sistema embarcado Raspberry-pi, que é capaz de gerenciar sensores através de seus pinos de entrada e saída – *General Purpose Input/Output* (GPIO) -, além de ter dispositivos de comunicação serial UART e I²C. As configurações inicias que possibilitam o uso do sistema foram feitas de acordo com Richardson e Wallace, 2013.

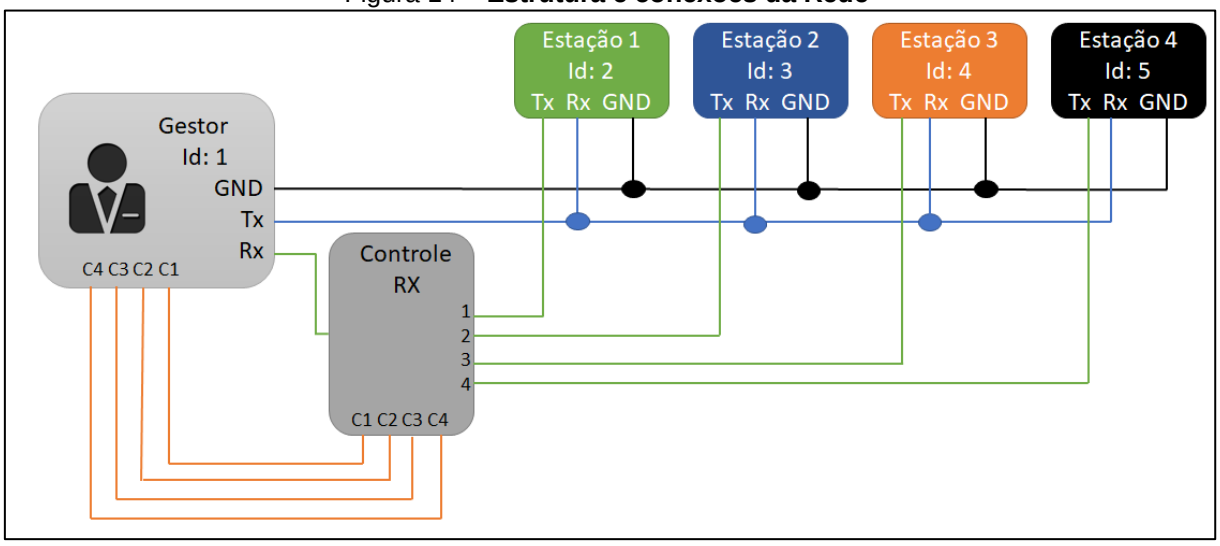
A rede pode ser estruturada em três elementos, modo de comunicação, protocolo e meio físico. O modo de comunicação consiste de como os bits são orientados, serial ou paralelo, se tem bit de *start* e *stop* e o tamanho de cada quadro. O protocolo define o *frame* da informação, a organização dos bits, quantidade de quadros que compõe o pacote e o que cada conjunto representa. O meio físico diz onde e como a informação vai ser transmitida para os elementos da rede, por meio de fio, ondas eletromagnéticas, luz, entre outros. Em uma analogia para melhor compreensão, o modo de comunicação ensina o sistema a “falar”, o protocolo é o idioma que todo mundo vai conversar e o meio físico coloca todo mundo na mesma sala, para ouvir e falar.

Sendo assim, o modo da rede é UART, e perite o ajuste do valor de BaudRate em 9600, 19200, 38400 e 115200. A flexibilidade deste ajuste permite a interação do sistema com outros dispositivos que fazem o uso da comunicação UART. Foi configurado também o tamanho da mensagem em 8 bits, sem bit de paridade, apenas *start* e *stop.*

O protocolo desenvolvido foi inspirado na estrutura de *frame* do Modbus, com campo de identificação, função, endereço, dados e verificação de segurança. A diferença está nas funções, que originalmente, tem valores pré-definidos pela Modicon. Sendo assim, as funções foram definidas de acordo com a estação que irá receber o pacote.

O meio físico é composto por cabos que interligam os pinos de transmissão (TX), recepção (RX) e a referência do sinal (GND) de todas estações. A estação que controla a rede, é quem inicia todas as comunicações e só permite a chegada da resposta da estação que faz interação. A estrutura da rede está exemplificada na figura 14.

Figura 14 – **Estrutura e conexões da Rede**



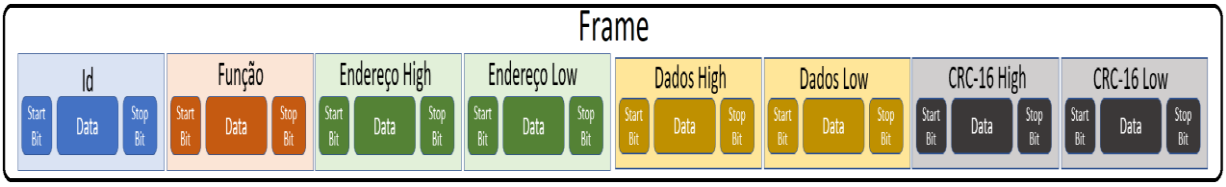
Fonte: Autoria Própria

O gestor manda um pacote na rede com o ID da estação na qual deseja comunicar e libera no bloco Controle Rx a conexão com a estação, por exemplo, se o gestor envia uma mensagem para estação 2, é feita a conexão do canal 2 (Tx estação 2) com o Rx do gestor. O controle do bloco “Controle RX” é feito através dos sinais

C1, C2, C3 e C4. Cada sinal seleciona uma entrada e conecta na saída. Não há conexões simultâneas.

Os *softwares* das estações montam o *frame* com 8 quadros UART. A figura 15 mostra o pacote enviado na rede.

Figura 15 – **Frame enviado na comunicação.**



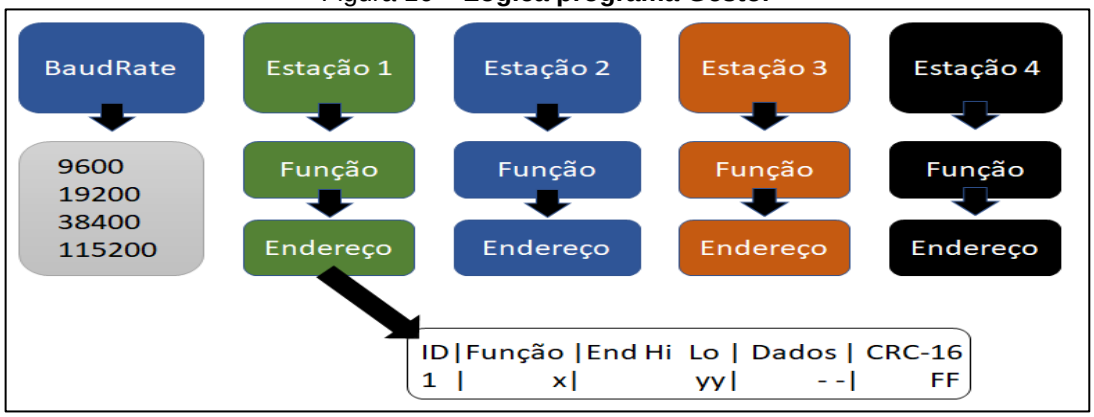
Fonte: Autoria Próprio

Cada quadro contém 10 bits, sendo que a informação tem 8 bits, equivalente à um byte ou dois números hexadecimais. Este tamanho de *data* permite uma escala de 256 valores. Nos campos que contém dois quadros, como endereço, dados e CRC-16, o tamanho é de 65536 valores.

Na Raspberry Pi 2 e 3, Tx e Rx, são os pinos 8 e 10 respectivamente. O endereço do dispositivo UART no Sistema Operacional é /dev/ttyAMA0 para o modelo 2 e /dev/serial0 para o modelo 3.

O programa do gestor é um menu de opções que gera o pacote a ser enviado na rede. Ele define todos os campos do *frame* de acordo com a informação que se deseja obter. A Figura 16 mostra a formulação do frame de acordo com as opções selecionadas.

Figura 16 – **Logica programa Gestor**



Fonte: Autoria Própria.

Os valores x e y são transparentes ao gestor, pois ele escolhe a informação e o programa gera os valores dos campos. Estes valores serão apresentados junto às respectivas estações.

3.1. ESTAÇÃO 1 - FAZENDA

A estação número 1 é responsável pelo monitoramento do desenvolvimento da cana-de-açúcar – matéria prima dos produtos gerados pela usina. Para avaliar o desenvolvimento desta cultura, algumas informações são importantes como temperatura, umidade do solo e intensidade luminosa.

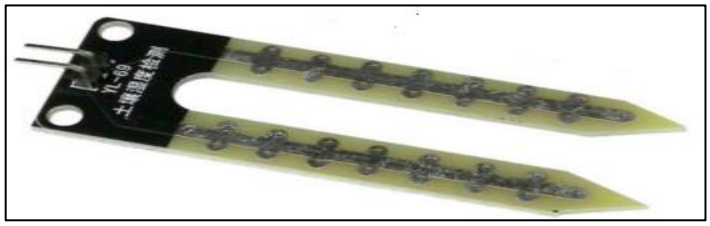
Para aquisição da temperatura local, utilizou-se um NTC 10k, disposto em um circuito divisor de tensão. À medida que a temperatura varia, a tensão de saída da relação também se altera. A saída interpretável pelo sistema é uma tensão analógica. A Figura 17 mostra a placa com o sensor desenvolvida neste trabalho.

Figura 17– **Placa com divisor de tensão e NTC 10k**

Fonte: Autoria Própria.

Para aquisição de umidade do solo foi utilizado o transdutor YL-69, que consiste em um sensor capacitivo de placas paralelas que utiliza a terra como material dielétrico. A umidade do solo varia a capacitância e, consequentemente, a tensão mensurada entre os terminais do dispositivo. A Figura 18, mostra o transdutor utilizado no trabalho.

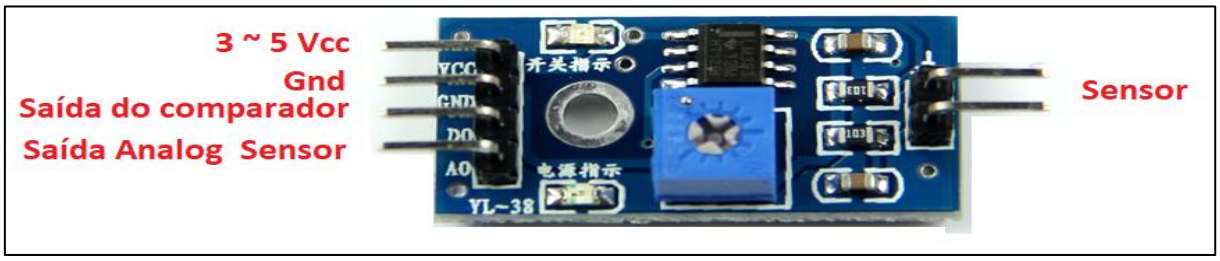
Figura 18 – **Transdutor YL-69, para umidade do solo.**



Fonte: Autoria Própria.

O sensor foi conectando na placa YL-38, que o alimenta, e permite trabalhar a saída da tensão variada, ou ligar em um comparador de tensão, configurando um ponto de referência. Neste caso, é definida a tensão de referência no potenciômetro e quando a tensão do sensor for maior, a saída do pino D0, que normalmente é nível lógico baixo, fica em nível alto. A Figura 19 mostra o *shild* YL-38 indicado o ponto de conexão do sensor.

Figura 19 - **Comparador de tensão YL-38.**



Fonte: Autoria Própria.

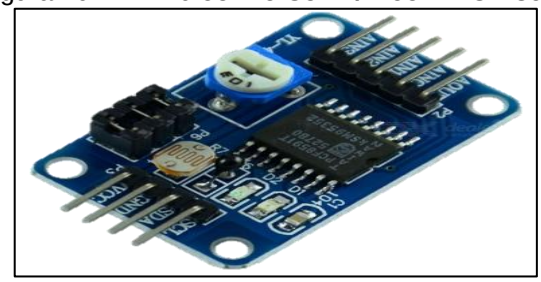
A lógica digital permite ao sistema saber o momento em que ouve irrigação, porque é definida uma tensão de referência para terra seca, e quando o solo for irrigado, a saída digital passa a ter nível alto. A saída A0, analógica, informa ao sistema a variação de umidade do solo, já que a quantidade de água altera o valor de tensão do sensor.

As fazendas separam grande áreas como talhões – área com tamanhos pré definodos para manejo separado na cultura agrária-, para controlar o plantio, irrigação e colheita. A temperatura local engloba toda a área da fazenda, mas a irrigação tem de ser tratada de forma individual, portanto há um sensor de umidade de solo para cada talhão.

A maioria dos sinais gerados pelos sensores são analógicos, e para um sistema digital microcontrolado é necessário converte-lo. A Raspberry não dispõe de um

conversor A/D embutido, portanto utilizou-se o módulo conversor YL-40 que contém o CI PCF8951 da Philips, disposto na Figura 20.

Figura 20 – **YL-40 conversor A/D com PCF-8951**

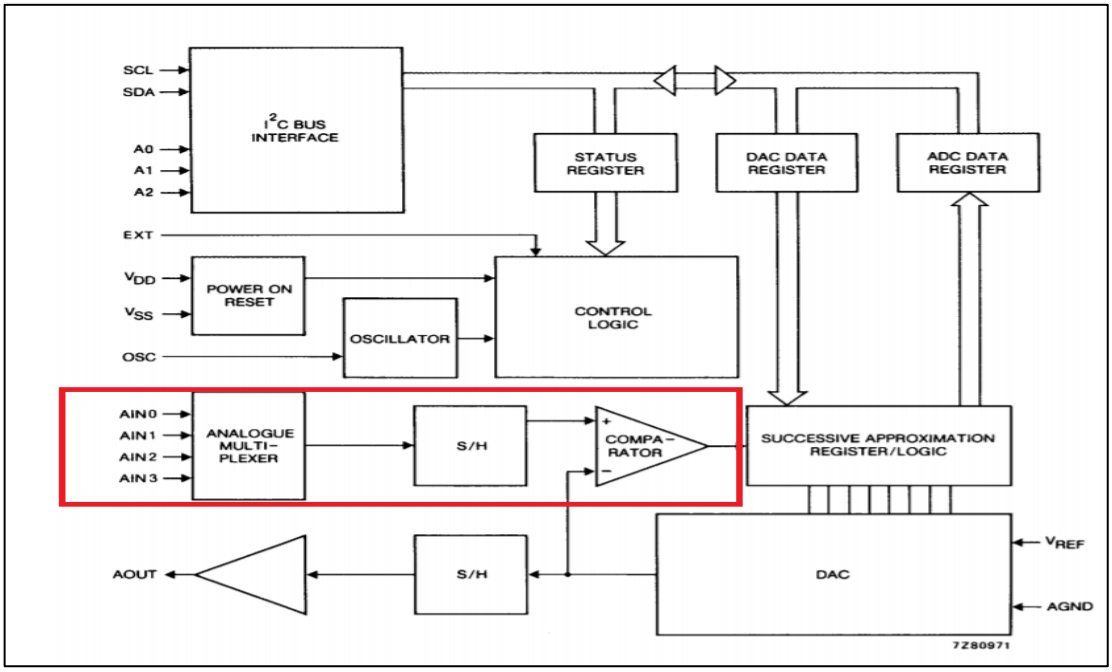


Fonte: Autoria Própria.

Este conversor tem quatro canais de entrada e um de saída analógica. Ele usa comunicação serial I²C para interface com microcontrolador. O módulo usa o princípio comparador de tensão para conversão A/D, e a saída tem resolução de 8 bits. A placa do conversor tem alguns sensores – LDR, NTC e Potenciômetro -, embutidos e habilitáveis através de jumpers. Caso o usuário faça uso destes sensores, ele inviabiliza o pino de entrada analógica da placa, uma vez que só é possível conectar uma entrada das quatro existentes no CI conversor.

A Figura 21 mostra o esquema em blocos do PCF8591, destacando os blocos relacionados ao A/D.

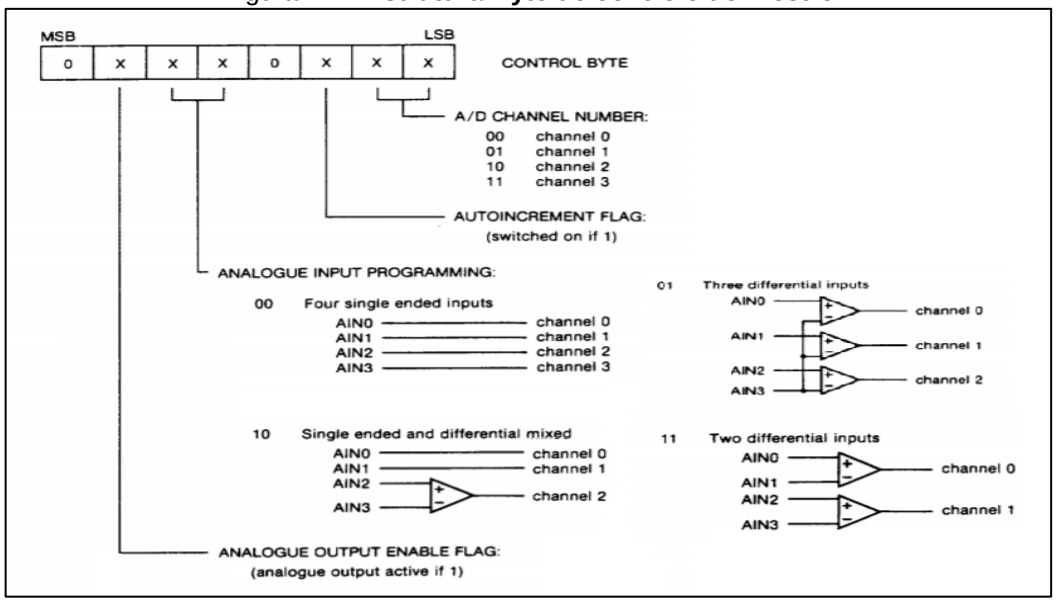
Figura 21 – **Blocos internos PCF-8591.**



Fonte: (PHILIPS, 1998)

Os sinais analógicos gerados pelos sensores de luminosidade, umidade do solo e temperatura entram nos canais de entrada AIN0, AIN1, AIN2 e AIN3. Após a conversão o resultado fica armazenado na memória/registrador, e quando solicitado pelo mestre na comunicação serial, o módulo manda os dados pelo barramento SDA. A comunicação é feita através de um Byte de controle. A Figura 22 mostra a configuração e solicitação que devem ser enviados.

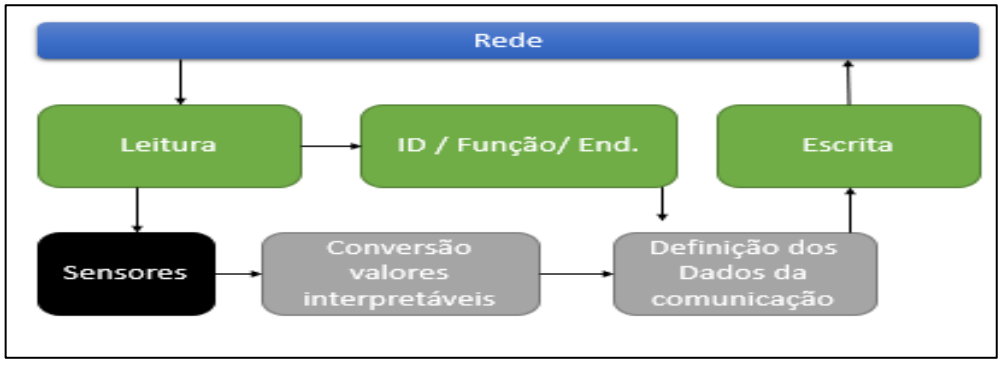
Figura 22 – **Estrutura Byte de controle do mestre**



Fonte: (PHILIPS, 1998)

O *software* da estação recebe o pacote pela rede, faz a aquisição dos valores dos sensores e comunicar ao mestre o valor solicitado. Sendo assim, a Figura 23 mostra o diagrama de blocos da lógica.

Figura 23 – **Fluxo de funcionamento do algoritmo da estação 1.**



Fonte: Autoria Própria.

No fluxograma é possível ver que a estação recebe o pacote da rede e faz a leitura. Então o programa verifica o id do pacote, se for o da estação, ela vai verificar qual é a função e o endereço que deve ser executado, caso contrário o pacote é descartado. Em seguida é feita a aquisição dos valores de tensão dos sensores. No bloco “Conversão de valores interpretáveis” é feita a conversão de tensão para a unidade de medida relacionada ao sensor, por exemplo o NTC, sendo que a tensão passa ser informada em graus Celsius. No bloco “Definição dos Dados da comunicação”, com base no endereço e função recebida na comunicação, o *software* identifica e escreve no pacote de resposta, específicamente no campo de dados, o valor de qual sensor deve retornar ao gestor.

As funções e endereços permissíveis da estação 1 estão listados na Quadro 2.

Quadro 2 – **Instruções admissíveis recebidas pela comunicação.**

	Nome da Função	Id	Função	End. High	End. Low
Reservado	Interferência Magnética	2	1	-	-
Sensores	Temperatura		2	0	1
	Umidade solo Talhão 1			0	2
	Umidade solo Talhão 2			0	3
	Luminosidade			0	4
Indicadores	Temperatura		3	0	1
	Luminosidade			0	2

Fonte: Autoria Própria.

Por exemplo, quando a estação recebe um pacote com id =2, função =1, ela executa a função de interferência magnética. Esta função consiste em responder ao gestor o *frame:* 02|1|AA|AA|FF (ID | Função | End hi lo | dados | CRC-16). O propósito desta função é o gestor perceber se a comunicação está acontecendo sem interferências, se os bits do pacote não estão sendo corrompidos ou alterados. Por isso o campo dados e endereço recebem o valor AA em hexadecimal, que corresponde a sequência binária 10101010, ou seja, se alguma coisa interferir no barramento da comunicação, irá alterar o nível lógico do bit e, consequentemente, o valor hexadecimal.

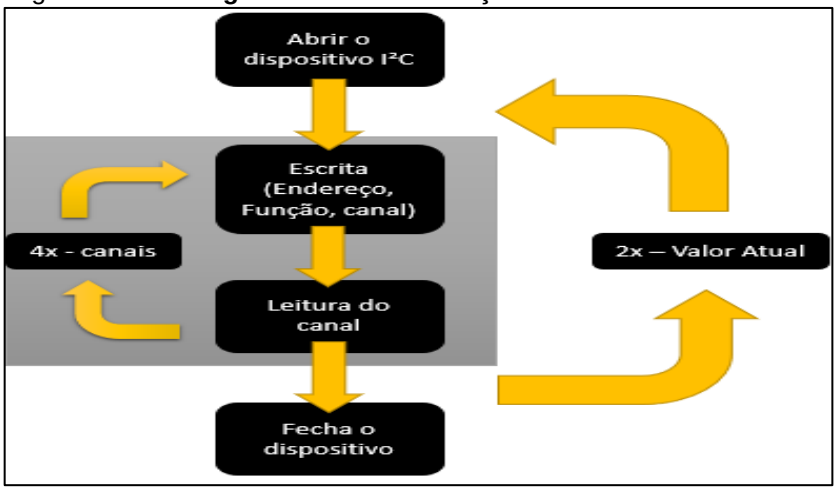
A função 2 relacionada ao endereço, seleciona o sensor cujo o valor será enviado ao gestor. O *frame* de resposta sempre contém o id da estação que manda o pacote, a função e o endereço que foram recebidos e executados.

Foi definido um *range* abaixo do ideal, o ideal e acima; para temperatura e luminosidade. A função “indicadores” diz ao gestor em qual ponto e unidade mensurada se encontra.

Para formar o *frame* de resposta, a estação atribui seu id, a função, o endereço da função que o gestor solicitou, além do valor convertido do sensor, que ocupa o quadro de dados.

Os valores dos sensores vêm do conversor A/D, este que se comunica com a Raspberry por meio de comunicação I²C. A figura 24 mostra a estrutura interna do bloco de sensores, com fluxograma de comunicação e aquisição dos valores dos canais.

Figura 24 – **Fluxograma da Comunicação I²C com conversor A/D.**



Fonte: Autoria Própria.

O código abre o dispositivo I²C por atribuição do endereço de acesso do sistema operacional. Em cada escrita e leitura, o microcontrolador passa o endereço do conversor, a função e o valor de um canal, por isso é necessária uma interação para cada canal. É importante destacar que o valor recebido da comunicação corresponde ao que estava armazenado no registrador, portanto este valor vem de um instante anterior a solicitação de leitura. Para saber o atual valor do conversor, são necessárias duas interações. Na estrutura da figura 24, este procedimento está indicado “2x – Valor Atual”.

O valor recebido do conversor é um inteiro de 0 a 255 (8 bits), que representa a tensão associada ao sensor. Para interpretação do usuário é necessário manipular para transformá-lo em uma grandeza compreensível. Para o NTC esse valor deve ser expresso em graus Celsius, para isto é utilizada a equação de Steinhart-Hart e seus

coeficientes, que relaciona resistência e temperatura de um semicondutor (LAUDARES, et al.2014).

$$T = \frac{1}{a + b * \ln(R_{ntc}) + c * ln^3(R_{ntc})} \quad [2]$$

Sendo:

- T é temperatura em Kelvins
- a, b e c coeficientes de Steinhart-Hart
- R_{ntc} Resistência do NTC

Os valores dos coeficientes são disponibilizados pelo fabricante do NTC 10k e para este trabalho foram adotados (EPCOS, 2006):

- a = 0,0011303
- b = 0,0002339
- c = 0,00000008863

A Figura 25 mostra a parte do código que pega o valor do conversor e o processo de transforma-lo em graus Celsius.

Figura 25 – **Conversor Valor A/D para temperatura**.

```
double vr, rntc,logntc,b;
float c;
#define resistor 10000

vr = ((valor[3]*3.3)/255);
rntc = (((5*resistor) - (resistor*vr))/vr);  Rntc

logntc = log(rntc);
b = (0.0002339*logntc); b
c = pow (logntc, 3);
c = c*0.00000008863;  c

  printf ("valor [3]: %d \n",valor[3]);
  printf ("Vr = %.2f Rntc = %.2f Log rntc = %f b=%f c=%f\n", vr, rntc, logntc,b,c);
                  a
   temp = (1/(0.0011303+b+c))-273.15;
    printf ("Temperatura:  %.2f\n",temp);
```

Fonte: Autoria Própria

Para utilizar a fórmula de Steinhart-Hart, é necessário saber o valor da resistência do NTC. O sensor está associado a um divisor de tensão com um resistor de 10 kΩ e alimentação de 5 V. A informação que o sistema tem, é a tensão de saída

do divisor, mas conhecendo os valores da resistência fixa e da alimentação, é possível chegar na resistência do NTC com a seguinte expressão:

$$V_o = \frac{V_{cc} * R_{ntc}}{R_{ntc} + R_1} \quad [3]$$

Sendo:

- V_{cc} tensão de alimentação do divisor (5v).
- V_0 tensão de saída do divisor e entrada do conversor A/D.
- R_{ntc} Resistencia do NTC.
- R_1 resistor fixo de 10 kΩ.

Foram utilizados dois sensores de umidade de solo, cada um correlacionado à um talhão, ocupando um canal do conversor. O sensor é tratado como informação analógica que é capaz de definir o quão úmido o solo está.

Mediu-se experimentalmente o valor de tensão do solo totalmente seco e úmido, definindo assim uma escala de operação. Após a conversão do valor do canal, o *software* encontra este ponto dentro da escala, e traduz ao usuário qual é a percentual de umidade no solo. A Figura 26 mostra a lógica programada que faz a conversão da medição da umidade em porcentagem.

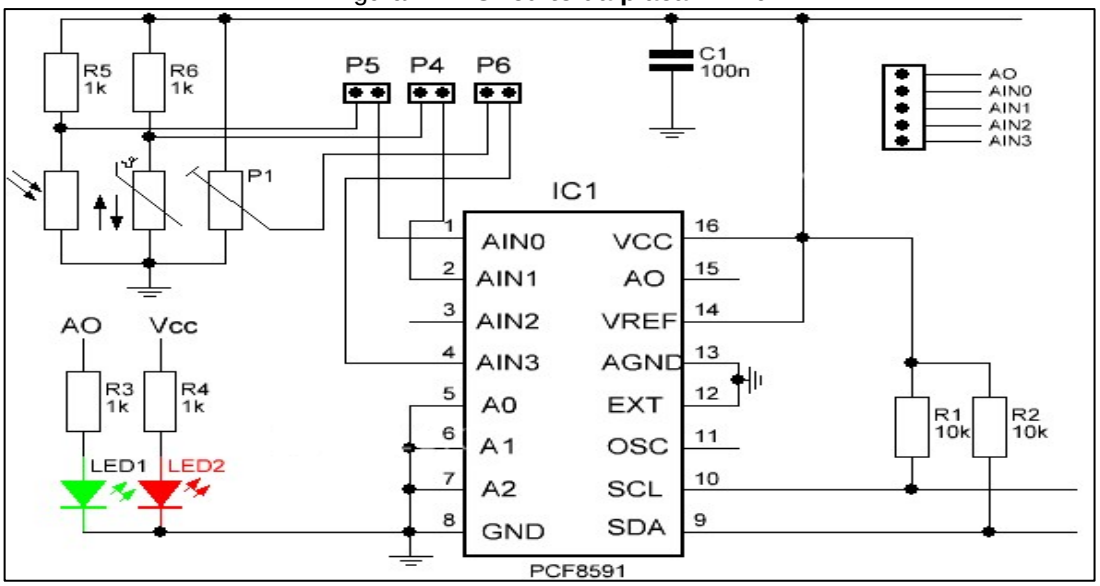
Figura 26 – **Conversão da umidade em porcentagem.**

```
humidadesolo1 = 255 - valor[1];
humidadesolo2 = 255 - valor[2];
pcenthumidadesolo1 = ((humidadesolo1/115)*100);//140 -> 100% 255 - 140 = 115
pcenthumidadesolo2 = ((humidadesolo2/115)*100);
printf ("%.2f %% de humidade no solo 1\n", pcenthumidadesolo1);
printf ("%.2f %% de humidade no solo 2\n", pcenthumidadesolo2);
```

Fonte: Autoria Própria.

Para monitorar a incidência de luz, foi utilizado o LDR disposto na placa YL-40. Desta forma não deve ser utilizada a porta AIN0 por segurança do *hardware*. Para o PCF8591 nada se altera com relação ao byte de controle, pois o endereço de leitura ainda é referente ao canal zero. A Figura 27 mostra o circuito da placa.

Figura 27 – **Circuito da placa YL-40**



Fonte: (WEIKEDZ, 2017) Adaptado

O sensor varia a resistência conforme a incidência da luz, portanto para obter a quantidade de LUX, utilizou-se a mesma lógica da temperatura no NTC, sendo que o LDR faz o divisor de tensão com resistor de 1 KΩ, R5 da Figura 27. É possível calibrar o senhor via código, que possibilita o uso do sensor para difenretes situações de incidência luminosa. A Figura 28 mostra a parte de conversão de resistência em LUX.

Figura 28 – **Conversão da resistência no LDR em LUX**

```
double vldr, rldr,lux;
vldr = ((valor[0]*3.3)/255);
rldr = (((5*1000) - (1000*vldr))/vldr);
printf ("Valor [0]: %d\nVldr: %.2f Rldr:%.2f\n",valor[0],vldr,rldr);
#define calibraldr 200
lux = (calibraldr * rldr)/1000;

printf("\n Lux: %.2f\n", lux);
```

Fonte: Autoria Própria.

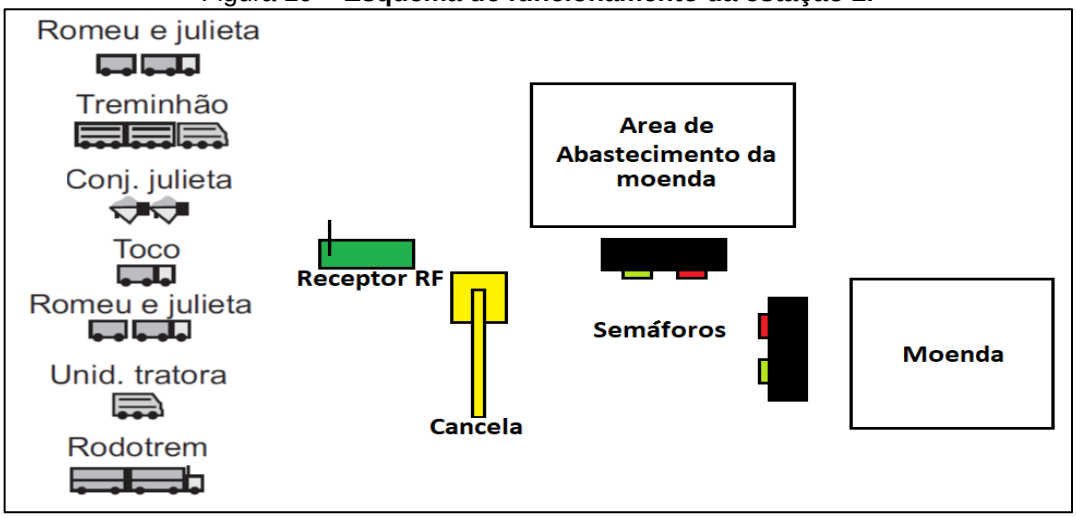
3.2. ESTAÇÃO 2 – CONTROLE DA FILA DE ALIMENTAÇÃO DA MOENDA

A importância de alimentar corretamente as moendas, controlar e organizar as filas de descarregamento é abordada por Iannoni e Morabito (2002). Além da descarga dos caminhões direto no abastecimento da moenda, ainda existe uma área para despejo, para otimizar o descarga da cana e controlar o nível de alimentação.

A principal função da estação é a logística de descarga, para sempre manter as moendas alimentadas dentro do ideal, além de que um caminhão parado, esperando para fazer a descarga, é um caminhão ocioso que não contribui para o fluxo de produção, contabilizando prejuízo em temporal e financeiro.

Para esse processo a estação deve saber como está a alimentação da moenda, para decidir se deve encaminhar a descarga direta ou para uma zona de reabastecimento controlado. É necessário identificar qual caminhão é o primeiro da fila para saber quanto de cana está chegando para o sistema manejar. O esquema de orientação e funcionamento da estação está apresentado na Figura 29.

Figura 29 – **Esquema de funcionamento da estação 2.**

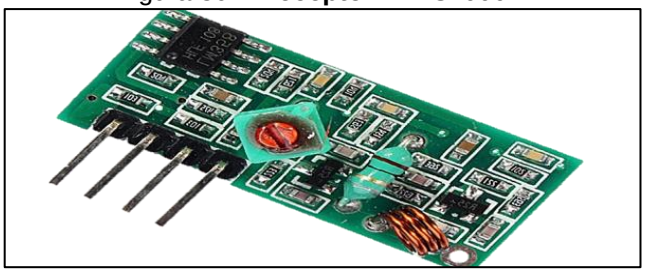


Fonte: Autoria Própria.

O caminhão chega na cancela, passa sua identificação que é recebida pelo sistema no Receptor RF, a cancela se abre e o semáforo indica o caminho a ser seguido.

Os tipos de caminhão destacados na figura 29, estão relacionados a capacidade de carga. Além da capacidade, pode haver mais de um caminhão do mesmo tipo. Para identificar o caminhão é necessário atribuir um rótulo numérico que o sistema seja capaz de identificar. O sensor fs1000A, que é receptor de radiofrequência – faixa de operação 433 MHz-, recebe o sinal eletromagnético que contém a identificação. A Figura 30 mostra o receptor.

Figura 30 – **Receptor RF fs1000A**



Fonte: Autoria Própria.

O sinal de saída do módulo é uma sequência de bits altos e baixos que varia a largura dos pulsos conforme a modulação do sinal recebido. O processo de demodulação ou extração da informação, é feito em software. Para enviar o id usou-se o controle transmissor RF modelo 2005-TXCGABR fabricado pela Gmax, disposto na figura 31, que transmite um id decodificado em 433 MHz.

Figura 31 – **Controle RF 2005-TXCGABR - Gmax**

Fonte: (Gmax, 2005)

Este tipo de transmissor manda uma identidade numérica modulada em uma onda portadora de 433 Mhz. Cada controle tem seu próprio valor de id. Em geral controles deste seguimento apresentam o mesmo tamanho do pacote com a informação modulada e a mesma quantidade de número em seu id.

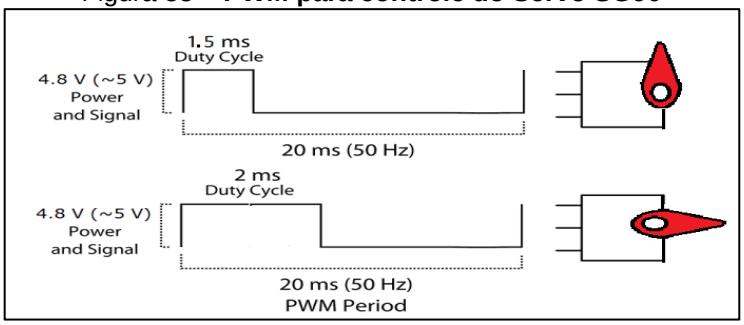
Para a cancela, foi utilizado o servo-motor SG90 fabricado pela Tower Pro, disponível na Figura 32.

Figura 32 – **Servo Motor SG90**

Fonte: (Tower Pro, 2017)

A posição do eixo do motor é controlada por um sinal modulado em pulso, PWM (*Pulse-Width-Modulation*). São necessárias duas posições de eixo, a primária tem orientação horizontal paralela ao chão e a secundária perpendicular. A Figura 33 mostra o ângulo relacionado ao pulso de controle (Tower Pro, 2017).

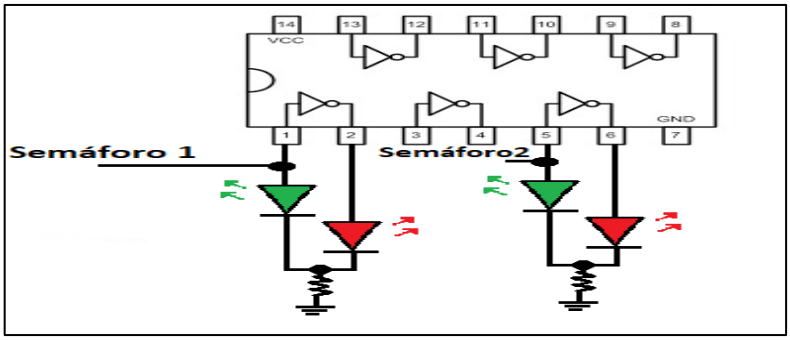
Figura 33 – **PWM para controle do Servo SG90**



Fonte: Autoria Própria.

Os semáforos foram feitos com Led's (*light emissor diode*). São dois semáforos composto de duas luzes cada. Seriam necessários 4 sinais para controlar as luzes individualmente. Para otimizar este controle, foi utilizada a lógica com porta lógica NOT com o circuito integrado 74HC04. O esquema do controle de semáforo está na Figura 34, onde "Semáforo 1" e "Semáforo 2" são os sinais de controle que podem assumir nível lógico alto ou baixo.

Figura 34 - **Controle Semáforo**



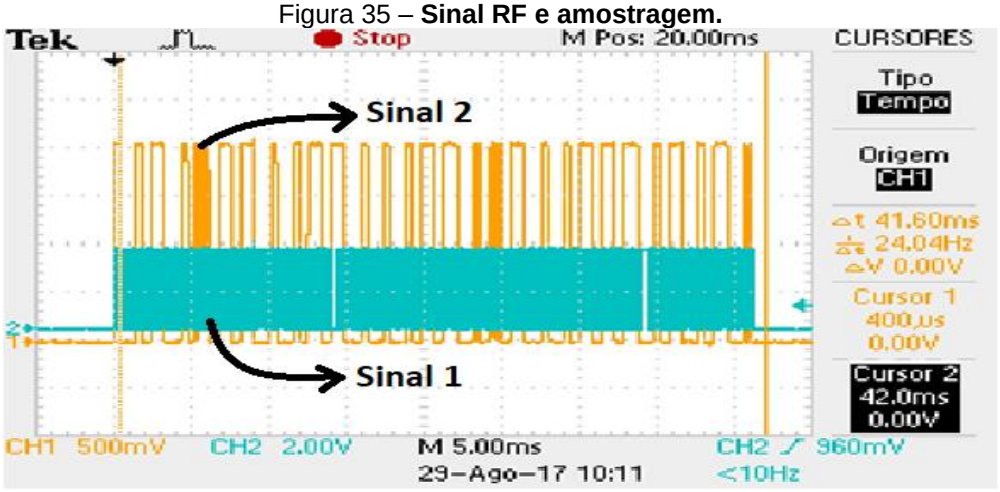
Fonte: Autoria Própria.

A estação roda seu conjunto de rotinas – recebe pacote do rf, valida o pacote, identifica o caminhão, acionar cancela e semáforo-, para manter o funcionamento da estação, além de receber instruções da rede.

O sensor fica a todo tempo recebendo sinais eletromagnéticos e convertendo em informação na saída. A função de receber e validar o pacote do RF, serve para

dizer quando é realmente um pacote válido ou se não é apenas um sinal oriundo de ruído eletromagnético que chegou do sensor. Se o pacote é válido, o programa faz a leitura com uma amostragem de 240 pontos do sinal em um intervalo de 42 milissegundos, para criar uma versão digitalizada do sinal. A Figura 35 mostra o sinal do sensor em “Sinal 2”e em “sinal 1” os momentos de amostragem.

Figura 35 – **Sinal RF e amostragem.**

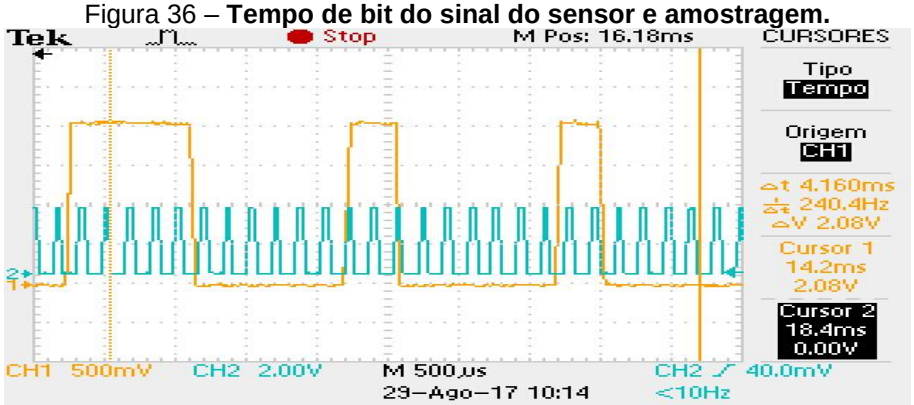


Fonte: Autoria Própria.

Cada pulso do sinal 1 é uma leitura do sistema- na figura, é possível ver uma grande sequancia de pulso em um tempo curto-, identificando se o sinal do sensor (sinal 2) está em nível alto ou baixo. A amostragem cobre todo o pacote, sem perda de informação válida.

O sinal é modulado em frequência, então a informação contida muda o tempo de bit alto e o tempo de bit baixo. A Figura 36 mostra um pedaço da onda do sinal que contem variações do tempo de bit alto e baixo em laranja, e em azul a amostragem.

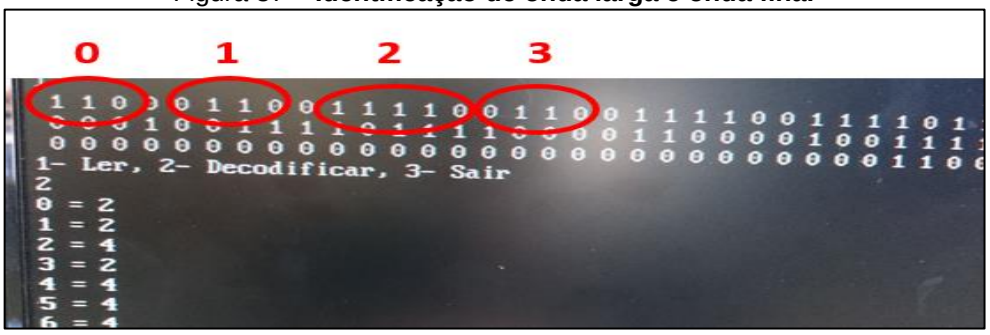
Figura 36 – **Tempo de bit do sinal do sensor e amostragem.**



Fonte: Autoria Própria.

Para desenvolvimento da lógica de decodificação, usou-se dois tipos de informação do sinal, onda mais larga e onda estreita em relação a largura da onda com nível logíco alto. A onda amostrada na Figura 36 tem o seguinte valor: 00111110000001110000001100000, portanto aos olhos do sistema, uma sequência de 1 maior representa onda larga e sequencia de 1 menor, onda estreita. A Figura 37 mostra o processo descrito do começo do pacote da imagem 36.

Figura 37 – **Identificação de onda larga e onda fina.**



Fonte: Autoria Própria.

Para identificar as ondas criou-se uma margem, pois as larguras delas variam. Para onda estreita, o sistema entende que onda fina é uma sequência entre 2 à 4 1's e para onda larga de 5 a 7. Um único 1 solitário pode ter sido gerado por uma interferência.

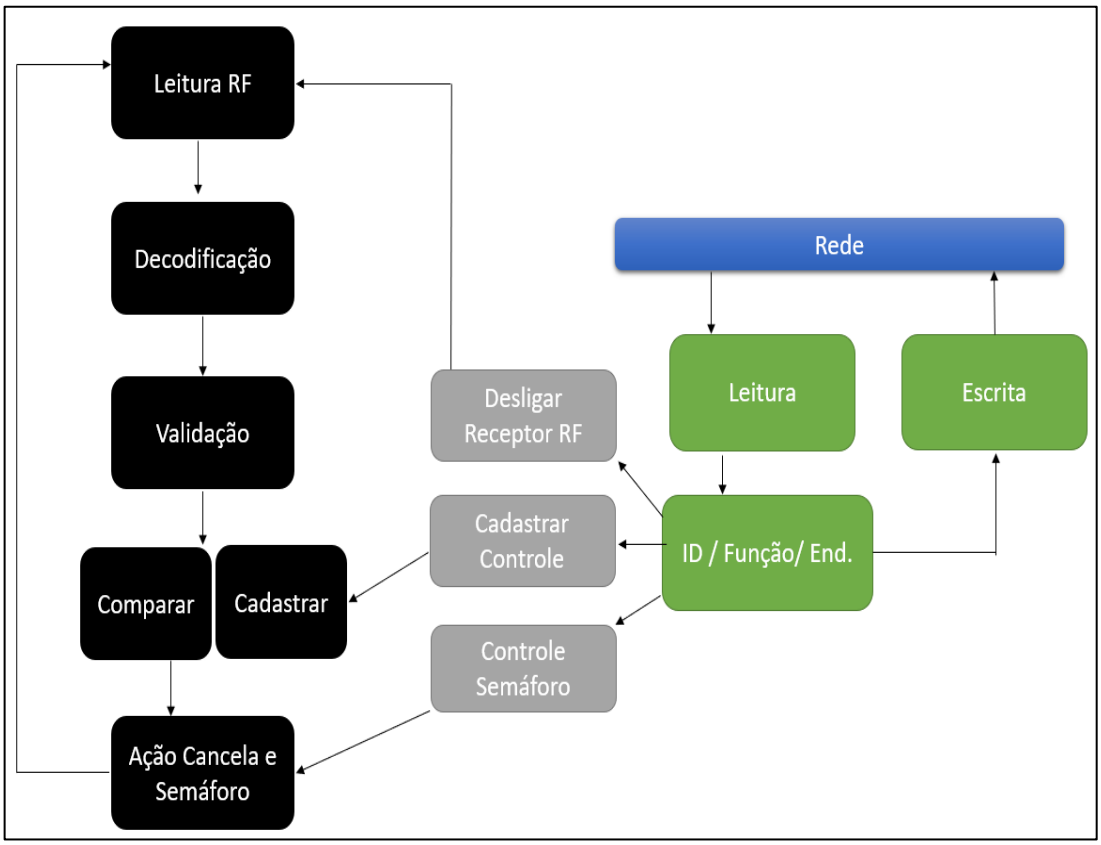
Depois de identificar as ondas, atribui-se um valor numérico para elas, onda fina vale 1 e onda larga 2, sendo assim o exemplo destacado anteriormente (00111110000001110000001100000) teria o id 211.

Esse id é um vetor com 29 valores, de onda larga ou estreita. Para compartilha este numero de identidade dentro da rede, seriam necessários o envio de vários pacotes contando dois valores do vetor. Para os 29 valores, seriam necessários 15 frames enviados na rede. Para transformar em um único valor, em vez de 29 valores do vetor, fez o deslocamento binário. Para logica binária a onde onda larga vale 1 e onda estreita 0. O algoritmo compara o tipo da onda e a posição dela dentro do vetor. A posição diz se o bit é mais ou menos significativo. Por exemplo, o vetor x (1,2,1,1,1,2,2,2,1,2,1) assume o valor binário 01000111010, que representa 570 em decimal. Desta forma, um vetor com 11 posições virou um valor único inteiro.

Na estação ficam cadastrados os id's de todos caminhões junto a sua capacidade de carga. Sendo assim, quando chega um pacote do RF, a estação faz a

comparação da identidade com todas armazenadas em seu banco de dados, e o que tiver maior compatibilidade é a identificada. Sendo assim, a estação pode definir o caminho que o caminhão deve seguir, controlando o volume de cana que deve ser mantido na alimentação direta da moenda. A Figura 38 mostra a estrutura e blocos das rotinas do *software.*

Figura 38 – **Estrutura em blocos do software da estação 3.**



Fonte: Autoria Própria.

A comunicação com o gestor serve para a estação cadastrar um novo controle, controlar diretamente os semáforos e desligar o receptor RF. A estação responde com “confirmando” quando finaliza o cadastro de um novo controle enviando o id do controle, informando que foi ligado/desligado o RF ou liberada/bloqueada a fila diretamente no semáforo. Sendo assim, o Quadro 3 mostra as funções e endereços validos na comunicação.

Quadro 3 – **Funções e Endereços validos da comunicação na estação 3.**

	Nome da Função	Id	Função	End. High	End. Low
Reservado	Interferência Magnética	3	1	-	-
Cadastrar Controle	Controle 1		3	0	1
	Controle 2			0	2
	Controle 3			0	3
Receptor RF	Desligar		4	0	1
	Ligar			0	2
Semáforos	Proibir todas		5	0	0
	Liberar principal			0	1
	Liberar secundaria			0	2

Fonte: Autoria Própria.

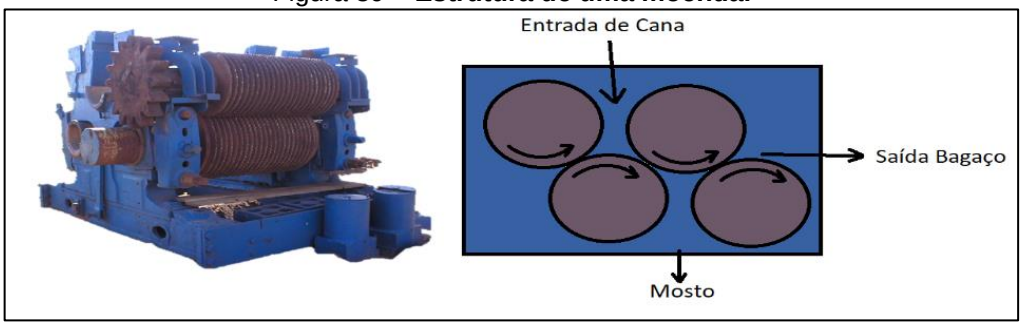
O gestor pode programar manutenções preventivas ou reparos emergenciais, sendo assim, ele pode desativar o receptor RF ou proibir todas as filas para interromper o fluxo de chegada de cana.

3.3 ESTAÇÃO 3 – MONITORAMENTO DA MOENDA

A estação 4 monitora a moenda através da vibração mecânica, sendo sensível ao nível de alimentação - quanto de cana entra em um determinado tempo-, e a integridade física de algumas peças.

A moenda é foco de estudo desta estação. Então primeiro é necessário compreensão de seu funcionamento para conseguir correlacionar a vibração com os objetos de estudo. Sendo assim, a Figura 39 mostra a estrutura de uma moenda.

Figura 39 – **Estrutura de uma moenda.**



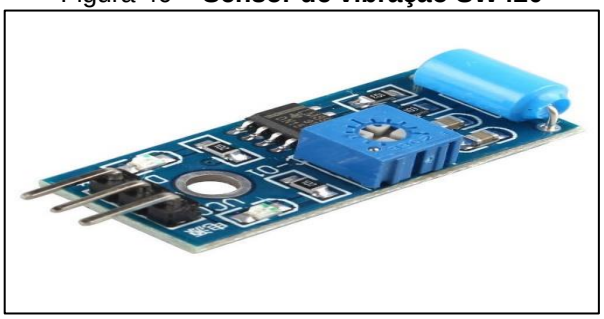
Fonte: Autoria Própria

Quando uma certa quantidade de cana passa pelos rolos da moenda, elas provocam um afastamento momentâneo dos rolos, sendo que quantidades maiores, provocam maior afastamento. O movimento de afastamento variado provoca vibração. A alimentação não é uniforme. Além da quantidade de cana, a matéria prima varia de tamanho e diâmetro. Estes fatores também contribuem para a vibração. Com o tempo as peças vão ficando gastas e com folga, o que provoca um padrão diferente de vibração. Por exemplo, com os dentes desgastados do cilindro, há mais espaço para a cana, provocando uma vibração menor que a usual. A variação do volume da quantidade de cana que entra na moenda e o desgaste físico dos componentes, é diretamente proporcional a vibração.

Portanto a estação consiste em reconhecimento de padrões. Ela aprende de uma forma supervisionada, por reforço do usuário que manipula o programa, o que significa cada padrão.

Para identificar a vibração da moenda foi utilizado o sensor de vibração SW420, apresentado na Figura 40.

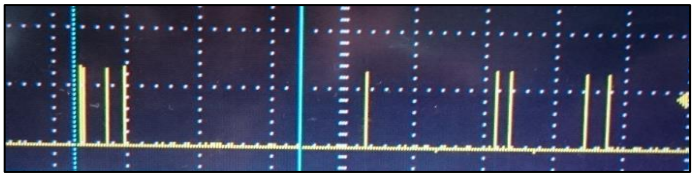
Figura 40 – **Sensor de vibração SW420**



Fonte: (WEIKEDZ, 2017)

O transdutor trabalha com a descontinuidade da saída provocado por um movimento mecânico. No sensor ele é conectado ao comparador de tensão e a saída é normalmente baixa, quando há vibração tem-se um pulso na saída, quanto maior a vibração, mais pulsos seguidos. A Figura 40 mostra a saída do sensor.

Figura 40 – **Sinal de saída do sensor de vibração.**



Fonte: Autoria Própria.

O comportamento do sinal de saída é contínuo e aleatório, diferente do sinal de RF, que vem com um pacote definido com a informação contida. Então, para identificar foram feitas dez mil amostras do sinal. A leitura consiste em 1, quando a saída do sensor é nível alto ou zero, caso contrário. O comportamento do sensor identificado é que o nível de vibração aumenta o número de pulsos e, consequentemente, o número de 1 no sinal amostrado.

Para gravar este comportamento/padrão no sistema, é somado o número de 1 dentro das 10 mil amostras. Sendo que a vibração mais intensa na moenda, leva a somatória a valores maiores. Correlacionando vibração à uma situação física, como a alimentação das moendas, pouca cana entrando, vibração baixa e, consequentemente, número resultante final baixo. Quando se tem muita cana entrando a vibração alta e o número final também.

Como há vários comportamentos vibratórios de uma moenda, foi definido uma faixa para cada padrão. Um padrão é definido pela *range* de vibração. Para criar esta faixa de valores, é feita a média de 10 leituras de 10 mil amostras, depois é somado e subtraído 800 no valor médio para definir o limite superior e inferior da *range*. Sendo assim, quando o sistema faz uma nova leitura de 10 mil amostras, ele compara para ver se esta nova leitura esta dentro da range padronizada.

Para este trabalho estudou-se só os padrões de alimentação, pois entender os comportamentos da vibração para falhas mecânicas podem ser um pouco mais complexos, e difícil garantir a identificação com apenas um sensor.

O *software* da estação grava três padrões da alimentação, abaixo do nível bom, no nível e acima. Quando o gestor pergunta como que está o nível de alimentação, o programa faz uma amostra, e localiza em qual *range* o valor se encontra. Este processo permite, por exemplo, a estação de controle de fila identificar se o nível de cana está baixo e manda o caminhão direto para o abastecimento da moenda.

A comunicação com a rede diz a estação para gravar um padrão, ou informar ao gestor a *range* dos padrões, superior e inferior. Sendo assim, os valores de função e endereço validos na comunicação estão na Quadro 4.

Quadro 4 – **Funções e endereços validos na comunicação.**

	Nome da Função	Id	Função	End. High	End. Low
Reservado	Interferência Magnética	4	1	-	-
Status	Verifica em qual range que está.		2	-	-
Cadastrar Padrão	Padrão 1 abaixo		3	0	1
	Padrão 2 ideal.			0	2
	Padrão 3 acima.			0	3
Range	Padrão 1		4	0	1
	Padrão 2			0	2
	Padrão 3			0	3

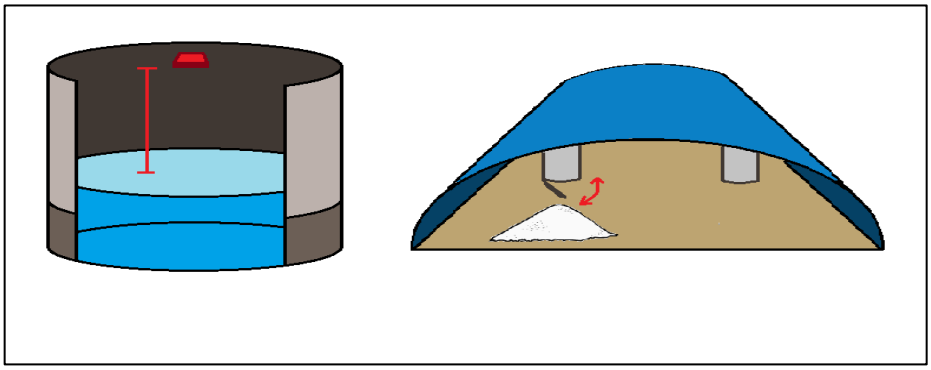
Fonte: Autoria Própria.

A função "*Range"* passa ao gestor os limites superior e inferior do padrão que ele solicitou.

3.4 ESTAÇÃO 4 – ARMAZENAMENTO E ESTOCAGEM

A estação faz o monitoramento dos tanques de armazenamento de etanol e dos barracões de armazenamento de açúcar. Estas informações permitem ao gestor ver o fluxo de entrada da produção e a saída. A figura 41 mostra o funcionamento e disposição dos sensores para planejamento da lógica de funcionamento.

Figura 41 – **Disposição dos sensores.**



Fonte: Autoria Própria.

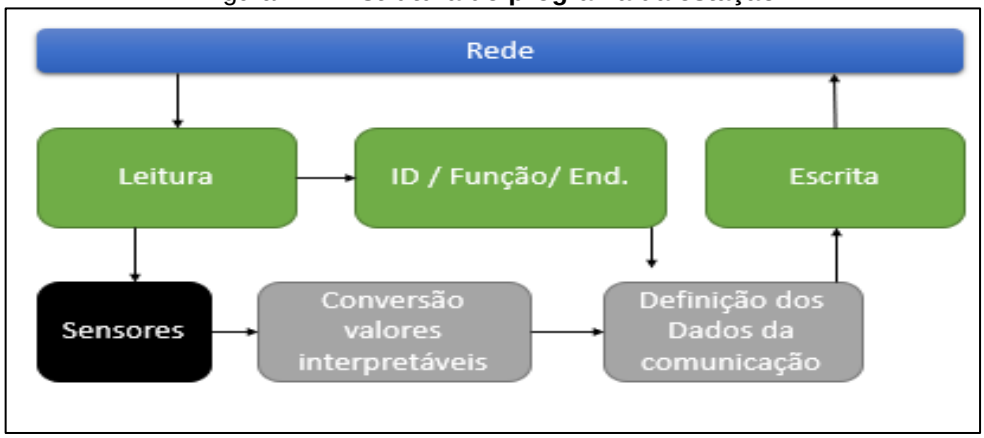
Para monitorar o etanol, foi utilizado o sensor ultrassônico HC-SR04 colocado no ponto mais alto do tanque. Quando há entrada de etanol a distância mensurada

pelo sensor diminui, e quando há saída, aumenta. Sabendo o tamanho do tanque totalmente vazio, é possível calcular a capacidade utilizada e disponível com a distância do sensor.

O açúcar chega ao armazém por tubulações, então para monitorar se está chegando produto, foi projetado um esquema de tampa, quando não está escoando açúcar pela tubulação, a tampa está fechada. O escoamento de açúcar provaca a abertura da tampa, sendo assim, o sistema identifica o estado da tampa para correlacionar com o escoamento. Quando a tampa está fechada, ela fecha uma chave e conduz o sinal elétrico, com está logica, o sistema consegue marcar o tempo de tampa aberta, que é consequentemente o tempo de escoamento de produto. Conhecendo o diâmetro do cano, é possível estimar o volume de produto que chega ao armazém. Como há ar junto ao açúcar que chega, o valor informado pelo sistema é apenas uma estimativa. Dado o volume de produção, na maior parte do tempo de escoamento, a tubulação está completa com produto e não ar, sendo assim, a estimativa é valida.

A estação responde ao gestor o nível do etanol, porcentagem da capacidade do tanque utilizada, se está escoando açúcar e o tempo do último escoamento. Sendo assim, o algoritmo da estação responde a solicitações da rede, sem rotina definida. A figura 42 mostra a estrutura do programa.

Figura 42 – **Estrutura do programa da estação**.



Fonte: Autoria Própria

A conversão de valores é a transformação da distância em informação interpretáveis pelo gestor, por exemplo, a porcentagem do tanque utilizada e/ou

quanto de espaço vazio ainda tem. As funções e endereços validos da recepção do pacote da comunicação estão na tabela 5.

Tabela 5 – **Funções e endereço das funções da estação 4.**

	Nome da Função	Id	Função	End. High	End. Low
Reservado	Interferência Magnética	5	1	-	-
Tanque Etanol	Altura do Produto		2	0	1
	% utilizado			0	2
Escoamento Açúcar	Status de Escoamento		3	0	1
	Tempo último escoamento			0	2

Fonte: Autoria Própria.

4. RESULTADOS E DISCUSSÃO

A Raspberry mostrou exelente capacidade de processamento para todos os algoritmos. O programa que mais gasta tempo de máquina em processamento é o da estação 4, que na função de gravar um padrão roda 30 mil leituras Gpio, faz a soma dos bits 1 da leitura e faz a média para definir a range em menos de 5 segundos.

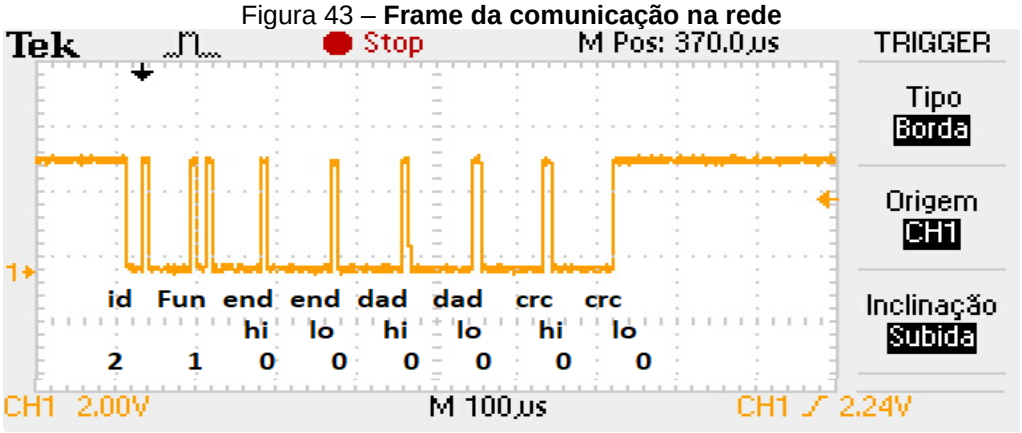
O sistema operacional embutido se mostrou um recurso valioso na facilidade de desenvolvimento do trabalho. Ele permite acessar os dispositivos e recursos da placa com definição de endereço do *device* no SO, e configurar o mesmo passando alguns parâmetros ou *flags.* Todos os algoritmos foram desenvolvidos em linguagem C e compilados na própria placa.

A contribuição que o sistema embarcado tem para o trabalho é o conceito chamado *plug-and-play,* que se refere à um sistema pronto que ao ser ligado já está apto para funcionar. Em outras palavras ele permite que o sistema desenvolvido funcione paralelamente ao sistema de controle que já está aplicado em uma usina, não interferindo nas rotinas do sistema e agregando valor as informações geradas ao gestor, desde que esteja operando com o protocolo *ModBus.*

A rede mostrou bom funcionamento para todos os *BaudRate* definidos para aplicação do trabalho. Não há necessidade de alta velocidade na transmissão da informação, pois o sistema é controlado por uma pessoa e ele não manda muitos pacotes em curto período de tempo, portanto usar uma taxa de transmissão mais alta como 115200 ou mais baixa como 9600 não é perceptível ao usuário.

A Figura 43 mostra um *frame* de comunicação na rede.

Figura 43 – **Frame da comunicação na rede**

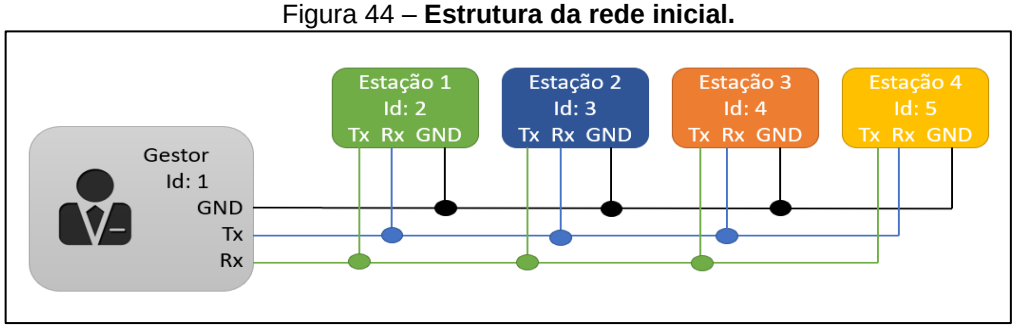


Fonte: Autoria Própria.

Quando o gestor manda o pacote na rede, todas estações o recebem, portanto, a extração das informações como id, função e endereço do pacote é importante para o funcionamento da rede.

A estrutura da rede inicialmente era em outra topologia, sem controle de acesso ao Rx do gestor. A figura 44 mostra a estrutura da rede inicial.

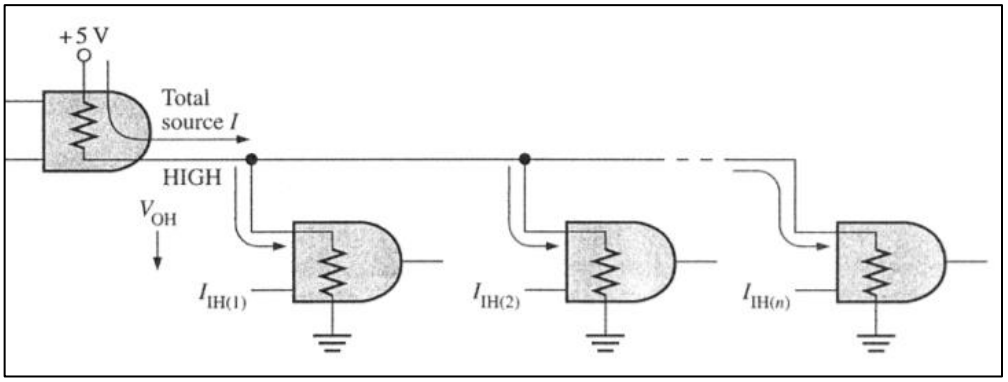
Figura 44 – **Estrutura da rede inicial.**



Fonte: Autoria Própria.

Durante o desenvolvimento do trabalho constatou-se que este modelo não funcionava, porque os transmissores das estações estão ligados em um mesmo barramento. Ligando os pinos direto em um barramento configura uma conexão TTL e quando há uma porta com sinal HIGH, ligada paralelamente às outras portas, ocorre o problema de redução de amplitude do sinal. A figura 45 mostra a conexão citada.

Figura 45 – **Conexão TTL com cargas paralelas.**



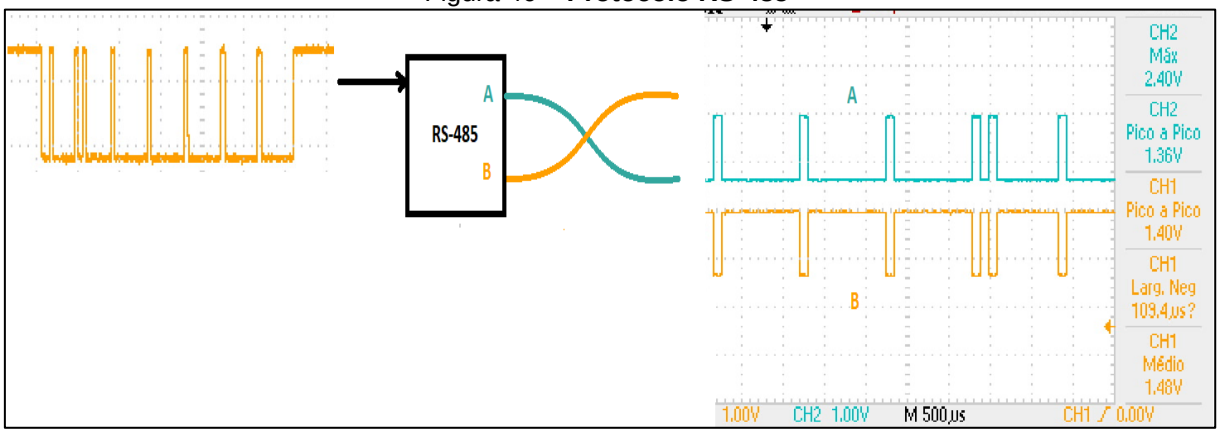
Fonte: (CASTRO, 2011)

A corrente I aumenta à medida que cresce o número de portas utilizadas como carga. Com aumento de corrente, aumenta a queda de resistência interna na porta que manda a informação e isto faz com que a tensão Voh não assuma valor baixo. A informação orientada no protocolo UART, faz o sinal de saída do pino mudar de alto

para baixo como apresentado na figura 43. Sendo assim, o problema de muitas portas paralelas não permite a comunicação.

Este trabalho resolveu esse problema permitindo apenas uma conexão por vez. Há protocolos que orientam a comunicação na camada física, como o RS-485, por exemplo. Neste protocolo o sinal de mensagem é duplicado e um invertido em relação ao outro. A figura 46 mostra o sinal e a topologia do protocolo.

Figura 46 – **Protocolo RS-485**



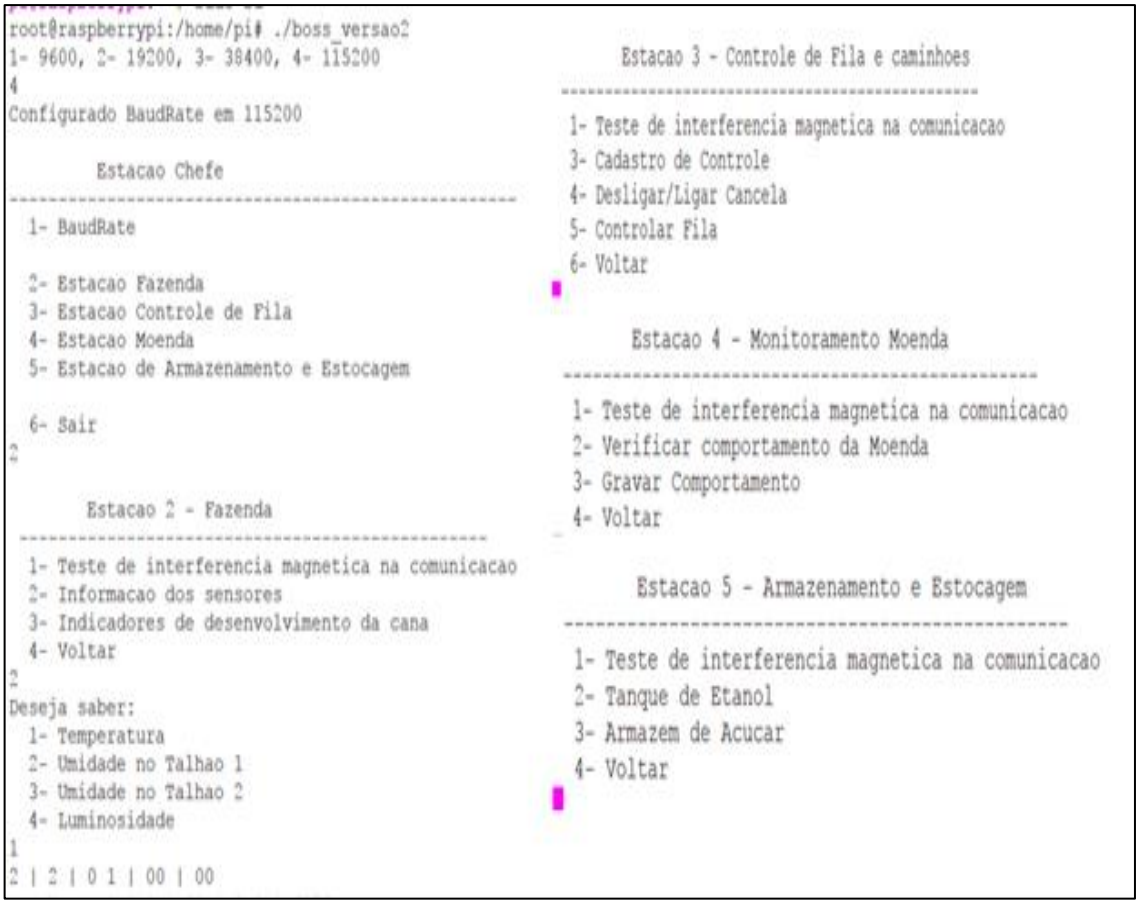
Fonte: Autoria Própria

Nessa configuração, são utilizados dois fios para o Tx e dois para o Rx. Os fios condutores, A e B, são conhecidos como par trançado. Essa configuração é para anular os ruídos eletromagnéticos que poderiam corromper a informação. Este método permite a disposição de todos elementos na rede como na topologia apresentada na figura 44, a diferença é que cada fio da figura seria na verdade um par trançado de cabos.

A alteração dos significados dos valores das funções originalmente estabelecidos pela Modcon para o protocolo Modbus não afetam o fato que outro sistema com o protocolo possa interagir com o sistema desenvolvido pelo trabalho.

O *software* do gestor é um menu de opções que gera os valores dos campos do *frame* a ser mandado na rede. A figura 47 mostra o menu do programa com as opções das estações.

Figura 47 – **Interface do programa do Gestor.**

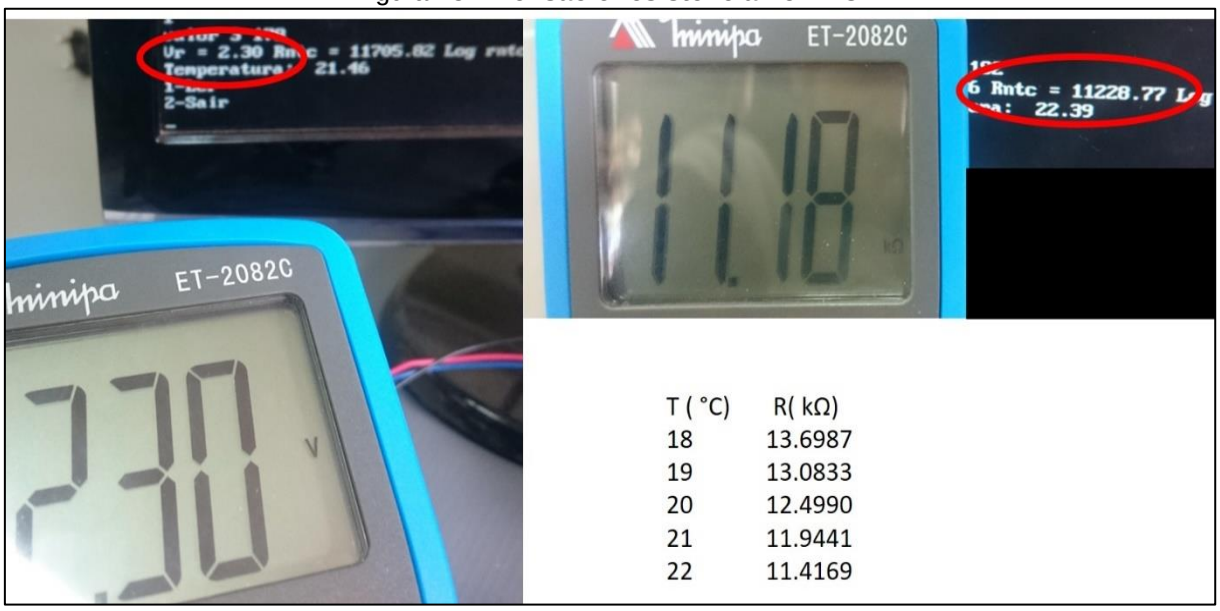


Fonte: Autoria Própria.

O menu leva a três níveis; o primeiro no qual o gestor escolhe qual estação quer solicitar informação, define o id. No segundo nível, onde contém as opções de cada estação, define a função. O último define o endereço para funções que tem mais de uma opção. Por exemplo, para verificar a temperatura, o gestor seleciona 2, depois 2 também no segundo menu e por fim 1. No fim da imagem é possível ver o pacote formado e enviado na rede.

Nas estações foi necessário um processo de validação do funcionamento dos sensores. Para validar a temperatura do sistema, verificou-se experimentalmente os valores de tensão e resistência dos componentes relacionados ao NTC e os valores gerados pelo sistema. A figura 48 mostra o processo.

Figura 48 – **Tensão e resistência no NTC**



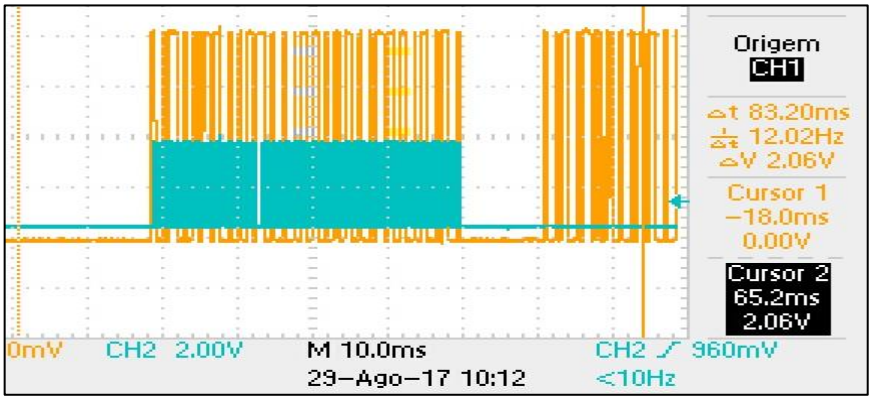
Fonte: Autoria Própria

A diferença do valor do Rntc deu um pouco diferente pois o resistor que compõe o divisor não é preciso, mas para este erro interferir no valor nesta faixa de temperatura, a diferença tem que estar de 400 a 600 ohms. A tabela completa referente aos valores de temperatura relacionada a resistência está na figura 4.

Para umidade do solo mediu-se os valores de tensão em cima do transdutor para o que foi considerado solo totalmente seco e totalmente úmido. Para totalmente seco, a tensão do transdutor é igual à alimentação VCC, 3.3 V, e para totalmente úmido, 2,14 V. O sistema dentro da *range* de 1,15 V determina a porcentagem de umidade.

Na estação que faz o controle de fila, a identificação do pacote RF é a parte mais importante. O algoritmo identifica onde começa e termina a informação, para minimizar os resultados errados. A figura 49 mostra a recepção de dois pacotes enviados pelo controle e a amostragem em azul, selecionando apenas um pacote completo.

Figura 49 – **Pacotes RF e amostragem com seleção do pacote.**



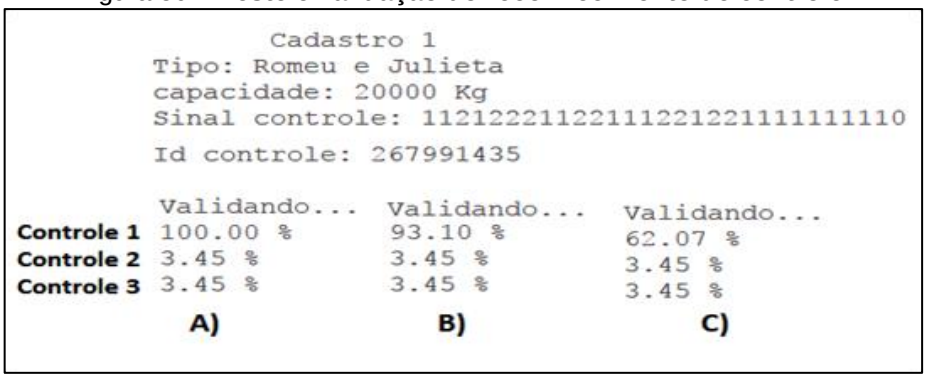
Fonte: Autoria Própria

A estação gera um valor inteiro do id para viabilizar a disposição desta informação na comunicação, pois se fosse transmitir o valor do vetor que contém o id, seriam necessárias 29 posições de dados no *frame* ou a transmissão de 16 frames. Um número inteiro é para a transmissão de um único pacote, mas o valor que o inteiro pode assumir é muito grande, 29 bits, e a comunicação com dois campos para dados chega no máximo à 16 bits. Para resolver este problema seria necessário aumentar o número de quadros de dados para 4.

Para comparar uma nova leitura do RF com os ids armazenados não se utilizou o valor inteiro, pois uma mínima interferência, que mudaria um único bit, poderia provocar um desvio muito grande do id. Por exemplo, se por interferência o bit mais significativo passou de 0 para 1, o valor dá um salto gigantesco no resultante. Por isso para comparar qual id que chegou pelo pacote usa-se todos os valores armazenados em um vetor. Se um valor estiver diferente, em qualquer posição do vetor, outros 28 estão certos, o que oferece uma chance de 96 % de ser o id. O sistema entende que se a chance for acima de 92%, é o id registrado.

Para distância pequena entre o receptor e o controle, até 10 cm, o nível de acerto é alto para diferentes modelos de controle. Quando a distância aumenta, o erro do sistema passa a errar. Isto acontece porque o receptor não tem uma antena apropriada, desta forma, o pacote recebido pelo receptor vem com muita falha. A Figura 50 mostra as porcentagens de compatibilidade com os ids cadastrados.

Figura 50 – **Teste e validação do reconhecimento de controle.**



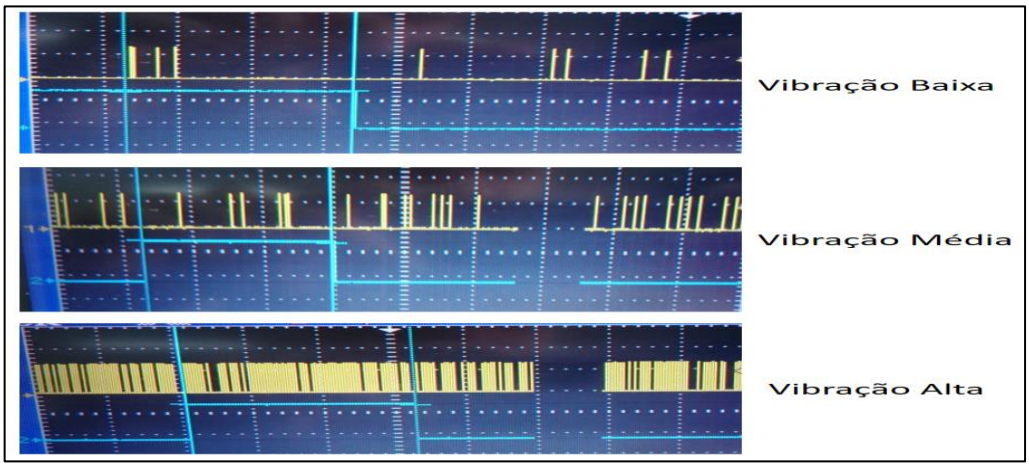
Fonte: Autoria Própria.

Foi cadastrado apenas um controle, com identificação do tipo de caminhão, a carga, o sinal do controle decodificado e o número inteiro do id. Na figura, A, B e C, marcam o teste com mesmo controle cadastrado com distâncias diferentes. O processo de validar é para rotular o pacote recebido como informação de um controle, ou apenas ruído.

Para o sinal de vibração o problema é ele ser contínuo e aleatório, não tem a definição de um pacote contendo informação, como no RF. Para um sinal ter validade com essas características, é necessária uma amostragem grande, por isso 10 mil amostras. Mesmo pegando uma grande janela do sinal, a probabilidade deste quadro ser igual a um novo em uma amostra do mesmo sinal é pequena.

A característica observada então é que os pulsos provocados pela vibração acontecem em tempos diferentes, aleatórios, mas sempre em mesma quantidade.
A amostragem do sinal comparada ponto-a-ponto, vai apresentar bastante diferença, mas a soma de pulsos dentro das 10 amostras vai ser bem próxima. Sendo assim, este é o reconhecimento de padrão utilizado pelo sistema. A figura 51 mostra o processo de amostragem do sinal de vibração.

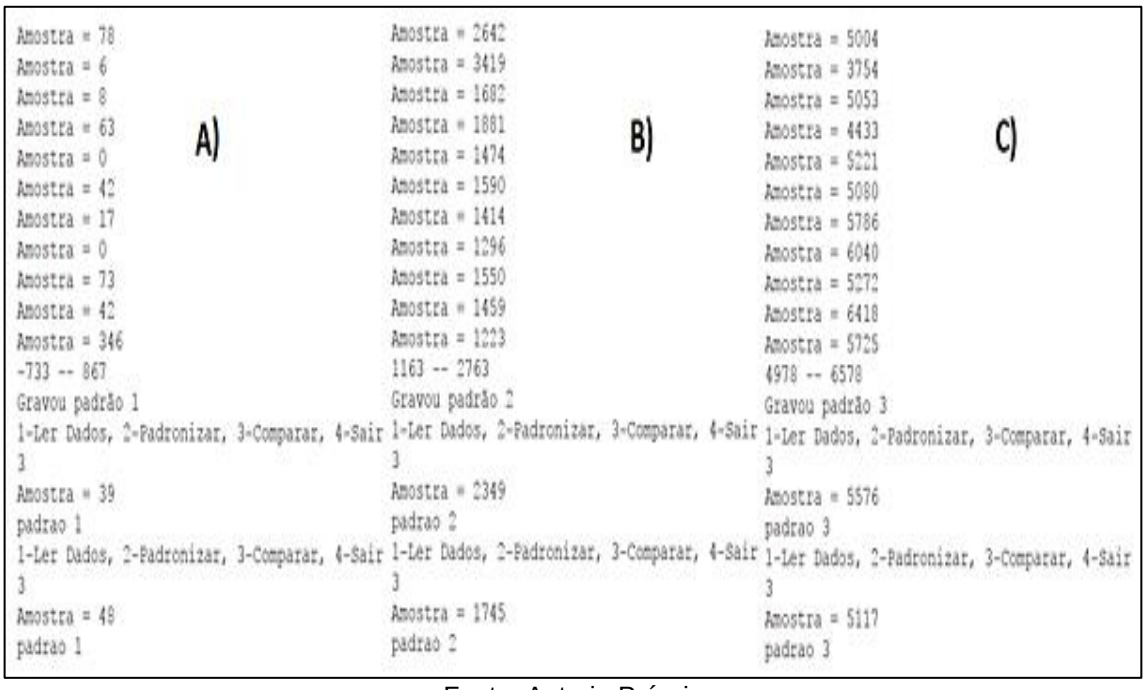
Figura 51 – **Sinais de Vibração**



Fonte: Autoria Própria

É possível perceber a aleatoriedade e o comportamento de soma de pulsos do sinal do sensor em amarelo e em azul a janela de amostra. A Figura 52 mostra o processo de padronização e comparação do padrão em *software.*

Figura 52 – **Padronização e comparação da vibração.**



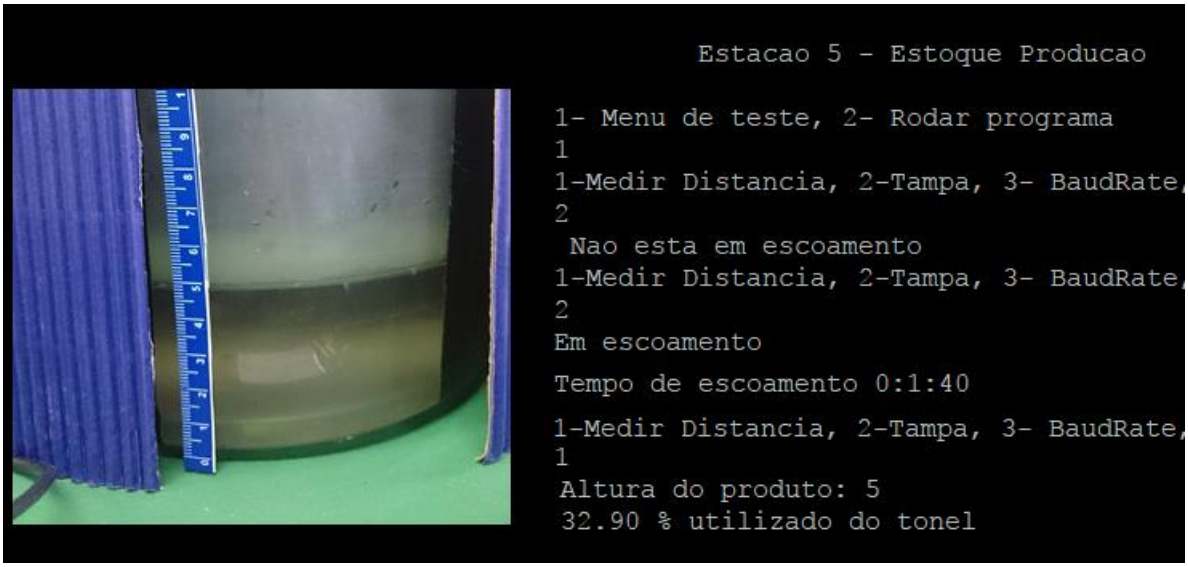
Fonte: Autoria Própria

A definição das faixas de padrões é importante, pois como visto na figura anterior, os valores adquiridos da amostra variam. Para o padrão 1 na figura A, com *range* de 800 para mais e para menos da media, chegou-se ao limite inferior negativo, sendo que o sensor nunca irá apresentar valores menores que zero. A largura do intervalo do padrão define a sensibilidade do sistema, quanto menor, mais padrões

podem ser definidos, e melhor a identificação do sistema de um comportamento. Mas a definição errada dos limites pode fazer o sistema classificar padrões errados.

Para estação de controle de estoque, a variável tempo é a base de funcionamento dos sensores. Para ultrassônico, o tempo de viagem da onda sonora determina a distância. Para o escoamento de açúcar, o tempo pode dar uma noção de volume de produção d produção. A figura 53 mostra o programa de teste da estação

Figura 53 – **Programa de teste da estação 5.**



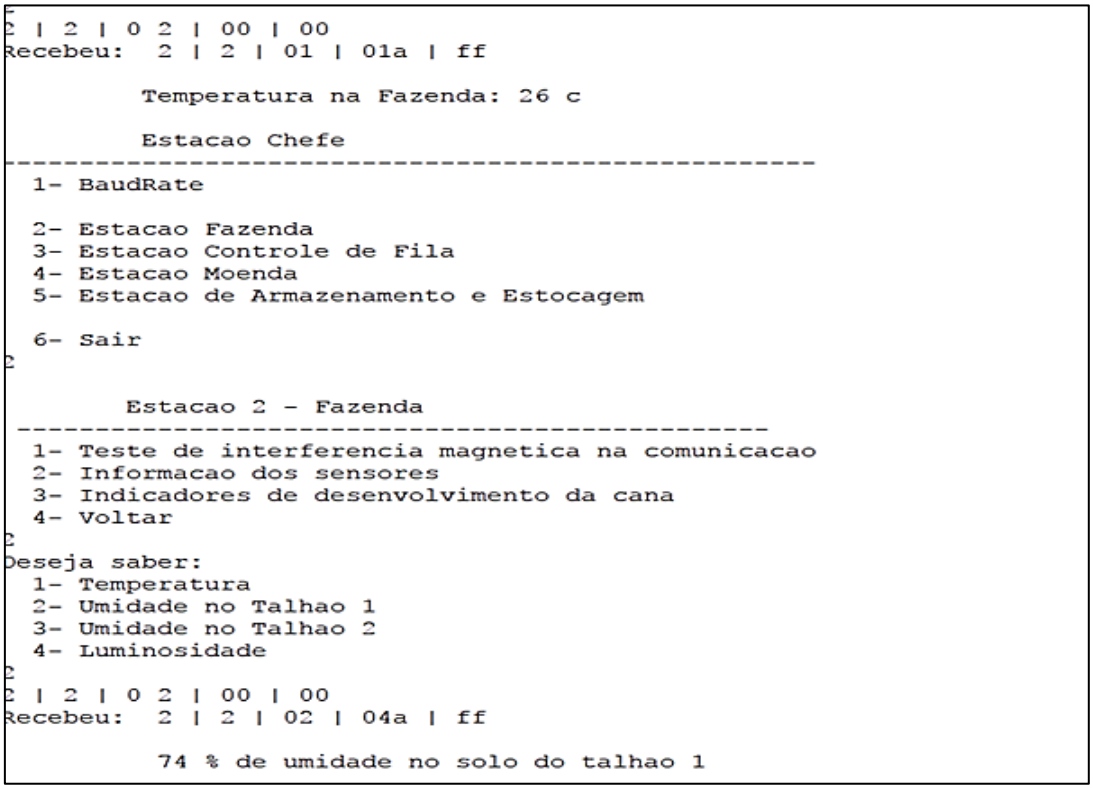
Fonte: Autoria Própria

A função que monitora o escoamento indica no momento da solicitação o *status* da atual condição, está em escoamento ou não. Quando a tampa se abre para o açúcar escoar, o sistema marca o tempo e quando fecha também. Sendo assim o escamento da imagem foi de 1minuto e 40 segundos. O sistema consegue marcar até 24 horas de escoamento.

O sistema operacional que marca o tempo, mesmo se configurado inicialmente, quando a energia da Raspberry é encerrada o sistema deixa de marcar a hora correta. Para as funções da estação a hora correta é irrelevante, apenas a contagem correta dos segundos, milissegundo e hora é importante.

De maneira geral as estações atendem solicitação e encaminham a resposta. O funcionamento de cada uma individualmente é transparente ao gestor. O resultado final do trabalho está representado na figura 54.

Figura 54 – **Resultado final do sistema.**



Fonte: Autoria Própria

Na figura é possível ver duas respostas a duas solicitações diferentes da estação 2 – Fazenda. Os códigos de todos os programas desenvolvidos estão listados em anexo.

5. CONCLUSÃO

O trabalho abordou diversos problemas pontuais que compõem o objetivo final do projeto. O conhecimento de diversas áreas da eletrônica permitiu a solução eficaz e satisfatória para a viabilidade do sistema.

A ideia inicial foi o desenvolvimento de um sistema de controle moderno que se auto gerencia. Para que um sistema tome decisões de como gerenciar suas aplicações, ele necessita de informações em tempo real. Portanto o sistema desenvolvido, com algumas modificações de rede, pode alcançar este propósito.

A forma de desenvolvimento do sistema foi escolhida para ser a base de algo que possa crescer e evoluir; por isso o uso da Raspberry, que permite o uso de banco de dados, interface web, comunicação sem fio e processador multinúcleos.

O trabalho atual está em condições de evoluir também para um sistema supervisório com uma interface homem-máquina gráfica moderna e manipulação via *web*. Como discutido anteriormente, o sistema permite a conexão de outros dispositivos com comunicação ModBus, por exemplo, interagir com CLP (controle lógico programável) e interfaces de controle existente que dispõe o protócolo. Este recurso agrega flexibilidade e acessibilidade ao sistema.

A rede é um dos principais recursos deste projeto, apesar dos problemas durante o desenvolvimento, o resultado final foi satisfatória. Foram estudadas outras alternativas de protocolo para camada física, como o RS-485. No entanto, aplicá-los em toda rede seriam necessários mais módulos, que momentaneamente não era viável.

Equiparando o sistema desenvolvido à uma ferramenta de auxílio ao gestor, o resultado final é satisfatório e cumpre seu objetivo. Analogamente o sistema é para o gestor, como a bolsa de valores é para o mercado de ações, onde os indicadores informam aos correntistas quais ações comprar ou vender.

6. REFERÊNCIAS

AURÉLIO. **Dicionário Aurélio de Português Online** – *BR*. Disponível em: <www.dicionariodoaurelio.com/>. Acessado em: outubro de 2017.

BESTEP. **Chave de vibração, SW-420 (HDX-2)**. 2017

BOCCATO, Levy. **Comunicações: Protocolos. Notas de Aula: Introdução ao Projeto de Sistemas Embarcados**. Universidade Estadual de Campinas (UNICAMP). Disponível em: <http://www.dca.fee.unicamp.br/~lboccato/>. Acessado em outubro de 2017.

BOYLESTAD, Robert L. **Introdução à Análise de Circuitos.**10ed. São Paulo: Pearson, 2004.

CASTRO, Fábio C. C. **Famílias Lógicas. Capítulo 4. Departamento de engenharia Elétrica.** 2011

CONAB. Companhia Nacional de Abastecimento. **Acompanhamento da Safra Brasileira. Cana-de-Açúcar. Quarto levantamento (v.2 Safra 2015/16 – N.4 | Abril 2016)**. Disponível em:< http://www.conab.gov.br>. Acessado em maio de 2017.

CUNHA, Alessandro. **Sistema Embarcados**. Revista Sabert Eletrônica 414. Editora Saber 2007.

DECAGON. ***Decagon Devices*** – USA. Disponível em: <www.decagon.com>. Acessado em: outubro de 2017.

EID, R. **Progresso Técnico na Agroindústria Sucroalcooleira.** Informações Econômicas – Governo do estado de São Paulo, 1996.

EPECOS. **NTC *Thermistors for temperature measurement.*** Série B57164, 2006

FOROUZAN, Behrouz A. Comunicação de dados e redes de computadores. Quarta edição. Editora AMGH,2010.

Gmax. **Controle RF 433 MHZ modelo 2005-TXCGABR.** Gmax Dispositivos para Automação – BR. 2005

GUARESE, Giuliano B. M. **Arquitetura Híbrida de Comunicação Para Ambientes de Automação Industrial: Protocolos IEEE 802.15.4 e Modbus RTU sobre RS485**. 2011

IANNONI, Ana P. MORABITO, Reinaldo. **Análise do Sistema Logístico de Recepção de Cana-de-açúcar: Um Estudo de Caso Utilizando Simulação Discreta.** 2002

INMETRO. **Sistema Internacional de Medidas SI.** Primeira edição Brasileira da oitava edição do BIPM. Rio de Janeiro, 2012

LATHI, B. P. **Sinais e Sistemas Lineares.** 2ed, Bookman. São Paulo, 2007.

LAUDARES, F. A. L., CRUZ, F. A. O., CRUZ, T., Bigansolli, A. R. **Instrumentação para Ensino de Física da UFRuralRJ: experiências docentes para a introdução tecnológica.** Revista de *Formación e Innovación* Educativa Universitária. Vol, 7. 2014

LPRS. ***LDR CdS PHOTO CELL* - N5AC-50108.** 2012

MODBUS. *Modbus Organization.* Estados Unidos – *US*. Disponível em: <http://www.modbus.org/>. Acessado em maio de 2017.

MODBUSTOOLS. *Witte Software.* Dinamarca – *DK.* Disponível em: <http://www.modbustools.com/ >. Acessado em maio de 2017.

O´BRIEN, James A. **Sistemas de Informação e as Decisões Gerenciais na Era Internet.** São Paulo: Saraiva 2004

PHILIPS. **PCF8591 8-bit A/D and D/A converter**. 1998

PIRES, S. R. I. **Gestão da cadeia de suprimentos (*Supply Chain Management*): conceitos, estratégias e casos**. São Paulo: Atlas, 2004.

RADIO ACADEMY*.* ***Basic Radio Awareness Modulation and Radio Building Blocks*** – USA. Disponivel em:<www.taitradioacademy.com>. Acessado em: outubro de 2017.

RASPBERRYPI, ***Raspberry pi Fundation.*** **Reino Unido - *UK*** Disponível em: <https://www.raspberrypi.org/>. Acessado em maio de 2017.

RASPBIAN**. Welcome to Raspbian**. Disponível em: <https://www.raspbian.org/>. Acessado em maio de 2017.

RICHARDSON, Matt; WALLACE, Shawn. **Primeiros passos com o raspberry pi**. 1 ed. São Paulo: Novatec, 2013.

ROCHA, Rui. **Norma RS232-C IEEE *Academic***. IEEE *Fundation*, janeiro de 2013.

RUBINSTEIN, Marcelo G.; REZENDE, José F. **Qualidade de serviço em redes 802.11.** XX Simpósio Brasileiro de Redes de Computadores (SBRC2002), 2002.

SILVA, Aneirson Francisco. **Modelagem do planejamento agregado da produção de uma usina sucroalcooleira**. Tese de Doutorado. UNIVERSIDADE FEDERAL DE ITAJUBÁ.2009

SOLOMAN, Sabrie. ***Sensor HandBook.***2 ed.New York – USA: McGraw-Hill, 2009.

TANENBAUM, Andrew S. **Sistemas Operacionais Modernos.** 4ed. São Paulo: Pearson, 2016.

TEXAS INSTRUMENTS. ***Quadruple Bilateral Analog Switch (sn74hc4066).*** 2016

THOMAZINI, Daniel; ALBUQUERQUE, Pedro U. B. **Sensores Industriais – Fundamentos e Aplicações.** 5ed. São Paulo: Érica, 2005.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas Digitais – Princípios e aplicações.** 10ed. São Paulo: Pearson 2007

Tower Pro. **Servo Motor SG90**. Disponível em: <http://www.towerpro.com.tw>. Acessado em outubro de 2017

UNICA, União da Industria de Cana-de-açúcar. **Cenário e Desafios para a Expansão do Setor Sucroenergético.** Disponível em: < http://www.unica.com.br/>. Acessado em maio de 2017.

VERONESI, Ricardo L. M. **RtrASSoc51–Módulo de Comunicação I2C Reconfigurável–rI2C**. 2006.

WEIKEDZ. ***Sensor Modules***. Shenzhen WeiKedz Techonology Co. – CHN.2017

WEILL, P.; ROSS, W. J. **Governança de TI – como as empresas com melhor desempenho administram os direitos decisórios de TI na busca por resultados superiores.** 1.ed. São Paulo: M. Books do Brasil, 2006.

7. ANEXO

7.1 ANEXO A – GESTOR

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <time.h>


#define IN 0
#define OUT 1
#define LOW 0
#define HIGH 1

#define ctrla 23
#define ctrlb 24
#define ctrlc 25
#define ctrld 8

//comunicacao

      int confirmarecebeu =0;
      //ler
      int id, funcao,endhi,endlow,dados1,dados0,crchi,crclow;
      //escrever
      char escreverid, escreverfuncao,escreverendlow;


static int criargpio (int pin)
{
    #define buffer_max 3
    int fd;
    char buffer[buffer_max];
    ssize_t byte_writen;

    fd = open("/sys/class/gpio/export", O_WRONLY);
    byte_writen = snprintf(buffer, buffer_max,"%d", pin);
    write(fd, buffer, byte_writen);

    close (fd);
    return (0);
}

static int direcaogpio (int pin, int dir)
{
    static const char s_directions_str[] = "in\0out";
    #define direction_max 35
    char path[direction_max];
    int fd;

    snprintf(path, direction_max, "/sys/class/gpio/gpio%d/direction", pin);
    fd = open (path, O_WRONLY);

    if (-1 == write(fd, &s_directions_str[IN == dir ? 0:3], IN == dir ?
2:3)){
```

```
      return (-1);}
    close(fd);
    return (0);
}

static int lergpio(int pin)
{
    #define valor_max 30
    char path[valor_max];
    char valor_str[3];
    int fd;

    snprintf(path, valor_max,"/sys/class/gpio/gpio%d/value", pin);
    fd = open(path, O_RDONLY);
    if (-1 == read(fd, valor_str,3)){
    return (-1);}
    close (fd);
    return (atoi(valor_str));
}

static int escrevergpio(int pin, int value)
{

    static const char s_valor_str[] = "01";
    char path [valor_max];
    int fd;

    snprintf(path, valor_max, "/sys/class/gpio/gpio%d/value", pin);
    fd = open (path, O_WRONLY);

    if(1 != write(fd, &s_valor_str[LOW == value ? 0 : 1],1)){
      return(-1);}
    close(fd);
    return (0);
}

void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);
      if (bd == 1)
      {
          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
          printf("Configurado BaudRate em 9600\n");
      }

      if (bd == 2)
      {
          printf("Configurado BaudRate em 19200\n");
          options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 3)
```

```
      {
          printf("Configurado BaudRate em 38400\n");
          options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 4)
      {
          printf("Configurado BaudRate em 115200\n");
          options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
      }

      options.c_iflag = IGNPAR;
      options.c_oflag = 0;
      options.c_lflag = 0;

      tcflush   (uart0_filestream, TCIFLUSH);
      tcsetattr (uart0_filestream, TCSANOW, &options);
    }
}

int ler ()
{

    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/ttyAMA0", O_RDONLY | O_NOCTTY |
O_NDELAY);

    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
    else{
      unsigned char rx_buffer [200];

while (confirmarecebeu !=1)
{
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 200);

      if(rx_length <= 0)
      {
            confirmarecebeu = 0;
      }
            //confirmarecebeu = 1;

if ((rx_length > 1)&&(rx_buffer[0]<6))
   {
          confirmarecebeu =1;
          rx_buffer[rx_length];

          id = rx_buffer[0];
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];

         // rx_buffer[rx_length] ='\0';

          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n", id,
funcao,endhi,endlow,dados1,dados0,rx_buffer[6],rx_buffer[7]);
```

```
          return (0);
   }
}
    }
    close (uart0_filestream);

}

void escrever()
{
    int uart0_filestream = -1;
    int i;
    uart0_filestream = open("/dev/ttyAMA0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];


      tx_buffer[1] = escreverfuncao;
      tx_buffer[2] = 0x0;                //end hi
      tx_buffer[3] = escreverendlow;
      tx_buffer[4] = 0x0;                //dados1
      tx_buffer[5] = 0x0;                //dados0
      tx_buffer[6] = 0x0;                //crc
      tx_buffer[7] = 0x0;                //crc
      tx_buffer[0] = escreverid;

    printf ("%x | %x | %x %x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0], 7);

      close (uart0_filestream);
      if ( count<0)
      {
            printf("Problema na transmissao\n");
      }

}

void limpacomunicacao()
{
      confirmarecebeu =0;
      //ler
      id=0;
      funcao=0;
      endhi=0;
      endlow=0;
      dados1=0;
      dados0=0;
      crchi=0;
      crclow=0;

      //escrever
      escreverid = 0;
      escreverfuncao=0;
      escreverendlow=0;
}
```

```
void traducao ()
{
      if (funcao == 0x1)
      {
            if (endlow + endhi + dados1+dados0 == 40)
            {
              printf("\n\t Sem interferencia magnetica na estacao %d\n",
id);
            }
      }

    if (id == 0x2)
    {
      if (funcao == 0x2)
      {
            if (endlow == 0x1)
            {
              printf ("\n\t Temperatura na Fazenda: %d c \n",dados0);
            }
            if (endlow == 0x2)
            {
              printf ("\n\t  %d %% de umidade no solo do talhao 1
\n",dados0);
            }
            if (endlow == 0x3)
            {
              printf ("\n\t  %d %% de umidade no solo do talhao 2
\n",dados0);
            }
            if (endlow == 0x4)
            {
              printf ("\n\t  %d %% de luminosidade na fazenda \n",dados0);
            }
      }
      if (funcao == 0x3)
      {
            if (endlow == 0x1)
            {
              printf("\n\t Indicador de Temperatura:");
              if (dados0 == 1)
              {
                  printf (" a baixo do ideal\n");
              }
              if (dados0 == 2)
              {
                  printf (" na faixa ideal\n");
              }
              if (dados0 == 3)
              {
                  printf (" acima do ideal\n");
              }

            }
            if (endlow == 0x2)
            {
              int numero = dados1*100+dados0*10;
              printf ("\n\t  Indicador de luminosidade: %d Lux \n",numero);
            }
      }
    }
```

```
      if (id == 0x3)
      {
            if (funcao == 0x3)
            {
                  if (endlow == 0x1)
                  {
                        printf ("\n\t Gravou controle 1\n\t Id controle: %d
\n\t Capacidade de transporte do caminhao: %d\n", dados1, dados0);
                  }
                  if (endlow == 0x2)
                  {
                        printf ("\n\t Gravou controle 2\n\t Id controle: %d
\n\t Capacidade de transporte do caminhao: %d\n", dados1, dados0);
                  }
                  if (endlow == 0x3)
                  {
                        printf ("\n\t Gravou controle 3\n\t Id controle: %d
\n\t Capacidade de transporte do caminhao: %d\n", dados1, dados0);
                  }
            }
            if (funcao == 0x4)
            {
                  if (endlow == 0)
                  {
                        printf ("Receptor RF desligado\n");
                  }
                  if (endlow == 1)
                  {
                        printf ("Receptor RF Ligado\n");
                  }
            }
            if (funcao == 0x5)
            {
                  if (endlow == 0)
                  {
                        printf ("Proibidas todas filas\n");
                  }
                  if (endlow == 1)
                  {
                        printf ("Liberado Fila principal\n");
                  }
                  if (endlow == 2)
                  {
                        printf ("Liberado Fila secundaria\n");
                  }

            }

      }

      if (id == 0x4)
      {
            if (funcao == 0x2)
            {
                  if (endlow == 0x1)
                  {
                        printf ("\n\t Moenda esta a baixo do nivel de
alimentacao\n");
                  }
                  if (endlow == 0x2)
```

```
                  {
                        printf ("\n\t Moenda esta no nivel bom de
alimentacao\n");
                  }
                  if (endlow == 0x3)
                  {
                        printf ("\n\t Moenda esta acima do nivel de
alimentacao\n");
                  }
            }
            if (funcao == 0x3)
            {
                  if (endlow == 0x1)
                  {
                        printf ("\n\t Gravou Padrao de alimentacao
baixo\n");
                  }
                  if (endlow == 0x2)
                  {
                        printf ("\n\t Gravou Padrao de alimentacao Bom\n");
                  }
                  if (endlow == 0x3)
                  {
                        printf ("\n\t Gravou Padrao de alimentacao
Alto\n");
                  }
            }
      }
    if (id == 0x5)
    {

      if (funcao == 0x2)
      {
          if ( endlow == 0x1)
          {
            printf ("\n\t %d cm de Etanol no tanque\n", dados0);
          }
          if ( endlow == 0x2)
          {
            printf ("\n\t %d %% usado da capacidade do tanque\n", dados0);
          }
      }

      if (funcao == 0x3)
      {
          if ( endlow == 0x1)
          {
            if ( dados0 == 1)
            {
                printf ("\n\t Esta chegando acucar da producao agora \n");
            }
            else
            {
                printf ("\n\t Nao esta escoando acucar agora\n");
            }
          }
          if ( endlow == 0x2)
          {
            printf ("Ultimo escoamento foi de 0:%d:%d \n",dados1,dados0 );
          }
      }
```

```
    }
}

int main ()
{
      criargpio (ctrla);
      criargpio (ctrlb);
      criargpio (ctrlc);
      criargpio (ctrld);

      direcaogpio (ctrla, OUT);
      direcaogpio (ctrlb, OUT);
      direcaogpio (ctrlc, OUT);
      direcaogpio (ctrld, OUT);

      escrevergpio (ctrla, LOW);
      escrevergpio (ctrlb, LOW);
      escrevergpio (ctrlc, LOW);
      escrevergpio (ctrld, LOW);

      int bd0, repet=0,i=0;
          limpacomunicacao();
          printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
          scanf("%d",&bd0);
          baudrate(bd0);

    int comando, funcaoestacao=0,endlowestacao=0;
    while ( comando != 6)
    {
      //printf("\e[H\e[2J");
      printf ("\n\t Estacao Chefe\n----------------------------------------
------------\n");
      printf("  1- BaudRate  \n\n  2- Estacao Fazenda  \n  3- Estacao
Controle de Fila  \n  4- Estacao Moenda  \n  5- Estacao de Armazenamento e
Estocagem  \n\n  6- Sair\n");
      scanf("%d", &comando);

      if (comando == 1)
      {
          printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
          scanf("%d",&bd0);
          baudrate(bd0);
      }
      if (comando == 2)
      {
      escrevergpio (ctrla,HIGH);
            printf (" \n\tEstacao 2 - Fazenda\n ----------------------------
---------------------\n");
            printf ("  1- Teste de interferencia magnetica na comunicacao
\n  2- Informacao dos sensores \n  3- Indicadores de desenvolvimento da
cana \n  4- Voltar\n");
            scanf  ("%d",&funcaoestacao);
            if (funcaoestacao == 1)
            {

                  escreverid     = 2;
                  escreverfuncao = 1;
                  escreverendlow = 0;

                  escrever();
                  limpacomunicacao();
```

```
                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;

            }

            if (funcaoestacao == 2)
            {
                  printf("Deseja saber:\n  1- Temperatura \n  2- Umidade no
Talhao 1 \n  3- Umidade no Talhao 2 \n  4- Luminosidade \n");
                  scanf ("%d", &endlowestacao);

                  escreverid     = 2;
                  escreverfuncao = 2;
                  escreverendlow = endlowestacao;

                  escrever();

                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;
                  endlowestacao = 0;
            }

            if (funcaoestacao == 3)
            {
                  printf("Deseja saber:\n  1- Indicador de Temperatura \n
2- Indicador de Luminosidade \n");
                  scanf ("%d", &endlowestacao);

                  escreverid     = 2;
                  escreverfuncao = 3;
                  escreverendlow = endlowestacao;

                  escrever();
                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;
                  endlowestacao = 0;
            }
      escrevergpio(ctrla, LOW);
      }// comando 2

      if (comando == 3)
      {
      escrevergpio(ctrlb, HIGH);
            printf (" \n\tEstacao 3 - Controle de Fila e caminhoes\n ------
----------------------------------------\n");
            printf ("  1- Teste de interferencia magnetica na comunicacao
\n  3- Cadastro de Controle \n  4- Desligar/Ligar Cancela \n  5- Controlar
Fila\n  6- Voltar\n");
            scanf  ("%d",&funcaoestacao);
            if (funcaoestacao == 1)
            {
```

```
                escreverid     = 3;
                escreverfuncao = 1;
                escreverendlow = 0;


                escrever();
                ler();

                //while (confirmarecebeu !=1)
                //{
                //    ler ();
                //}
                traducao ();
                limpacomunicacao();
                funcaoestacao = 0;
            }
            if (funcaoestacao == 3)
            {
                printf ("Deseja Gravar Controle:\n  1- Controle 1  \n  2-
Controle 2 \n  3- Controle 3 \n");
                scanf ("%d", &endlowestacao);

                escreverid     = 3;
                escreverfuncao = 3;
                escreverendlow = endlowestacao;

                escrever();
                ler();
                //limpacomunicacao();

                //while (confirmarecebeu !=1)
                //{
                //    ler ();
                //}
                traducao ();
                limpacomunicacao();
                funcaoestacao = 0;
                endlowestacao = 0;
            }
            if (funcaoestacao == 4)
            {
                printf ("Deseja:\n  0- Desligar receptor RF \n  1- Ligar
receptor RF \n  ");
                scanf ("%d", &endlowestacao);

                escreverid     = 3;
                escreverfuncao = 4;
                escreverendlow = endlowestacao;

                escrever();
                ler();
                //limpacomunicacao();

                //while (confirmarecebeu !=1)
                //{
                //    ler ();
                //}
                traducao ();
                limpacomunicacao();
                funcaoestacao = 0;
                endlowestacao = 0;
```

```
            }
            if (funcaoestacao == 5)
            {
                  printf ("Deseja:\n  0- Proibir todas Filas  \n  1-
Liberar fila principal \n  2- Liberar Fila Secundaria\n ");
                  scanf ("%d", &endlowestacao);

                  escreverid     = 3;
                  escreverfuncao = 5;
                  escreverendlow = endlowestacao;

                  escrever();
                  ler();
                  //limpacomunicacao();

                  //while (confirmarecebeu !=1)
                  //{
                  //    ler ();
                  //}
                  traducao ();
                  limpacomunicacao();
                  funcaoestacao = 0;
                  endlowestacao = 0;
            }
      escrevergpio(ctrlb, LOW);
      }//comando 3

      if (comando == 4)
      {
      escrevergpio(ctrlc, HIGH);
            printf (" \n\tEstacao 4 - Monitoramento Moenda\n --------------
----------------------------------\n");
            printf ("  1- Teste de interferencia magnetica na comunicacao
\n  2- Verificar comportamento da Moenda \n  3- Gravar Comportamento \n  4-
Voltar\n");
            scanf  ("%d",&funcaoestacao);
            if (funcaoestacao == 1)
            {
                  escreverid     = 4;
                  escreverfuncao = 1;
                  escreverendlow = 0;

                  escrever();
                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;
            }
            if (funcaoestacao == 2)
            {
                  escreverid     = 4;
                  escreverfuncao = 2;
                  escreverendlow = 0;

                  escrever();
                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;
```

```
            }
            if (funcaoestacao == 3)
            {
                  printf ("Deseja Gravar Nivel de alimentação de cana:\n
1- Baixo  \n  2- Bom \n  3- Alto ");
                  scanf ("%d", &endlowestacao);

                  escreverid     = 4;
                  escreverfuncao = 3;
                  escreverendlow = endlowestacao;

                  escrever();
                  ler ();
                  traducao ();
                  limpacomunicacao();

                  funcaoestacao = 0;
                  endlowestacao = 0;
            }

      escrevergpio(ctrlc, LOW);
      }//comando 4

      if (comando == 5)
      {
      escrevergpio(ctrld, HIGH);
            printf (" \n\tEstacao 5 - Armazenamento e Estocagem\n ---------
----------------------------------------\n");
            printf ("  1- Teste de interferencia magnetica na comunicacao
\n  2- Tanque de Etanol \n  3- Armazem de Acucar \n  4- Voltar\n");
            scanf  ("%d",&funcaoestacao);
            if (funcaoestacao == 1)
            {

                  escreverid     = 5;
                  escreverfuncao = 1;
                  escreverendlow = 0;

                  escrever();
                  ler ();
                  traducao ();

                  limpacomunicacao();
                  funcaoestacao = 0;
            }

            if (funcaoestacao == 2)
            {
                  printf ("Deseja saber:\n  1- Nivel do produto \n  2-
Capacidade utilizada \n ");
                  scanf ("%d", &endlowestacao);

                  escreverid     = 5;
                  escreverfuncao = 2;
                  escreverendlow = endlowestacao;

                  escrever();
                  ler ();
                  traducao ();

                  limpacomunicacao();
```

```
                funcaoestacao = 0;
                endlowestacao = 0;
            }

            if (funcaoestacao == 3)
            {
                printf ("Deseja saber:\n  1- Status de Producao \n  2-
Tempo de escoamento da ultima producao \n  3- Voltar\n ");
                scanf ("%d", &endlowestacao);

                escreverid     = 5;
                escreverfuncao = 3;
                escreverendlow = endlowestacao;

                escrever();
                ler ();
                traducao ();
                limpacomunicacao();

                funcaoestacao = 0;
                endlowestacao = 0;
            }
      escrevergpio(ctrld, LOW);
      }//comando 5
    }//while principal
}//main
```

7.2 ANEXO B – ESTAÇÃO CONTROLE DE FAZENDA

```
// compilar -lm no fim
#include <termios.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <math.h>

unsigned char valor[4];

double pcentluminosidade,  pcenthumidadesolo1, pcenthumidadesolo2,temp;
int indicadortemp,indicadorlux;

//comunicacao
    //ler
int identidade,endhi,endlow,funcao, dados1,dados0;

    //escrever
char  escreverfuncao, escreverendhi,escreverendlow,
escreverdados1,escreverdados0;


void lerdados()
{
    int repeticao =3;
    int i,r,fd;
    double luminosidade, humidadesolo1, humidadesolo2;

    unsigned char comando[2];
```

```
    useconds_t delay = 2000;

    char *dev = "/dev/i2c-1";
    int addr = 0x48;

    fd = open (dev, O_RDWR);
    if(fd<0)
      {
      perror("erro ao abriu i2c\n");
      }

    r = ioctl (fd,I2C_SLAVE, addr);
    if ( r<0){
      printf ("erro\n");}

    while(repeticao !=1)
    {
      repeticao  --;
      for(i=0; i<4; i++){
          comando[0] = 0x40 | ((i+1) & 0x03);
          comando[1]++;

          r = write(fd,&comando, 2);
          usleep(delay);

          r = read(fd,&valor [i], 1);
          if (r!=1){
            perror("Leitura i2c\n");}
          usleep(delay);
      }
    }

    double vr, rntc,logntc,b;
    float c;
    #define resistor 10000

    vr = ((valor[3]*3.3)/255);
    rntc = (((5*resistor) - (resistor*vr))/vr);

    logntc = log(rntc);
    b = (0.0002339*logntc);
    c = pow (logntc, 3);
    c = c*0.00000008863;
//    printf ("valor [3]: %d \n",valor[3]);
//    printf ("Vr = %.2f Rntc = %.2f Log rntc = %f b=%f c=%f\n", vr, rntc,
logntc,b,c);
    temp = (1/(0.0011303+b+c))-273.15;

//    printf("%d valor 1\n", valor[1]);
//    printf("%d valor 2\n", valor[2]);
//    printf("%d valor 0\n", valor[0]);

      luminosidade  = 255 - valor[0];
      humidadesolo1 = 255 - valor[1];
      humidadesolo2 = 255 - valor[2];

//    printf("%d umidadesolo 1\n",humidadesolo1);

       pcentluminosidade  = ((luminosidade /162)*100);//93 -> 100% portanto
255 - 93 = 162
```

```
       pcenthumidadesolo1 = ((humidadesolo1/115)*100);//140 -> 100% 255 -
140 = 115
       pcenthumidadesolo2 = ((humidadesolo2/115)*100);

//    printf ("%.2f %% luminoso\n", pcentluminosidade);
//    printf ("Temperatura:  %.2f\n",temp);
//    printf ("%.2f %% de humidade no solo 1\n", pcenthumidadesolo1);
//    printf ("%.2f %% de humidade no solo 2\n", pcenthumidadesolo2);

      close (fd);

}
void indicadores (int indicador)
{
    lerdados();

    if (indicador == 1)
    {
      if (temp >= 20 && temp <=35)
      {
          printf("\n temperatura dentro da faixa boa\n");
          indicadortemp = 2;
      }
      if (temp < 20)
      {
          printf("\n temperatura abaixo da faixa boa\n");
          indicadortemp = 1;
      }
      if (temp > 35)
      {
          printf("\n temperatura acima da faixa boa\n");
          indicadortemp = 3;
      }
    }

    if (indicador == 2)
    {
      double vldr, rldr,lux;
      vldr = ((valor[0]*3.3)/255);
      rldr = (((5*1000) - (1000*vldr))/vldr);
//    printf ("Valor [0]: %d\nVldr: %.2f Rldr:%.2f\n",valor[0],vldr,rldr);
      #define calibraldr 200
      lux = (calibraldr * rldr)/1000;
      indicadorlux = lux;
//    printf("\n Lux: %.2f\n", lux);
      printf("\n Lux: %d\n", indicadorlux);
    }

}
void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);
```

```
      if (bd == 1)
      {
          printf("Configurado BaudRate em 9600\n");
          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 2)
      {
          printf("Configurado BaudRate em 19200\n");
          options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 3)
      {
          printf("Configurado BaudRate em 38400\n");
          options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 4)
      {
          printf("Configurado BaudRate em 115200\n");
          options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
      }
      options.c_iflag = IGNPAR;
      options.c_oflag = 0;
      options.c_lflag = 0;

      tcflush   (uart0_filestream, TCIFLUSH);
      tcsetattr (uart0_filestream, TCSANOW, &options);
    }

}

void ler ()
{
    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/serial0", O_RDONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
    else{
      unsigned char rx_buffer [400];
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 400);

      if(rx_length <= 0)
      {

      }
      else
      {
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];
          identidade = rx_buffer[0];

          rx_buffer[rx_length] ='\0';
```

```
          if(rx_buffer[0] == 0x4);
          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n",
rx_buffer[0], rx_buffer[1],rx_buffer[2],
rx_buffer[3],rx_buffer[4],rx_buffer[5],rx_buffer[6],rx_buffer[7]);
      }
    }
    close (uart0_filestream);

}

void escrever()
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];
      unsigned char *p_tx_buffer;

    tx_buffer[0] = 0x02; //ID
    tx_buffer[1] = escreverfuncao;
    tx_buffer[2] = escreverendhi;
    tx_buffer[3] = escreverendlow;
    tx_buffer[4] = escreverdados1;
    tx_buffer[5] = escreverdados0;
    tx_buffer[6] = 0xf; //crc
    tx_buffer[7] = 0xf; //crc

    printf ("send:  %x | %x | %x%x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0],14);

    if (count<0)
    {
      printf ("problema na transmissao\n");
    }
    close (uart0_filestream);
}

void limpacomunicacao ()
{
    escreverfuncao  = 0;
    escreverendhi   = 0;
    escreverendlow  = 0;
    escreverdados1  = 0;
    escreverdados0  = 0;
    funcao = 0;
    endhi  = 0;
    endlow = 0;
    dados1 = 0;
    dados0 = 0;
    identidade = 0;
}

void main()
{
```

```
    int x1;
    printf ("\n\tEstacao 2 - Monitoramento Fazenda\n\n");
    printf ("1- Menu de teste, 2- Rodar programa \n");
    scanf ("%d", &x1);

    if (x1 == 1)
    {
      int comando;
      while (comando != 3)
      {
        printf("1-Ler, 2-Indicadores 3-Sair\n");
        scanf("%d",&comando);
        if (comando ==1)
        {
            lerdados();
        }
        if (comando ==2)
        {
          int esco;
          printf ("Indicadores\n 1-Temperatura, 2- Luminosidade\n");
          scanf("%d",&esco);
          indicadores(esco);
        }
      }
    }
    if (x1 == 2)
    {
       int bd0, temporario=0;
       printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
       scanf("%d",&bd0);
       baudrate(bd0);

       while (1)
       {
          ler ();
          usleep(1000000);//1s

            if ((identidade == 0x2) && (funcao == 0x2))
            {
                  lerdados();
                  escreverfuncao = 0x2;
                  if(endlow == 1)
                    {
                        int temperatura;
                        temperatura = temp;
                        printf("Temperatura: %d\n", temperatura);

                        escreverendhi  = 0x0;
                        escreverendlow = 0x1;
                        escreverdados1 = 0x0;
                        escreverdados0 = temperatura;

                        escrever();
                        limpacomunicacao ();

                  }
                  if(endlow == 2)
                 {
                        temporario = pcenthumidadesolo1;
                        printf("Umidade Solo talhao 1: %d %%\n",
temporario);
```

```
                        escreverendhi  = 0x0;
                        escreverendlow = 0x2;
                        escreverdados1 = 0x0;
                        escreverdados0 = temporario;

                        escrever();
                        limpacomunicacao ();
                    }
                    if(endlow == 3)
                   {
                        temporario = pcenthumidadesolo2;
                        printf("Umidade Solo talhao 2: %d %%\n",
temporario);

                        escreverendhi  = 0x0;
                        escreverendlow = 0x3;
                        escreverdados1 = 0x0;
                        escreverdados0 = temporario;

                        escrever();
                        limpacomunicacao ();
                    }
                    if(endlow == 4)
                   {
                        temporario = pcentluminosidade;
                        printf("Luminosidade: %d %%\n", temporario);

                        escreverendhi  = 0x0;
                        escreverendlow = 0x4;
                        escreverdados1 = 0x0;
                        escreverdados0 = temporario;

                        escrever();
                        limpacomunicacao ();
                    }

                    temporario = 0;
                }
                if ((identidade == 0x2) && (funcao == 0x3))
                {
                    printf("\tIndicadores\n");
                    if (endlow == 1)
                    {
                      indicadores (1);
                      escreverendlow = 0x1;
                      temporario = indicadortemp;
                      escreverdados1  = 0x0;
                      escreverdados0 = temporario;
                    }
                    if (endlow == 2)
                    {
                      indicadores (2);
                      escreverendlow = 0x2;

                      float temporario1 = indicadorlux;
                      temporario1 = temporario1/1000;
                      temporario = temporario1;
                      escreverdados1  = temporario;

                      temporario1 = (temporario1-temporario)*100;
```

```
                    temporario = temporario1;
                    escreverdados0 = temporario;

                    printf ("Lux comunicacao %d%d\n",escreverdados1,
escreverdados0);
                }

                escreverfuncao = 0x3;
                escreverendhi  = 0x0;

                escrever();
                limpacomunicacao ();
                temporario = 0;
            }

            if ((identidade == 0xf) && (funcao == 0xf))
            {
                printf("BroadCast\n");
                escreverfuncao = 0xf;
                escreverendhi  = 0xf;
                escreverendlow = 0xf;
                escreverdados1 = 0xf;
                escreverdados0 = 0xf;

                escrever();
                limpacomunicacao ();
            }

            if ((identidade == 0x2) && (funcao == 0x1))
            {
                printf("Interferencia Magnetica\n");
                escreverfuncao = 0x1;
                escreverendhi  = 0xa;
                escreverendlow = 0xa;
                escreverdados1 = 0xa;
                escreverdados0 = 0xa;

                escrever();
                limpacomunicacao ();
            }

       }
     }
}
```

7.3 ANEXO C – ESTAÇÃO CONTROLE FILA

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>


#define IN   0
#define OUT  1
#define LOW  0
```

```
#define HIGH 1

#define dados   18
#define lp1     23
#define ls1     24
#define vccsens 21
#define servo   25

int amostra1[29];
int amostra2[29];
int amostra3[29];

int leitura[240];
int traducao[29];
int idcontrole1,idcontrole2,idcontrole3 =0;
double pcent1, pcent2, pcent3;

//cadastro caminhao
char tipocaminhao1[100],tipocaminhao2[100], tipocaminhao3[100];
int capacidade1,capacidade2,capacidade3;

//comunicacao
//ler
int identidade,endhi,endlow,funcao, dados1,dados0;

//escrever
char  escreverfuncao, escreverendhi,escreverendlow,
escreverdados1,escreverdados0;


static int criargpio (int pin)
{
    #define buffer_max 3
    int fd;
    char buffer[buffer_max];
    ssize_t byte_writen;

    fd = open("/sys/class/gpio/export", O_WRONLY);
    byte_writen = snprintf(buffer, buffer_max,"%d", pin);
    write(fd, buffer, byte_writen);

    close (fd);
    return (0);
}

static int direcaogpio (int pin, int dir)
{
    static const char s_directions_str[] = "in\0out";
    #define direction_max 35
    char path[direction_max];
    int fd;

    snprintf(path, direction_max, "/sys/class/gpio/gpio%d/direction", pin);
    fd = open (path, O_WRONLY);

    if (-1 == write(fd, &s_directions_str[IN == dir ? 0:3], IN == dir ?
2:3)){
      return (-1);}
    close(fd);
    return (0);
}
```

```
static int lergpio(int pin)
{
    #define valor_max 30
    char path[valor_max];
    char valor_str[3];
    int fd;

    snprintf(path, valor_max,"/sys/class/gpio/gpio%d/value", pin);
    fd = open(path, O_RDONLY);
    if (-1 == read(fd, valor_str,3)){
    return (-1);}
    close (fd);
    return (atoi(valor_str));
}

static int escrevergpio(int pin, int value)
{
    static const char s_valor_str[] = "01";
    char path [valor_max];
    int fd;

    snprintf(path, valor_max, "/sys/class/gpio/gpio%d/value", pin);
    fd = open (path, O_WRONLY);

    if(1 != write(fd, &s_valor_str[LOW == value ? 0 : 1],1)){
      return(-1);}
    close(fd);
    return (0);
}

void lerdados()
{
    int estadopres=0,j=0;

    while (estadopres ==0 )
    {
      estadopres = lergpio(dados);
      if ( estadopres ==1)
      {
          while ( j != 240)
          {
            leitura[j] = lergpio(dados);
            usleep (30);
            j++;
          }
      }
    }
//escrevergpio (vccsens, LOW);
}

void decodificar (int prin)
{
    int k=0,i=0,j=0,aux=0;
    int acu[29];

    for (k=0; k<=29; k++)
    {
      acu[k] =0;
      traducao[k] =0;
    }
```

```
    while (i != 240)
    {
      if ( leitura[i] ==1)
      {
          acu[j] ++;
          aux = 1;
      }
      if (leitura [i] ==0 && aux ==1)
      {
          j++;
          aux =0;
      }
      i++;
    }

    j=0;
    while (j != 28)
    {
      if(acu[j] <= 5)
      {
          traducao[j] = 1;
      }
      else
      {
          traducao[j] = 2;
      }
      j++;
    }

    if ( prin == 0 )
    {
      for (k=0; k<=29; k++)
      {
          printf (" %d ",traducao[k]);
      }
      printf("\n");
    }

}

void comparar ()
{
    int i;
    float cont1=0,cont2=0,cont3=0;

    for (i = 0; i <= 28; i++)
      {
          if (amostra1[i] == traducao[i])
          {
            cont1++;
          }
      }
    for (i = 0; i <= 28; i++)
      {
          if (amostra2[i] == traducao[i])
          {
            cont2++;
          }
      }
    for (i = 0; i <= 28; i++)
      {
```

```
          if (amostra3[i] == traducao[i])
          {
            cont3++;
          }
       }

      pcent1 =(cont1/29)*100;
      pcent2 =(cont2/29)*100;
      pcent3 =(cont3/29)*100;

     printf ("%.2f %% \n",pcent1);
     printf ("%.2f %% \n",pcent2);
     printf ("%.2f %% \n",pcent3);

}

void controlefila (int lp, int ls)
{
    if (lp==1)
    {
//    printf("Liberou Principal \n");
      escrevergpio(lp1, HIGH);
    }
    if (ls==1)
    {
//    printf("Liberado Secundario \n");
      escrevergpio(ls1, HIGH);
    }
    if (lp==0)
    {
      escrevergpio(lp1, LOW);
    }
    if (ls==0)
    {
      escrevergpio(ls1, LOW);
    }
}

void acaofila()
{
    printf("Acao fila\n");
/*
    escrevergpio(servo, HIGH);
    usleep (2000);
    escrevergpio(servo, LOW);
    usleep (18000);
    escrevergpio(servo, HIGH);
    usleep (2000);
    escrevergpio(servo, LOW);
    escrevergpio(servo, HIGH);
    usleep (2000);
    escrevergpio(servo, LOW);
    usleep (18000);
    escrevergpio(servo, HIGH);
    usleep (2000);
    escrevergpio(servo, LOW);
*/

    if (pcent1 >=90)
    {
      escrevergpio(servo, HIGH);
```

```
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);

  controlefila(1,0);
  usleep (5000000);
  controlefila(0,0);

}
if (pcent2 >=90)
{
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);

  controlefila(0,1);
  usleep (5000000);
  controlefila(0,0);

}
if (pcent3 >=90)
{
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);
  usleep (18000);
  escrevergpio(servo, HIGH);
  usleep (2000);
  escrevergpio(servo, LOW);

  controlefila(0,1);
  usleep (5000000);
  controlefila(0,0);
```

```
    }

    escrevergpio(servo, HIGH);
    usleep (1000);
    escrevergpio(servo, LOW);
    usleep (19000);
    escrevergpio(servo, HIGH);
    usleep (1000);
    escrevergpio(servo, LOW);
    usleep (19000);

}

void cadastrar(int cad)
{
    int i;
    printf("Cadostro %d\n",cad);
    if (cad == 1)
    {
      int ch;
      while ((ch=fgetc(stdin)) != EOF && ch !='\n'){}
      printf("Digite o tipo de caminhao: ");
      fgets (tipocaminhao1, sizeof(tipocaminhao1),stdin);
      printf("\nCapacidade de Transporte em Kg: ");
      scanf ("%d",&capacidade1);
      printf("\n");

      for (i = 0; i <= 28; i++)
      {
          amostra1[i] =traducao[i] ;
      }

      printf("Cadastrou 1\n");
    }
    if (cad == 2)
    {

      int ch;
      while ((ch=fgetc(stdin)) != EOF && ch !='\n'){}
      printf("Digite o tipo de caminhao: ");
      fgets (tipocaminhao2, sizeof(tipocaminhao2),stdin);
      printf("\nCapacidade de Transporte em Kg: ");
      scanf ("%d",&capacidade2);
      printf("\n");
      for (i = 0; i <= 28; i++)
      {
          amostra2[i] =traducao[i] ;
      }
      printf("Cadastrou 2\n");
    }
    if (cad == 3)
    {
      int ch;
      while ((ch=fgetc(stdin)) != EOF && ch !='\n'){}
      printf("Digite o tipo de caminhao: ");
      fgets (tipocaminhao3, sizeof(tipocaminhao3),stdin);
      printf("\nCapacidade de Transporte em Kg: ");
      scanf ("%d",&capacidade3);
      printf("\n");
      for (i = 0; i <= 28; i++)
      {
```

```
          amostra3[i] =traducao[i] ;
      }
      printf("Cadastrou 3\n");
    }
}

void consultacadastro(int consulta)
{
    int i;
    printf("Consulta de Cadastro\n");

    if (consulta == 1 )
    {
      printf("\n\tCadastro 1 \n Tipo: %s capacidade: %d Kg \n Sinal
controle: ",tipocaminhao1, capacidade1);
      for (i = 0; i <= 28; i++)
      {
          printf("%d",amostra1[i]);
      }
      printf ("\n");
    }

    if (consulta == 2 )
    {
      printf("\n\tCadastro 2 \n Tipo: %s capacidade: %d Kg \n Sinal
controle: ",tipocaminhao2, capacidade2);
      for (i = 0; i <= 28; i++)
      {
          printf("%d",amostra2[i]);
      }
      printf ("\n");
    }

    if (consulta == 3 )
    {
      printf("\n\tCadastro 3 \n Tipo: %s capacidade: %d Kg \n Sinal
controle: ",tipocaminhao3, capacidade3);
      for (i = 0; i <= 28; i++)
      {
          printf("%d",amostra3[i]);
      }
      printf ("\n");
    }
}

int validapacote ()
{
    printf ("Validando...\n");
    int i, cont=0;
    for (i = 0; i <= 28; i++)
    {
      cont = cont + traducao[i];
      //printf ("%d", traducao[i]);

    }
    //printf ("\n");
    //printf ("cont: %d\n", cont);
    if ( cont <=28)
    {
      //printf("ruido\n");
      return 0;
```

```
    }
    else
    {
      return 1;
    }
}

void comunicacontrole (int comunicaid)
{
    int  x=0, i=0;

    if (comunicaid ==1)
    {
      while ( i != 28)
      {
          x = 28 - i;
          if ( amostra1[i] == 1)
          {
            idcontrole1 |= (1 << i);
          }
          else
          {
            idcontrole1 |= (0 << i);
          }
          i++;
      }
    printf ("Id controle: %d\n", idcontrole1);
    }
    x=0;
    i=0;
    if (comunicaid ==2)
    {
      while ( i != 28)
      {
          x = 28 - i;
          if ( amostra2[i] == 1)
          {
            idcontrole2 |= (1 << i);
          }
          else
          {
            idcontrole2 |= (0 << i);
          }
          i++;
      }
    printf ("Id controle 2: %d\n", idcontrole2);
    }
    x=0;
    i=0;

    if (comunicaid ==3)
    {
      while ( i != 28)
      {
          x = 28 - i;
          if ( amostra3[i] == 3)
          {
            idcontrole3 |= (1 << i);
          }
          else
          {
```

```
            idcontrole3 |= (0 << i);
          }
          i++;
      }
    printf ("Id controle 3: %d\n", idcontrole3);
    }


}

void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);

      if (bd == 1)
      {
          printf("Configurado BaudRate em 9600\n");
          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 2)
      {
          printf("Configurado BaudRate em 19200\n");
          options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 3)
      {
          printf("Configurado BaudRate em 38400\n");
          options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 4)
      {
          printf("Configurado BaudRate em 115200\n");
          options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
      }
      options.c_iflag = IGNPAR;
      options.c_oflag = 0;
      options.c_lflag = 0;

      tcflush   (uart0_filestream, TCIFLUSH);
      tcsetattr (uart0_filestream, TCSANOW, &options);
    }

}

void ler ()
{
    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/serial0", O_RDONLY | O_NOCTTY |
O_NDELAY);
```

```
    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
    else{
      unsigned char rx_buffer [400];
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 400);

      if(rx_length <= 0)
      {

      }
      else
      {
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];
          identidade = rx_buffer[0];

          rx_buffer[rx_length] ='\0';

          if(rx_buffer[0] == 0x4);
          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n",
rx_buffer[0], rx_buffer[1],rx_buffer[2],
rx_buffer[3],rx_buffer[4],rx_buffer[5],rx_buffer[6],rx_buffer[7]);
      }
    }
    close (uart0_filestream);

}

void escrever()
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];
      unsigned char *p_tx_buffer;

    tx_buffer[0] = 0x03; //ID
    tx_buffer[1] = escreverfuncao;
    tx_buffer[2] = escreverendhi;
    tx_buffer[3] = escreverendlow;
    tx_buffer[4] = escreverdados1;
    tx_buffer[5] = escreverdados0;
    tx_buffer[6] = 0xf; //crc
    tx_buffer[7] = 0xf; //crc

    printf ("send:  %x | %x | %x%x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0],14);

    if (count<0)
    {
```

```
      printf ("problema na transmissao\n");
    }
    close (uart0_filestream);
}

void limpacomunicacao ()
{
    escreverfuncao  = 0;
    escreverendhi   = 0;
    escreverendlow  = 0;
    escreverdados1  = 0;
    escreverdados0  = 0;
    funcao = 0;
    endhi  = 0;
    endlow = 0;
    dados1 = 0;
    dados0 = 0;
    identidade = 0;
}


int main ()
{
    int x,x1;
    int inicio1, inicio2, inicio3;
    criargpio(dados);
    direcaogpio(dados, IN);

    criargpio(lp1);
    direcaogpio(lp1, OUT);
    criargpio(ls1);
    direcaogpio(ls1, OUT);

    criargpio(vccsens);
    direcaogpio (vccsens, OUT);
    escrevergpio (vccsens, HIGH);

    criargpio(servo);
    direcaogpio (servo, OUT);

    escrevergpio(servo, HIGH);
    usleep (1000);
    escrevergpio(servo, LOW);
    usleep(19000);
    escrevergpio(servo, HIGH);
    usleep (1000);
    escrevergpio(servo, LOW);


    printf ("\n\tEstacao 3 - Controle de fila \n\n");
    printf ("1- Menu de teste, 2- Rodar programa \n");
    scanf ("%d", &x1);

    if (x1 == 1)
    {
      while (x!=8)
      {
          int controle;
          printf("1- Ler, 2- Cadastrar, 3- Comparar, 4- Fila, 5- Consultar
Cadastro, 6- Id controle, 7- Cancela 8- Sair\n");
          scanf("%d", &x);
```

```
            if (x==1)
            {
              controle=0;
              while (controle !=1)
              {
                  lerdados();
                  decodificar(0);
                  controle = validapacote();
              }
            }
            if (x==2)
            {
              printf("Cadastrar controle\n");
              controle=0;
              int cadastro;
              while (controle !=1)
              {
                  lerdados();
                  decodificar(0);
                  controle = validapacote();
              }
              printf("Digite o id do controle\n");
              scanf("%d",&cadastro);
              cadastrar(cadastro);
            }
            if (x==3)
            {
              int j;
              controle=0;
              while (controle !=1)
              {
                  lerdados();
                  decodificar(0);
                  controle = validapacote();
              }
              comparar();
            }
            if (x==4)
            {
              int liga;
              printf ("1- Liberar principal, 2- Liberar secundaria, 3-
Proibir todas\n");
              scanf("%d",&liga);
              if (liga == 1)
              {
                  controlefila(1,0);
              }
              if (liga == 2)
              {
                  controlefila(0,1);
              }
              if (liga == 3)
              {
                  controlefila(0,0);
              }
            }
            if (x==5)
            {
              int con;
              printf("Consultar Cadastro 1, 2 ou 3\n");
              scanf("%d",&con);
```

```
        consultacadastro(con);
      }
      if (x==6)
      {
        int con;
        printf("Id controle 1, 2 ou 3\n");
        scanf("%d",&con);
        comunicacontrole(con);
      }
      if (x==7)
      {
        int con;
        printf("1- Abrir, 2- Fechar \n");
        scanf("%d",&con);
        if(con ==1)
        {
            escrevergpio(servo, HIGH);
            usleep (2000);
            escrevergpio(servo, LOW);
            usleep (18000);
            escrevergpio(servo, HIGH);
            usleep (2000);
            escrevergpio(servo, LOW);
            escrevergpio(servo, HIGH);
            usleep (2000);
            escrevergpio(servo, LOW);
            usleep (18000);
            escrevergpio(servo, HIGH);
            usleep (2000);
            escrevergpio(servo, LOW);


        }
        if(con ==2)
        {
            escrevergpio(servo, HIGH);
            usleep (1000);
            escrevergpio(servo, LOW);
            usleep(19000);
            escrevergpio(servo, HIGH);
            usleep (1000);
            escrevergpio(servo, LOW);

        }
      }
    }
}
  if (x1 == 2)
  {
      int bd0,controle2;
      printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
      scanf("%d",&bd0);
      baudrate(bd0);

      while (1)
      {
        controle2=0;
        lerdados(0);
        decodificar(0);
        controle2 = validapacote();
        if (controle2 == 1)
```

```
{
    comparar();
    acaofila();
    usleep(1000000);//1s
}

ler();
usleep(1000000);//1s

if ((identidade == 0x3) && (funcao == 0x3))
{
    printf ("Cadastrar controle\n");
    controle2=0;
    if (endlow ==1)
    {
      while (controle2 !=1)
      {
          lerdados(1);
          decodificar(1);
          controle2 = validapacote();
      }
      cadastrar(1);
      comunicacontrole(1);
      escreverendlow = 0x1;
      escreverdados1 = idcontrole1;
      escreverdados0 = capacidade1;
      controle2=0;
    }
    if (endlow == 2)
    {
      while (controle2 !=1)
      {
          lerdados(1);
          decodificar(1);
          controle2 = validapacote();
      }
      cadastrar(2);
      comunicacontrole(2);
      escreverendlow = 0x2;
      escreverdados1 = idcontrole2;
      escreverdados0 = capacidade2;
      controle2=0;

    }
    if (endlow == 3)
    {
      while (controle2 !=1)
      {
          lerdados(1);
          decodificar(1);
          controle2 = validapacote();
      }
      cadastrar(3);
      comunicacontrole(3);
      escreverendlow = 0x3;
      escreverdados1 = idcontrole3;
      escreverdados0 = capacidade3;
      controle2=0;
    }

      escreverfuncao = 0x3;
```

```
      escreverendhi  = 0x0;

      escrever();
      limpacomunicacao ();
}

if ((identidade == 0x3) && (funcao == 0x4))
{
    escreverfuncao = 0x4;
    escreverendhi  = 0x0;
    escreverdados1 = 0x0;

    if(endlow == 0)
    {
      printf("Desligar Receptor controle\n");
      escreverdados0 = 0x0;
      escreverendlow = 0x0;

      escrevergpio(vccsens,LOW);
    }
    if(endlow == 1)
    {
      printf("Ligar Receptor controle\n");
      escreverdados0 = 0x1;
      escreverendlow = 0x1;

      escrevergpio(vccsens,HIGH);
    }

    escrever();
    limpacomunicacao ();
}

if ((identidade == 0x3) && (funcao == 0x5))
{
    printf("controle de fila\n");
    escreverfuncao = 0x5;
    escreverendhi  = 0x0;
    escreverdados1 = 0x0;

    if(endlow == 0)
    {
      printf("Proibir todas filas\n");
      controlefila(0,0);
      escreverdados0 = 0x0;
      escreverendlow = 0x0;

    }
    if(endlow == 1)
    {
      printf("Liberar Principal\n");
      controlefila(1,0);
      escreverdados0 = 0x1;
      escreverendlow = 0x1;

    }
    if(endlow == 2)
    {
      printf("Liberar Secundaria\n");
      controlefila(0,1);
      escreverdados0 = 0x2;
```

```
                escreverendlow = 0x2;

            }
            escrever();
            limpacomunicacao ();
        }

        if ((identidade == 0xf) && (funcao == 0xf))
        {
            printf("BroadCast\n");
            escreverfuncao = 0xf;
            escreverendhi  = 0xf;
            escreverendlow = 0xf;
            escreverdados1 = 0xf;
            escreverdados0 = 0xf;

            escrever();
            limpacomunicacao ();
        }

        if ((identidade == 0x3) && (funcao == 0x1))
        {
            printf("Interferencia Magnetica\n");
            escreverfuncao = 0x1;
            escreverendhi  = 0xa;
            escreverendlow = 0xa;
            escreverdados1 = 0xa;
            escreverdados0 = 0xa;

            escrever();
            limpacomunicacao ();
        }

      }//while
    }//if rodar programa
}// main
```

Estação monitoramento moenda

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>

#define IN 0
#define OUT 1
#define LOW 0
#define HIGH 1

#define DADO 17

int range11,range12,range21, range22, range31, range32;
int soma,status;
```

```
//comunicacao

//ler
int identidade,endhi,endlow,funcao, dados1,dados0;

//escrever
char  escreverfuncao, escreverendhi,escreverendlow,
escreverdados1,escreverdados0;

static int criargpio (int pin)
{
    #define buffer_max 3
    int fd;
    char buffer[buffer_max];
    ssize_t byte_writen;

    fd = open("/sys/class/gpio/export", O_WRONLY);
    byte_writen = snprintf(buffer, buffer_max,"%d", pin);
    write(fd, buffer, byte_writen);

    close (fd);
    return (0);
}

static int direcaogpio (int pin, int dir)
{
    static const char s_directions_str[] = "in\0out";
    #define direction_max 35
    char path[direction_max];
    int fd;

    snprintf(path, direction_max, "/sys/class/gpio/gpio%d/direction", pin);
    fd = open (path, O_WRONLY);

    if (-1 == write(fd, &s_directions_str[IN == dir ? 0:3], IN == dir ?
2:3)){
      return (-1);}
    close(fd);
    return (0);
}

static int lergpio(int pin)
{
    #define valor_max 30
    char path[valor_max];
    char valor_str[3];
    int fd;

    snprintf(path, valor_max,"/sys/class/gpio/gpio%d/value", pin);
    fd = open(path, O_RDONLY);
    if (-1 == read(fd, valor_str,3)){
    return (-1);}
    close (fd);
    return (atoi(valor_str));
}

static int escrevergpio(int pin, int value)
{
    static const char s_valor_str[] = "01";
    char path [valor_max];
    int fd;
```

```
    snprintf(path, valor_max, "/sys/class/gpio/gpio%d/value", pin);
    fd = open (path, O_WRONLY);

    if(1 != write(fd, &s_valor_str[LOW == value ? 0 : 1],1)){
      return(-1);}
    close(fd);
    return (0);
}

void lerdados ()
{

    int i1, amostra[10000];
    i1 = 0;
    soma =0;
    while (i1 !=10000)
    {
        amostra[i1] = lergpio(DADO);
        i1++;
    }

    for (i1=0; i1<= 10000; i1++)
    {
      if( amostra[i1] == 1)
      {
          soma++;
      }
    }

printf("Amostra = %d\n", soma);

}

void padraodados (int pd)
{
    int i,acc,media,x;
    i=media=acc=0;

    for (i=0; i <= 10; i++)
    {
      lerdados();
      acc = acc + soma;
    }

    media=acc/10;

    if ( pd == 1)
    {
      range11 = media - 500;
      range12 = media + 500;
      printf("%d -- %d \n",range11, range12);
      pd = 0;
      printf("Gravou padrÃ£o 1 \n");
    }
    if ( pd == 2)
    {
      range21 = media - 800;
      range22 = media + 800;
      printf("%d -- %d \n",range21, range22);
      printf("Gravou padrÃ£o 2 \n");
```

```
      pd = 0;
    }
    if ( pd == 3)
    {
      range31 = media - 800;
      range32 = media + 800;
      printf("%d -- %d \n",range31, range32);
      printf("Gravou padrÃ£o 3 \n");
      pd = 0;
    }
}

void comparar ()
{
    int aux =0;
    lerdados();
    if ( (range11 <= soma)&& (soma <= range12))
    {
      aux =1;
      status = 1;
      printf ("padrao 1\n");
    }
    if ( (range21 <= soma)&& (soma <= range22))
    {
      aux =1;
      status =2;
      printf ("padrao 2\n");
    }
    if ( (range31 <= soma)&& (soma <= range32))
    {
      aux =1;
      status =3;
      printf ("padrao 3\n");
    }
    if (aux == 0)
    {
      status = 0;
      printf("Nenhum padrao\n");
    }
}
void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);

      if (bd == 1)
      {
          printf("Configurado BaudRate em 9600\n");
          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 2)
      {
```

```
            printf("Configurado BaudRate em 19200\n");
            options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
        }

        if (bd == 3)
        {
            printf("Configurado BaudRate em 38400\n");
            options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
        }

        if (bd == 4)
        {
            printf("Configurado BaudRate em 115200\n");
            options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
        }
        options.c_iflag = IGNPAR;
        options.c_oflag = 0;
        options.c_lflag = 0;

        tcflush   (uart0_filestream, TCIFLUSH);
        tcsetattr (uart0_filestream, TCSANOW, &options);
    }

}

void ler ()
{
    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/serial0", O_RDONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
    else{
      unsigned char rx_buffer [400];
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 400);

      if(rx_length <= 0)
      {

      }
      else
      {
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];
          identidade = rx_buffer[0];

          rx_buffer[rx_length] ='\0';

          if(rx_buffer[0] == 0x4);
          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n",
rx_buffer[0], rx_buffer[1],rx_buffer[2],
rx_buffer[3],rx_buffer[4],rx_buffer[5],rx_buffer[6],rx_buffer[7]);
      }
    }
    close (uart0_filestream);
```

```
}

void escrever()
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];
      unsigned char *p_tx_buffer;

    tx_buffer[0] = 0x04; //ID
    tx_buffer[1] = escreverfuncao;
    tx_buffer[2] = escreverendhi;
    tx_buffer[3] = escreverendlow;
    tx_buffer[4] = escreverdados1;
    tx_buffer[5] = escreverdados0;
    tx_buffer[6] = 0xf; //crc
    tx_buffer[7] = 0xf; //crc

    printf ("send:  %x | %x | %x%x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0],14);

    if (count<0)
    {
      printf ("problema na transmissao\n");
    }
    close (uart0_filestream);
}

void limpacomunicacao ()
{
    escreverfuncao  = 0;
    escreverendhi   = 0;
    escreverendlow  = 0;
    escreverdados1  = 0;
    escreverdados0  = 0;
    funcao = 0;
    endhi  = 0;
    endlow = 0;
    dados1 = 0;
    dados0 = 0;
    identidade = 0;
}

int main ()
{
    int x,x1,pd0;
    criargpio (DADO);
    direcaogpio (DADO,IN);

    printf ("\n\tEstacao 4 - Monitoramento Moenda \n\n");
    printf ("1- Menu de teste, 2- Rodar programa \n");
    scanf ("%d", &x1);

    if (x1 == 1)
```

```
 {
   while (x!=4){
         printf("1-Ler Dados, 2-Padronizar, 3-Comparar, 4-Sair\n");
         scanf("%d", &x);
         if (x==1)
         {
             lerdados();
         }
         if (x==2)
         {
             printf("Gravar padrao 1, 2 ou 3 \n");
             scanf ("%d", &pd0);
             padraodados(pd0);
         }
         if (x==3)
         {
             comparar();
         }
   }
}

if (x1 == 2)
{
   int bd0;
   printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
   scanf("%d",&bd0);
   baudrate(bd0);

   while (1)
   {
         ler();
         pd0 =0;
         usleep(1000);
         if ((identidade == 0x4) && (funcao == 0x2))
        {
               printf("Status...\n");
               comparar();
               escreverfuncao  = 0x2;
               escreverendhi   = 0x0;
               escreverendlow  = 0x0;
               escreverdados1  = 0x0;
               escreverdados0  = status;

               escrever();
               limpacomunicacao ();
         }

         if ((identidade == 0x4) && (funcao == 0x3))
        {
               printf("Gravar padrão...\n");
               pd0 = endlow;
               padraodados(pd0);
               escreverfuncao  = 0x3;
               escreverendhi   = 0x0;
               escreverendlow  = pd0;
               escreverdados1  = 0x0;
               escreverdados0  = 0x0;

               escrever();
               limpacomunicacao ();
         }
```

```
    if ((identidade == 0x4) && (funcao == 0x4))
    {
      printf("Range\n");
      if ( endlow == 1)
      {
          escreverfuncao = 0x4;
          escreverdados1 = range12;
          escreverdados0 = range11;
          escreverendhi  = 0x0;
          escreverendlow = 0x1;
      }
      if ( endlow == 2)
      {
          escreverfuncao = 0x4;
          escreverdados1 = range22;
          escreverdados0 = range21;
          escreverendhi  = 0x0;
          escreverendlow = 0x2;
      }
      if ( endlow == 3)
      {
          escreverfuncao = 0x4;
          escreverdados1 = range32;
          escreverdados0 = range31;
          escreverendhi  = 0x0;
          escreverendlow = 0x3;
      }
          escrever();
          limpacomunicacao ();
    }

      if ((identidade == 0xf) && (funcao == 0xf))
      {
          printf("BroadCast\n");
          escreverfuncao = 0xf;
          escreverdados1 = 0xf;
          escreverdados0 = 0xf;
          escreverendhi  = 0xf;
          escreverendlow = 0xf;

          escrever();
          limpacomunicacao ();
      }

    if ((identidade == 0x4) && (funcao == 0x1))
    {
          printf("Teste interferencia magnetica\n");
          escreverfuncao = 0x1;
          escreverdados1 = 0xa;
          escreverdados0 = 0xa;
          escreverendhi  = 0xa;
          escreverendlow = 0xa;

          escrever();
          limpacomunicacao ();
    }


} //while
```

```
   } //if

}//main
```

7.4 ANEXO D – ESTAÇÃO MONITORAMENTO MOENDA

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>

#define IN 0
#define OUT 1
#define LOW 0
#define HIGH 1

#define DADO 17

int range11,range12,range21, range22, range31, range32;
int soma,status;

//comunicacao

//ler
int identidade,endhi,endlow,funcao, dados1,dados0;

//escrever
char  escreverfuncao, escreverendhi,escreverendlow,
escreverdados1,escreverdados0;

static int criargpio (int pin)
{
    #define buffer_max 3
    int fd;
    char buffer[buffer_max];
    ssize_t byte_writen;

    fd = open("/sys/class/gpio/export", O_WRONLY);
    byte_writen = snprintf(buffer, buffer_max,"%d", pin);
    write(fd, buffer, byte_writen);

    close (fd);
    return (0);
}

static int direcaogpio (int pin, int dir)
{
    static const char s_directions_str[] = "in\0out";
    #define direction_max 35
    char path[direction_max];
    int fd;

    snprintf(path, direction_max, "/sys/class/gpio/gpio%d/direction", pin);
    fd = open (path, O_WRONLY);
```

```
    if (-1 == write(fd, &s_directions_str[IN == dir ? 0:3], IN == dir ?
2:3)){
      return (-1);}
    close(fd);
    return (0);
}

static int lergpio(int pin)
{
    #define valor_max 30
    char path[valor_max];
    char valor_str[3];
    int fd;

    snprintf(path, valor_max,"/sys/class/gpio/gpio%d/value", pin);
    fd = open(path, O_RDONLY);
    if (-1 == read(fd, valor_str,3)){
    return (-1);}
    close (fd);
    return (atoi(valor_str));
}

static int escrevergpio(int pin, int value)
{
    static const char s_valor_str[] = "01";
    char path [valor_max];
    int fd;

    snprintf(path, valor_max, "/sys/class/gpio/gpio%d/value", pin);
    fd = open (path, O_WRONLY);

    if(1 != write(fd, &s_valor_str[LOW == value ? 0 : 1],1)){
      return(-1);}
    close(fd);
    return (0);
}

void lerdados ()
{

    int i1, amostra[10000];
    i1 = 0;
    soma =0;
    while (i1 !=10000)
    {
        amostra[i1] = lergpio(DADO);
        i1++;
    }

    for (i1=0; i1<= 10000; i1++)
    {
      if( amostra[i1] == 1)
      {
          soma++;
      }
    }

printf("Amostra = %d\n", soma);

}
```

```
void padraodados (int pd)
{
    int i,acc,media,x;
    i=media=acc=0;

    for (i=0; i <= 10; i++)
    {
      lerdados();
      acc = acc + soma;
    }

    media=acc/10;

    if ( pd == 1)
    {
      range11 = media - 500;
      range12 = media + 500;
      printf("%d -- %d \n",range11, range12);
      pd = 0;
      printf("Gravou padrÃ£o 1 \n");
    }
    if ( pd == 2)
    {
      range21 = media - 800;
      range22 = media + 800;
      printf("%d -- %d \n",range21, range22);
      printf("Gravou padrÃ£o 2 \n");
      pd = 0;
    }
    if ( pd == 3)
    {
      range31 = media - 800;
      range32 = media + 800;
      printf("%d -- %d \n",range31, range32);
      printf("Gravou padrÃ£o 3 \n");
      pd = 0;
    }
}

void comparar ()
{
    int aux =0;
    lerdados();
    if ( (range11 <= soma)&& (soma <= range12))
    {
      aux =1;
      status = 1;
      printf ("padrao 1\n");
    }
    if ( (range21 <= soma)&& (soma <= range22))
    {
      aux =1;
      status =2;
      printf ("padrao 2\n");
    }
    if ( (range31 <= soma)&& (soma <= range32))
    {
      aux =1;
      status =3;
      printf ("padrao 3\n");
    }
```

```
    if (aux == 0)
    {
      status = 0;
      printf("Nenhum padrao\n");
    }
}
void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);

      if (bd == 1)
      {
          printf("Configurado BaudRate em 9600\n");
          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 2)
      {
          printf("Configurado BaudRate em 19200\n");
          options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 3)
      {
          printf("Configurado BaudRate em 38400\n");
          options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
      }

      if (bd == 4)
      {
          printf("Configurado BaudRate em 115200\n");
          options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
      }
      options.c_iflag = IGNPAR;
      options.c_oflag = 0;
      options.c_lflag = 0;

      tcflush   (uart0_filestream, TCIFLUSH);
      tcsetattr (uart0_filestream, TCSANOW, &options);
    }

}

void ler ()
{
    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/serial0", O_RDONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
```

```
    else{
      unsigned char rx_buffer [400];
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 400);

      if(rx_length <= 0)
      {

      }
      else
      {
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];
          identidade = rx_buffer[0];

          rx_buffer[rx_length] ='\0';

          if(rx_buffer[0] == 0x4);
          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n",
rx_buffer[0], rx_buffer[1],rx_buffer[2],
rx_buffer[3],rx_buffer[4],rx_buffer[5],rx_buffer[6],rx_buffer[7]);
      }
    }
    close (uart0_filestream);

}

void escrever()
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];
      unsigned char *p_tx_buffer;

    tx_buffer[0] = 0x04; //ID
    tx_buffer[1] = escreverfuncao;
    tx_buffer[2] = escreverendhi;
    tx_buffer[3] = escreverendlow;
    tx_buffer[4] = escreverdados1;
    tx_buffer[5] = escreverdados0;
    tx_buffer[6] = 0xf; //crc
    tx_buffer[7] = 0xf; //crc

    printf ("send:  %x | %x | %x%x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0],14);

    if (count<0)
    {
      printf ("problema na transmissao\n");
    }
    close (uart0_filestream);
}
```

```
void limpacomunicacao ()
{
    escreverfuncao  = 0;
    escreverendhi   = 0;
    escreverendlow  = 0;
    escreverdados1  = 0;
    escreverdados0  = 0;
    funcao = 0;
    endhi  = 0;
    endlow = 0;
    dados1 = 0;
    dados0 = 0;
    identidade = 0;
}

int main ()
{
    int x,x1,pd0;
    criargpio (DADO);
    direcaogpio (DADO,IN);

    printf ("\n\tEstacao 4 - Monitoramento Moenda \n\n");
    printf ("1- Menu de teste, 2- Rodar programa \n");
    scanf ("%d", &x1);

    if (x1 == 1)
    {
      while (x!=4){
            printf("1-Ler Dados, 2-Padronizar, 3-Comparar, 4-Sair\n");
            scanf("%d", &x);
            if (x==1)
            {
                lerdados();
            }
            if (x==2)
            {
                printf("Gravar padrao 1, 2 ou 3 \n");
                scanf ("%d", &pd0);
                padraodados(pd0);
            }
            if (x==3)
            {
                comparar();
            }
      }
   }

   if (x1 == 2)
   {
      int bd0;
      printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
      scanf("%d",&bd0);
      baudrate(bd0);

      while (1)
      {
            ler();
            pd0 =0;
            usleep(1000);
            if ((identidade == 0x4) && (funcao == 0x2))
```

```
 {
        printf("Status...\n");
        comparar();
        escreverfuncao  = 0x2;
        escreverendhi   = 0x0;
        escreverendlow  = 0x0;
        escreverdados1  = 0x0;
        escreverdados0  = status;

        escrever();
        limpacomunicacao ();
  }

  if ((identidade == 0x4) && (funcao == 0x3))
 {
        printf("Gravar padrão...\n");
        pd0 = endlow;
        padraodados(pd0);
        escreverfuncao  = 0x3;
        escreverendhi   = 0x0;
        escreverendlow  = pd0;
        escreverdados1  = 0x0;
        escreverdados0  = 0x0;

        escrever();
        limpacomunicacao ();
  }

if ((identidade == 0x4) && (funcao == 0x4))
{
  printf("Range\n");
  if ( endlow == 1)
  {
      escreverfuncao = 0x4;
      escreverdados1 = range12;
      escreverdados0 = range11;
      escreverendhi  = 0x0;
      escreverendlow = 0x1;
  }
  if ( endlow == 2)
  {
      escreverfuncao = 0x4;
      escreverdados1 = range22;
      escreverdados0 = range21;
      escreverendhi  = 0x0;
      escreverendlow = 0x2;
  }
  if ( endlow == 3)
  {
      escreverfuncao = 0x4;
      escreverdados1 = range32;
      escreverdados0 = range31;
      escreverendhi  = 0x0;
      escreverendlow = 0x3;
  }
      escrever();
      limpacomunicacao ();
}

  if ((identidade == 0xf) && (funcao == 0xf))
  {
```

```
                printf("BroadCast\n");
                escreverfuncao = 0xf;
                escreverdados1 = 0xf;
                escreverdados0 = 0xf;
                escreverendhi  = 0xf;
                escreverendlow = 0xf;

                escrever();
                limpacomunicacao ();
              }

           if ((identidade == 0x4) && (funcao == 0x1))
           {
                printf("Teste interferencia magnetica\n");
                escreverfuncao = 0x1;
                escreverdados1 = 0xa;
                escreverdados0 = 0xa;
                escreverendhi  = 0xa;
                escreverendlow = 0xa;

                escrever();
                limpacomunicacao ();
           }


      } //while

   } //if

}//main
```

7.5 ANEXO E – ESTAÇÃO ARMAZENAMENTO E ESTOQUE.

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>
#include <time.h>
#include <math.h>


#define IN 0
#define OUT 1
#define LOW 0
#define HIGH 1

#define TRIG  22
#define ECO   23
#define TAMPA 24

//comunicacao
//ler
int identidade,endhi,endlow,funcao, dados1,dados0;

//escrever
```

```
char  escreverfuncao, escreverendhi,escreverendlow,
escreverdados1,escreverdados0;
double tempoaberto;

//tampa
int segi, segf,mini,minf,hori, horf,horaberto,minaberto,segaberto;
int auxtampa = 0;

//data
int hora, minuto,segundo;

static int criargpio (int pin)
{
    #define buffer_max 3
    int fd;
    char buffer[buffer_max];
    ssize_t byte_writen;

    fd = open("/sys/class/gpio/export", O_WRONLY);
    byte_writen = snprintf(buffer, buffer_max,"%d", pin);
    write(fd, buffer, byte_writen);

    close (fd);
    return (0);
}

static int direcaogpio (int pin, int dir)
{
    static const char s_directions_str[] = "in\0out";
    #define direction_max 35
    char path[direction_max];
    int fd;

    snprintf(path, direction_max, "/sys/class/gpio/gpio%d/direction", pin);
    fd = open (path, O_WRONLY);

    if (-1 == write(fd, &s_directions_str[IN == dir ? 0:3], IN == dir ?
2:3)){
      return (-1);}
    close(fd);
    return (0);
}

static int lergpio(int pin)
{
    #define valor_max 30
    char path[valor_max];
    char valor_str[3];
    int fd;

    snprintf(path, valor_max,"/sys/class/gpio/gpio%d/value", pin);
    fd = open(path, O_RDONLY);
    if (-1 == read(fd, valor_str,3)){
    return (-1);}
    close (fd);
    return (atoi(valor_str));
}

static int escrevergpio(int pin, int value)
{
```

```
    static const char s_valor_str[] = "01";
    char path [valor_max];
    int fd;

    snprintf(path, valor_max, "/sys/class/gpio/gpio%d/value", pin);
    fd = open (path, O_WRONLY);

    if(1 != write(fd, &s_valor_str[LOW == value ? 0 : 1],1)){
      return(-1);}
    close(fd);
    return (0);
}


void medirdist ()
{
    float velo_som =  340.29;

    double time_viagem, distancia=0,dist=0,pcent=0;
    clock_t t0, tf;

    criargpio(TRIG);
    criargpio(ECO);
    direcaogpio(TRIG, OUT);
    direcaogpio(ECO, IN);

          escrevergpio(TRIG, LOW);
          usleep(1000000);
          escrevergpio(TRIG, HIGH);
          usleep(10);
          escrevergpio(TRIG, LOW);

          while (lergpio(ECO) == LOW)
          {
            t0 = clock();
          }
          while (lergpio(ECO) == HIGH)
          {
            tf = clock();
          }

          time_viagem = ((double)(tf-t0))/CLOCKS_PER_SEC;
          distancia = (time_viagem * velo_som)/2;

          dist = ((0.16 - distancia)*100);
          int dist2 = dist;
          pcent =((dist/16)*100);

          printf (" Altura do produto: %d\n %.2f %% utilizado do
tonel\n",dist2,pcent);

      if ( endhi == 0x0 && endlow == 0x1)
      {
          escreverdados1 = 0x0;
          escreverdados0 = dist2;
          escreverendhi  = 0x0;
          escreverendlow = 0x1;
      }
      if ( endhi == 0x0 &&  endlow == 0x2)
      {
          int pcent1 = pcent;
```

```
          escreverdados1 = 0x0;
          escreverdados0 = pcent1;
          escreverendhi  = 0x0;
          escreverendlow = 0x2;
      }
}

void positampa ()
{
    if ((lergpio(TAMPA) == HIGH) && (auxtampa !=1))
    {
      horaedata();
      segi = segundo;
      mini = minuto;
      hori = hora;
      auxtampa=1;
    }
    if ((lergpio(TAMPA) == LOW)&&(auxtampa !=0))
    {
      auxtampa =0;
      horaedata();
      segf = segundo;
      minf = minuto;
      horf = hora;

      horaberto = abs(horf - hori);
      minaberto = abs(minf - mini);
      segaberto = abs(segf - segi);
      printf ("Tempo de escoamento
%d:%d:%d\n",horaberto,minaberto,segaberto);

    }
}

int horaedata ()
{
    struct tm*local;
    time_t t;
    t = time (NULL);
    local = localtime(&t);

    hora    = local->tm_hour;
    minuto  = local->tm_min;
    segundo = local->tm_sec;
}

void baudrate (int bd)
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
    else
    {
      struct termios options;
      tcgetattr(uart0_filestream, &options);

      if (bd == 1)
      {
```

```
            printf("Configurado BaudRate em 9600\n");
            options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
        }

        if (bd == 2)
        {
            printf("Configurado BaudRate em 19200\n");
            options.c_cflag = B19200 | CS8 | CLOCAL | CREAD;
        }

        if (bd == 3)
        {
            printf("Configurado BaudRate em 38400\n");
            options.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
        }

        if (bd == 4)
        {
            printf("Configurado BaudRate em 115200\n");
            options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
        }
        options.c_iflag = IGNPAR;
        options.c_oflag = 0;
        options.c_lflag = 0;

        tcflush   (uart0_filestream, TCIFLUSH);
        tcsetattr (uart0_filestream, TCSANOW, &options);
    }

}

void ler ()
{
    int uart0_filestream = -1;
    uart0_filestream = open ("/dev/serial0", O_RDONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro conexao uart0\n");
    }
    else{
      unsigned char rx_buffer [400];
      int rx_length = read (uart0_filestream, (void*) rx_buffer, 400);

      if(rx_length <= 0)
      {

      }
      else
      {
          funcao = rx_buffer[1];
          endhi  = rx_buffer[2];
          endlow = rx_buffer[3];
          dados1 = rx_buffer[4];
          dados0 = rx_buffer[5];
          identidade = rx_buffer[0];

          rx_buffer[rx_length] ='\0';

          if(rx_buffer[0] == 0x5);
```

```
          printf ("Recebeu:  %x | %x | %x%x | %x%x | %x%x \n",
rx_buffer[0], rx_buffer[1],rx_buffer[2],
rx_buffer[3],rx_buffer[4],rx_buffer[5],rx_buffer[6],rx_buffer[7]);
      }
    }
    close (uart0_filestream);

}

void escrever()
{
    int uart0_filestream = -1;
    uart0_filestream = open("/dev/serial0", O_WRONLY | O_NOCTTY |
O_NDELAY);
    if (uart0_filestream == -1)
    {
      printf("Erro abrir UART0\n");
    }
      unsigned char tx_buffer [400];
      unsigned char *p_tx_buffer;

    tx_buffer[0] = 0x05; //ID
    tx_buffer[1] = escreverfuncao;
    tx_buffer[2] = escreverendhi;
    tx_buffer[3] = escreverendlow;
    tx_buffer[4] = escreverdados1;
    tx_buffer[5] = escreverdados0;
    tx_buffer[6] = 0xf; //crc
    tx_buffer[7] = 0xf; //crc

    printf ("send:  %x | %x | %x %x | %x%x | %x%x \n", tx_buffer[0],
tx_buffer[1],tx_buffer[2],
tx_buffer[3],tx_buffer[4],tx_buffer[5],tx_buffer[6],tx_buffer[7]);
    int count = write(uart0_filestream, &tx_buffer[0],14);


    if (count<0)
    {
      printf ("problema na transmissao\n");
    }
    close (uart0_filestream);
}


int main ()
{
    int x1,x,bd0;

    printf ("\n\t Estacao 5 - Estoque Produção \n\n");
    printf ("1- Menu de teste, 2- Rodar programa \n");
    scanf ("%d", &x1);
    criargpio (TAMPA);
    direcaogpio (TAMPA, IN);

    if (x1 == 1)
    {
      while (x!=6){
          printf("1-Medir Distancia, 2-Tampa, 3- BaudRate, 4- lerRx, 5-
Escrever, 6- Sair\n");
          scanf("%d", &x);
          if (x==1)
```

```
      {
        medirdist ();
      }

      if (x==2)
      {
        positampa ();
      }

      if (x == 3)
      {
        int bd0;
        printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
        scanf("%d",&bd0);
        baudrate(bd0);
      }
      if (x==4){
          ler ();}
      if (x==5){
          escrever ();}
  }
}

if (x1 == 2)
{
  int bd0;
  printf("1- 9600, 2- 19200, 3- 38400, 4- 115200\n");
  scanf("%d",&bd0);
  baudrate(bd0);

  while (1)
  {
      ler ();
      usleep(1000);
      positampa();

      if ((identidade == 0x05) && (funcao == 0x02))
      {
        printf("Medir tanque...\n");
        medirdist();
        escreverfuncao = 02;
        escrever();
      }
      if ((identidade == 0x05) && (funcao == 0x03))
      {
        escreverfuncao = 0x3;
        printf("Escoamento de aÃ§ucar: ");
        if ( endhi == 0x0 &&  endlow == 0x1)
        {
            if (( lergpio(TAMPA) == HIGH) )
            {
              printf("Em escoamento!\n");
              escreverdados1 = 0x0;
              escreverdados0 = 0x1;
              escreverendhi  = 0x0;
              escreverendlow = 0x1;
            }
            else
            {
              printf("Nao esta em escoamento!\n");
              escreverdados1 = 0x0;
```

```
                escreverdados0 = 0x0;
                escreverendhi  = 0x0;
                escreverendlow = 0x1;
              }
          }
          if ( endhi == 0x0 &&  endlow == 0x2)
          {
              escreverdados1 = minaberto;
              escreverdados0 = segaberto;
              escreverendhi  = 0x0;
              escreverendlow = 0x2;
              printf("%d:%d\n", minaberto,segaberto);
          }
          escrever();
        }
        if ((identidade == 0xf) && (funcao == 0xf))
          {
              printf("BroadCast\n");
              escreverfuncao = 0xf;
              escreverdados1 = 0xf;
              escreverdados0 = 0xf;
              escreverendhi  = 0xf;
              escreverendlow = 0xf;

              escrever();
          }
        if ((identidade == 0x5) && (funcao == 0x1))
        {
              printf("Teste interferencia magnÃ©tica\n");
              escreverfuncao = 0x1;
              escreverdados1 = 0xa;
              escreverdados0 = 0xa;
              escreverendhi  = 0xa;
              escreverendlow = 0xa;

              escrever();
        }

      }
    }
}
```