Fausto Humberto dos Reis Filho

Estudo de viabilidade e desenvolvimento de gôndola automatizada para varejo

UBERLÂNDIA

2023

Fausto Humberto dos Reis Filho

Estudo de viabilidade e desenvolvimento de gôndola automatizada para varejo

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Josué Morais

UBERLÂNDIA

2023

Humberto dos Reis Filho, Fausto
Estudo de viabilidade e desenvolvimento de gôndola automatizada para varejo/ **Fausto**. – **UBERLÂNDIA**, **2023**- 28 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Josué Morais

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia – UFU Faculdade de Engenharia Elétrica . **2023**.
Inclui bibliografia.

1. Gondola Automatizada 1. 2. Varejo 2. 3. Gerenciamento de Estoque 3. I. Josué Morais. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

Dedico este TCC a todos que participaram desta jornada. Aos meus pais, Fausto e Claudia, à minha irmã, Ana Laura. Aos meus companheiros de Faculdade e agora de vida Felipe Teles, Alexsander Camilo, Luiz Felipe Bueno, Igor Hasparyk, Elcio Rezende, Lincoln Tarsio, Henrique Augusto, Pedro Mariano, Vitor Mariano e Valdemir Queiroz. Ao meu irmão de longa data Caio Franco. Para ela que me ajudou a não desistir, Marta Batalini. Àquela que chegou por último, mas nada seria possível sem seu apoio Raquel Perin.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a todos que contribuíram para a realização deste Trabalho de Conclusão de Curso. Em primeiro lugar, quero expressar minha gratidão à minha família, em especial ao meu pai Fausto, minha mãe Claudia e minha irmã Ana Laura. Seu apoio incondicional, encorajamento e sacrifícios fizeram com que eu pudesse cursar a Universidade Federal de Uberlândia e concluir esta etapa importante da minha vida.

Também gostaria de agradecer a todos os amigos que fizeram parte da minha jornada na faculdade, tanto os que conheci desde o início como aqueles que se juntaram ao longo do caminho. Suas amizades foram essenciais para compartilhar conhecimentos, superar desafios e criar memórias inesquecíveis. Agradeço pela camaradagem, apoio mútuo e momentos de descontração que compartilhamos.

Além disso, não posso deixar de mencionar minha namorada, Raquel Perin, cujo apoio e força sempre estiveram presentes desde que estivemos juntos. Seu amor, compreensão e encorajamento constantes me motivaram a persistir e acreditar em mim mesmo. Sou grato por tê-la ao meu lado durante o final dessa jornada acadêmica e em todos os aspectos da minha vida.

Agradeço também aos meus orientadores, professores e demais membros da instituição que forneceram conhecimentos, orientações valiosas e estímulos ao longo dessa jornada acadêmica. Seu apoio e dedicação foram fundamentais para o meu crescimento pessoal e profissional.

Por fim, gostaria de expressar minha gratidão a todos que de alguma forma contribuíram para a conclusão deste trabalho, seja através de conselhos, sugestões, críticas construtivas ou simplesmente estando presentes em minha vida. Agradeço sinceramente a cada um de vocês por fazerem parte dessa conquista e por me impulsionarem a ir além.

Sem o apoio e contribuição de todos mencionados, essa jornada não teria sido possível. Sou imensamente grato e agradeço do fundo do meu coração por tudo que fizeram por mim.

“SIC PARVIS MAGNA –
Grandeza a partir de pequenos começos.”,
(Sir Francis Drake)

Resumo

Uma das grandes dificuldades dos pequenos e médios empreendedores no setor de varejo é o gerenciamento de estoque. Apesar de inúmeros processos existentes para o auxílio desta prática, é muito comum que sejam negligenciadas e pouco efetivos. Por isso, este trabalho tem como objetivo desenvolver um dispositivo que ajude na otimização do gerenciamento de estoque. Este dispositivo é uma gôndola automatizada que conta com um sistema de balança para cada uma das prateleiras. Esta tecnologia visa mensurar a quantidade de produtos presente nas prateleiras e enviar os dados via Modbus TCP para um supervisório desenvolvido no ScadaBR, que conta como principal forma de controle e gerenciamento. No trabalho é possível atestar o funcionamento da comunicação em tempo real e a aquisição de dados valiosos para análise por meio do sistema supervisório. Este trabalho serve como base de estudo para futuras incrementações que possam levar ao desenvolvimento completo da gôndola e testes de campo. Este projeto tem potencial de gerar grande impacto positivo para o desenvolvimento prospero de pequenas e médias empresas no setor de varejo, aumentando a sua produtividade e melhorando a experiência do cliente.

Palavras-chave: Gôndola Automatizada, Varejo, Gerenciamento de estoque.

Abstract

Managing inventory is a major challenge for small and medium-sized retailers. Despite the existence of numerous processes to assist in this practice, they are often neglected and ineffective. Therefore, this study aims to develop a device that aids in optimizing inventory management. The proposed device is an automated shelving unit equipped with a weighing system for each shelf. This technology aims to measure the quantity of products on the shelves and send the data via Modbus TCP to a supervisory system developed in ScadaBR, which serves as the main control and management interface. The study demonstrates the real-time communication functionality and the acquisition of valuable data for analysis through the supervisory system. This work serves as a basis for future enhancements that may lead to the complete development of the automated shelving unit and field testing. The project has the potential to generate a significant positive impact on the successful development of small and medium-sized retail businesses, increasing their productivity and improving the customer experience.

Keywords: Automated Shelf Unit. Retail Business. Inventory Management.

Lista de ilustrações

Figura 1 - Célula de Carga 17
Figura 2 - Ponte de Wheatstone 18
Figura 3 - Módulo HX711 19
Figura 4 - Diagrama de Blocos do módulo HX711 19
Figura 5 - Identificação dos Pinos do Módulo HX711 20
Figura 6 - Microcontrolador ESP8266 22
Figura 7 - Microcontrolador ESP32 23
Figura 8 - Símbolo do ScadaBr 25
Figura 9 - Identificação das Células de Carga 26
Figura 10 - Esquemático das Ligações do Sistema 27
Figura 11 - Impressão 3D dos Suportes de Células de Carga 28
Figura 12 - Sistema da prateleira montado com Ponte de Wheatstone completa 28
Figura 13 - Instalação do Pacote do ESP8266 no IDE Arduino 29
Figura 14 - Seleção do Microcontrolador ESP8266 no IDE Arduino 30
Figura 15 - Programação de teste no IDE Arduino 31
Figura 16 - Resultado na porta Serial da primeira programação 32
Figura 17 - Código do fator de calibração parte 1 32
Figura 18 - Código do fator de calibração parte 2 33
Figura 19 - Resultado na porta Serial para descoberta do fator de calibração 34
Figura 20 - Código Adaptado para comunicação Modbus TCP parte 1 36
Figura 21 - Código Adaptado para comunicação Modbus TCP parte 2 37
Figura 22 - Leitura da porta serial para captura de IP 38
Figura 23 - Como adicionar um Data Source 39
Figura 24 - Configuração do Data Source 39
Figura 25 - Configuração do Data Point 40
Figura 26 - Como adicionar nova tela no ScadaBR 41
Figura 27 - Configurando a nova tela 41
Figura 28 - Configuração da Tela de Gráficos 42
Figura 29 - Seleção do Data Point no Script para o servidor 42
Figura 30 - Importação do Script para o servidor 43
Figura 31 - Configuração da tela de Alarmes 43
Figura 32 - Acessando a Watch List 44
Figura 33 - Acessando Propriedades do Data Point 45
Figura 34 - Definindo os Alarmes 45
Figura 35 - Configuração de Tratador de Eventos 46
Figura 36 - Tela 1: Layout do Supermercado 48
Figura 37 - Tela 2: Supervisório das Prateleiras da Gôndola 49
Figura 38 - Tela 3: Gráfico de variação de peso medido na prateleira 50
Figura 39 - Dados exportados no Excel 51
Figura 40 - Tela 4: Lista de Alarmes disparados no teste 52
Figura 41 - E-mail automático de alarme do ScadaBR 52
Figura 42 - Erro identificado na porta serial do Arduino IDE 53
Figura 43 - ESP32 sendo utilizada exclusivamente como fonte para HX711 55
Figura 44 - Prateleira confeccionada para o projeto 55

Lista de tabelas

Tabela 1 – Esquema de Ligações entre Células de Carga e HX711.....16
Tabela 2 – Esquema de Ligações entre HX711 e ESP8266.....16

Lista de abreviaturas e siglas

ASCII	*American Standard Code for Information Interchange*
bps	*bits por segundo*
CLP	*Controlador Lógico Programável*
CRC	*Cyclical Redundancy Checking*
Hz	*Hertz - Unidade de medida de frequência*
IHM	*Interface Homem-Máquina*
kbps	*quilobit por segundo*
LD	*Ladder Diagram*
m	*Metros - Unidade de medida de distância*
MHz	Mega Hertz – Unidade de medida de frequência
MUX	Multiplexador
Mv	Mili Volt – Unidade de Medida
POP	Processo Operacional Padrão
RS-232	*Recommended Standard 232*
RS-485	*Recommended Standard 485*
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control and Data Acquisition*
SDK	Software Development Kit
TCP	*Transmission Control Protocol*

Sumário

1 Introdução 14

1.1 Justificativa 14

1.2 Objetivo 15

2 Figuras e Tabelas 16

3 Referencial Teórico 17

3.1 Células de Carga 17

3.2 Ponte de Wheatstone 18

3.3 Módulo HX711 19

3.4 ESP8266 e ESP32 21

3.5 ScadaBR 24

4 Metodologia 26

4.1 Montagem do Sistema 26

4.2 Testes com Células de Carga 29

4.1 Implementação da IHM 35

5 Resultados e Discussões 47

6 Conclusão e Trabalhos Futuros 56

Referências 58

1 Introdução

Através dos anos a tecnologia tem cada vez mais alcançado o cotidiano das pessoas. Munido de um simples smartphone é possível agilizar tarefas que antes poderiam levar muito tempo e evitar desde filas de pagamento em bancos até intermináveis minutos ao telefone com atendentes de telemarketing. Porém, o avanço tecnológico não se limita apenas ao consumidor final. O setor de prestação de serviço também recebe constantemente atualizações e novidades, que aparecem por meio de dispositivos que facilitam os diversos processos e por meio de procedimentos que otimizam o serviço a ser prestado.

O setor de varejo tem cada vez mais se desenvolvido nas questões logísticas e de processos operacionais padrão (POP), em que apenas com uma ferramenta computacional de controle e treinamento de pessoas têm feito grandes melhorias na área. Porém, este setor não recebe grandes novidades na parte de dispositivos tecnológicos, ou quando recebem, a maioria não consegue fazer o uso dela por causa de valores, já que grande parte dos empreendedores possuem comércios pequenos.

Estas parcelas destes micros a médios empreendimentos possuem grande dificuldade na parte logística e não possuem recursos e nem interesse em investir em POPs por acharem complexo ou ter de desprender um grande investimento. Por isso seria viável que houvesse mais dispositivos tecnológicos de custos baixos a razoáveis que ajudassem a resolver todos esses problemas de cultura e que independesse de processos.

1.1 Justificativa

Uma das grandes dificuldades de empreendedores pequenos e médios no setor de varejo é o controle e gerenciamento de estoque. Porque por mais que POPs e ferramentas logísticas sejam empregadas, a cultura de treinamento e rigor em seguir o procedimento é muitas vezes negligenciado, o que faz com que todo o esforço empregado para realizar tais ações sejam em vão. Por isso, a justificativa para este trabalho de conclusão de curso é ajudar no controle e otimização do gerenciamento entre o estoque e a área de vendas de um varejo, com uma solução que não tenha alto custo e não precise desprender muitos recursos em treinamento de pessoas.

1.2 Objetivo

O objetivo deste trabalho é desenvolver um protótipo de uma gôndola (estantes de varejo) que seja capaz de mensurar a quantidade de produtos por prateleira. Com base nos dados gerados pela prateleira, também usar um supervisório para fins de monitoramento no qual consigamos ver a quantidade de produtos em tempo real e que seja possível setar dados, enviar alertas e gerar relatórios.

Além da prototipagem, é também objetivo deste trabalho testar a gôndola em um ambiente de varejo real, a fim de coletar dados concretos acerca do impacto que tal dispositivo pode gerar para o gerenciamento de estoque.

2 Figuras e Tabelas

Tabela 1 – Esquema de Ligações entre Células de Carga e HX711

Célula 1	Célula 2	Célula 3	Célula 4	Pin HX711
Fio Vermelho	-	-	-	E+
Fio Branco	Fio Branco	-	-	-
Fio Preto	-	Fio Preto	-	-
-	Fio Vermelho	-	-	A-
-	Fio Preto	-	Fio Preto	-
-	-	Fio Vermelho	-	A+
-	-	Fio Branco	Fio Branco	-
-	-	-	Fio Vermelho	E-

Fonte: O Autor (2022)

Tabela 2 – Esquema de Ligações entre HX711 e ESP8266

PinOut Módulo HX711	Pin ESP8266
GND	GND
DT	D2
SCK	D3
VCC	3V3

Fonte: O Autor (2022)

3 Referencial Teórico

O projeto contará com alguns conhecimentos relacionados à montagem de Hardware e desenvolvimento de software. A ideia é mensurar a quantidade de produtos por prateleira (da gôndola) por meio do peso, em que deverá ser usado células de carga e diversos equipamentos que são necessários para a sua utilização.

3.1 Células de Carga

Uma célula de carga é um transdutor de força que converte a carga que atua sobre ele em uma saída elétrica mensurável. Embora existam vários tipos, as células de carga baseadas em sensores de deformação e tensão são as mais usadas. Vários tipos de células de carga são utilizados no ambiente laboratorial ou operacional. Quando é preciso converter força em uma saída elétrica mensurável, a célula ou transdutor de carga é a melhor opção.

Hoje em dia, com exceção de determinados laboratórios onde a balança mecânica de precisão ainda é utilizada em maior escala, a célula de carga com extensômetro (ou sensor de deformação) e tensão dominam o setor de pesagem. A célula de carga com sensor de deformação oferece precisão de 0,25% a 0,03% do fundo de escala e são adequadas para praticamente todas as aplicações industriais (OMEGA, 2015).

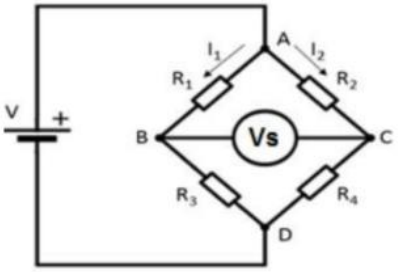
Figura 1 - Célula de Carga

Fonte: Disponível em robocore.net

3.2 Ponte de Wheatstone

A ponte de Wheatstone serve para converter a variação de resistência em uma saída em voltagem que é proporcional à força aplicada. (LIMA, Thiago, 2014). É composta por quatro elementos resistivos, podendo ser montada em três diferentes configurações: um quarto de ponte onde apenas um dos elementos é ativo, meia ponte onde dois elementos são ativos e ponte completa onde todos elementos são ativos (Figura 2). Através do desbalanceamento da ponte que será resultado da variação da resistência dos sensores, temse uma diferença de potencial na saída. Na Figura 2, os pontos A e D são conectados à fonte de alimentação, e os pontos B e C são a saída de sinal da ponte. Há uma variação (na casa de milivolts) de acordo com o desbalanceamento da mesma (MARCELINO, M. L.; MASSA, A.; FLORIAN, F.; BITTENCOURT, J, 2018).

Figura 2 - Ponte de Wheatstone



Fonte: NATIONAL INSTRUMENTS, 2017

Para calcular o valor da saída (Vs) que é um sinal diferencial entre o ponto B e o ponto C, analisa-se cada braço da ponte separadamente. Considerando o primeiro braço com R1 e R3 a voltagem no ponto B será:

$$VB = \frac{(V1.R3)}{R1 + R3}$$

Para o segundo braço com R2 e R4 a voltagem no ponto C será:

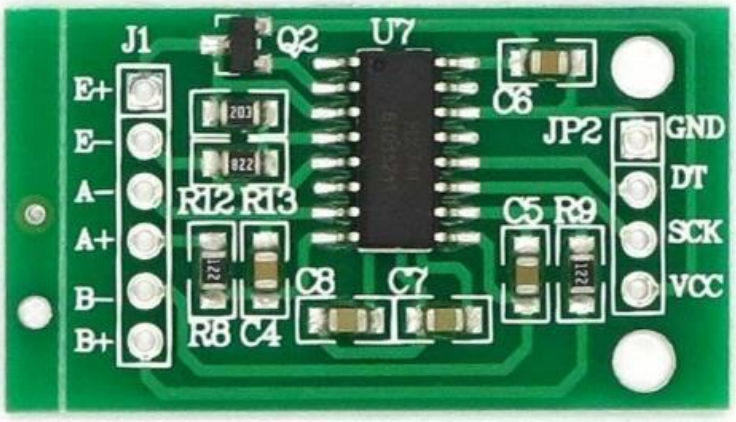
$$VC = \frac{(V1.R4)}{R2 + R4}$$

Fazendo a diferença da voltagem no ponto B com a voltagem do ponto C (VB – VC), tem-se a voltagem de saída (Vs) normalmente medida na ordem de milivolts (mV).

3.3 Módulo HX711

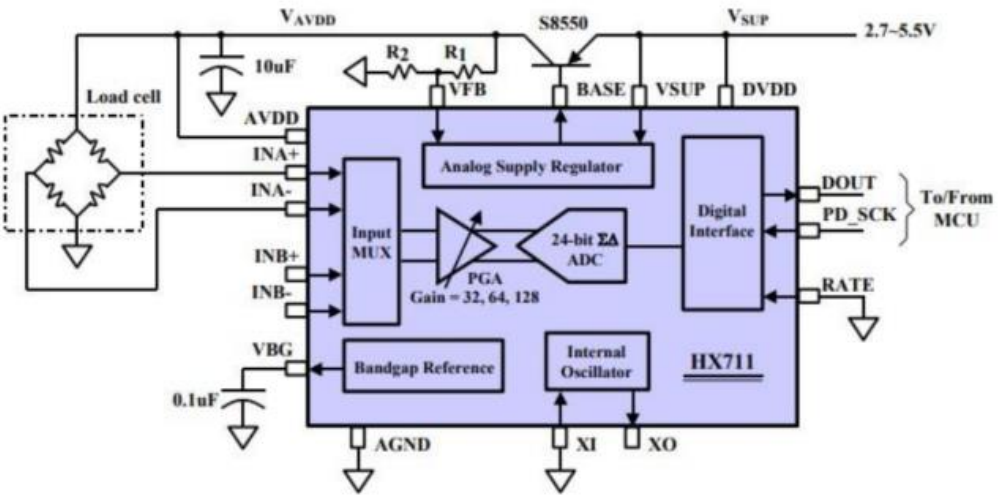
O HX711 é um módulo amplificador operacional de 24 bits que será utilizado para converter o sinal analógico em digital, além de amplificar a saída para que seja reconhecida pelo Arduino. Ele foi criado com a intenção de atuar com sensores de peso ligados em ponte, sendo que o multiplexador (MUX) interno seleciona a entrada diferencial (A+ e A-) ou (B+ e B-) para o amplificador de ganho (LOCATELLI, Caroline, 2019).

Figura 3 - Módulo HX711



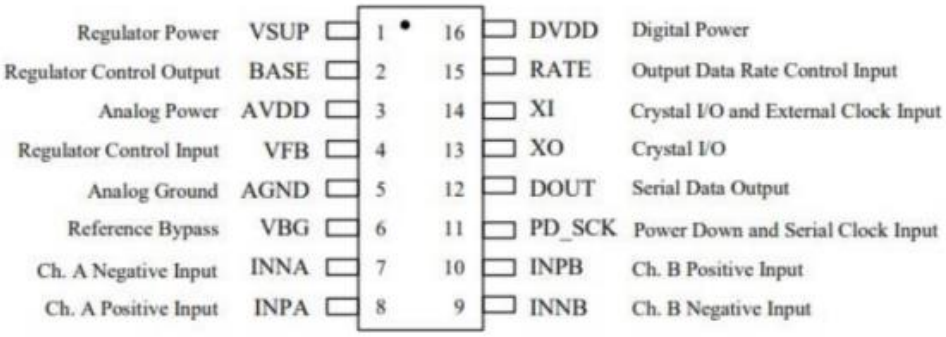
Fonte: Datasheet do fabricante (Avia Semiconductor)

Figura 4 - Diagrama de Blocos do módulo HX711



Fonte: Datasheet do fabricante (Avia Semiconductor)

Figura 5 - Identificação dos Pinos do Módulo HX711



Fonte: Datasheet do fabricante (Avia Semiconductor)

Para a utilização do módulo no arduino IDE é necessário a instalação de sua biblioteca e ter o conhecimento de duas de suas funções principais:

Função Scale:

A função "scale" do módulo HX711 no Arduino é utilizada para converter o valor bruto de leitura obtido pelo sensor de carga em uma escala ou unidade de medida significativa, como quilogramas (kg) em uma balança.

O módulo HX711 é comumente utilizado em balanças e sistemas de pesagem, e é capaz de ler a variação de tensão gerada pelo sensor de carga. No entanto, essa leitura é fornecida como um valor bruto, geralmente em forma de contagem ou bits.

A função "scale" é responsável por aplicar uma escala ao valor bruto, de acordo com a calibração específica da balança. Através dessa calibração, é possível estabelecer uma relação entre a leitura bruta do sensor e uma unidade de medida de peso, como quilogramas ou libras.

Para utilizar a função "scale", é necessário realizar uma calibração prévia da balança. Isso envolve pesar um objeto conhecido com precisão e registrar a leitura bruta obtida pelo sensor nessa condição. Com base nessa leitura e no peso real do objeto, é possível calcular um fator de escala.

O fator de escala é uma constante que é multiplicada pelo valor bruto do sensor para obter a leitura em uma unidade de medida específica. Por exemplo, se o fator de escala for 10, e o valor bruto do sensor for 500, a função "scale" irá retornar o valor de 5000, indicando 5000 gramas ou 5 quilogramas, dependendo da unidade de medida definida.

No código do Arduino, a função "scale" é geralmente aplicada após a leitura do valor bruto do sensor. O resultado é então utilizado para exibir a leitura de peso em um display, enviar para um sistema de controle ou realizar outras operações relevantes.

É importante ressaltar que a função "scale" precisa ser adaptada à calibração específica da balança em questão, levando em consideração as características do sensor de carga utilizado e as unidades de medida desejadas. Uma calibração precisa e correta é fundamental para obter resultados confiáveis e precisos na medição de peso em uma balança feita com Arduino e módulo HX711.

Função Tare:

A função `tare()` no módulo HX711 do Arduino é utilizada para definir o valor atual da leitura como uma leitura de referência, considerando-a como a leitura sem carga. Ela zera a escala do sensor, subtraindo o valor atual da leitura do valor de referência.

A função `tare()` não possui nenhum parâmetro e é chamada no objeto HX711 para realizar a operação de tara. Ao chamar essa função, o valor atual da leitura do sensor será considerado como 0, ou seja, a partir desse momento, as leituras serão relativas a essa leitura de referência sem carga. Isso é útil para calibrar a balança, permitindo que as leituras sejam feitas em relação a um ponto de partida conhecido.

Após chamar `tare()`, ao realizar uma leitura com `scale.read()`, por exemplo, o valor retornado será a diferença entre a leitura atual e o valor de referência estabelecido pela função `tare()`. Dessa forma, é possível obter leituras de peso corretas, levando em consideração a tara definida anteriormente.

3.4 ESP8266 e ESP32

O microcontrolador ESP8266 é um chip amplamente utilizado no campo da Internet das Coisas (IoT) devido às suas capacidades de conectividade Wi-Fi integradas. Ele foi desenvolvido pela Espressif Systems e oferece uma solução de baixo custo e baixo consumo de energia para conectar dispositivos à rede Wi-Fi.

As principais características e funções do ESP8266 incluem:

1. Conectividade Wi-Fi: O ESP8266 possui um módulo Wi-Fi embutido, permitindo que os dispositivos se conectem a redes Wi-Fi existentes e realizem comunicação sem fio.

2. Processamento e memória: Embora seja um microcontrolador, o ESP8266 também possui um processador embutido que permite executar código e processar dados. Ele possui uma quantidade razoável de memória flash para armazenar programas e dados.

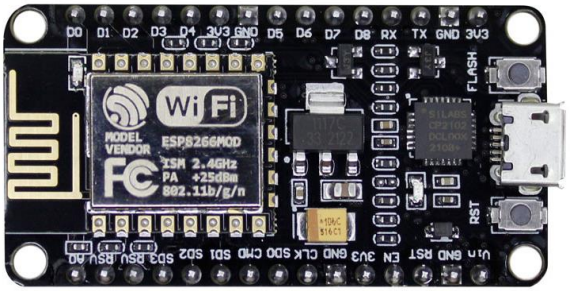
3. Interface GPIO: O ESP8266 possui uma série de pinos de entrada/saída (GPIO - General Purpose Input/Output), que podem ser usados para interagir com outros dispositivos eletrônicos, como sensores, atuadores e outros componentes.

4. Pilha TCP/IP integrada: O ESP8266 tem suporte nativo para uma pilha TCP/IP, o que permite a comunicação pela Internet usando os protocolos IP, TCP e UDP.

5. Protocolos de segurança: O chip ESP8266 oferece suporte a vários protocolos de segurança, como WPA/WPA2, TLS/SSL e criptografia AES, para garantir a segurança das comunicações.

6. Ambiente de desenvolvimento: Existem várias plataformas de desenvolvimento que oferecem suporte ao ESP8266, como a Arduino IDE, a Espressif IoT Development Framework (ESP-IDF) e outras bibliotecas e ferramentas de terceiros.

Figura 6 - Microcontrolador ESP8266



Fonte: WJ Componentes

Agora, em relação ao ESP32, é outro microcontrolador desenvolvido pela Espressif Systems e é considerado o sucessor do ESP8266. Também é considerado como um dos mais robustos e notórios controladores do mercado, tendo como fortes características sua velocidade de processamento, acessibilidade e conectividade, evidenciando essa última principalmente pela sua inteligibilidade com a conexão wi-fi (KOLBAN, 2018).

O ESP32 oferece um conjunto mais amplo de recursos e aprimoramentos em relação ao ESP8266, incluindo:

1. Conectividade Wi-Fi: O ESP32 também possui um módulo Wi-Fi integrado, mas adiciona suporte para Wi-Fi de banda dupla (2,4 GHz e 5 GHz) e capacidade de comunicação em modo estação e ponto de acesso simultaneamente.

2. Bluetooth: O ESP32 possui suporte nativo para Bluetooth, o que o torna mais versátil para aplicações IoT que requerem conectividade Bluetooth, além do Wi-Fi.

3. Mais núcleos de processamento: O ESP32 possui um processador dual-core, permitindo maior capacidade de processamento e multitarefa.

4. Mais memória: O ESP32 oferece mais memória RAM e flash em comparação com o ESP8266, permitindo a execução de aplicativos mais complexos e o armazenamento de mais dados.

5. Interface GPIO: Assim como o ESP8266, o ESP32 possui uma grande quantidade de pinos GPIO para interagir com dispositivos externos.

6. Outros recursos: O ESP32 também possui suporte para interfaces adicionais, como UART, I2C, SPI, ADC e DAC, ampliando as opções de comunicação e conectividade com outros dispositivos.

Apesar das diferenças, tanto o ESP8266 quanto o ESP32 são populares na comunidade de IoT e podem ser

programados usando várias plataformas de desenvolvimento e linguagens de programação, tornando-os acessíveis para projetos de IoT de diferentes escalas e complexidades.

O ESP32 é constituído por um robusto processador, que pode ser visualizado na Figura 7, foi projetado com um modelo que pode ser single ou dual-core de 32- bit (com dois núcleos físicos de processamento) que pode chegar a trabalhar com frequências de clock de até 240 MHz, além de contar com uma vantagem enorme com relação a sua capacidade de armazenamento sendo exponencialmente maior se comparada com a dos microcontroladores Arduíno.

Figura 7 - Microcontrolador ESP32

Fonte: Kolban (2018)

A conectividade do ESP32 é excelente apresentando dois módulos de integração, incorporados ao seu chip, com acesso a redes de transmissão sem fio, através de ondas de rádio, representadas pelo protocolo bluetooth e o wi-fi, se denotando um dos poucos controladores de pequeno porte que reúne todas essas propriedades, as quais geram além de uma maneira econômica de se ter uma plataforma embarcada de alta empregabilidade.

A programação do ESP32 é factível por meio da linguagem de programação C/C++, que pode ser desenvolvida através do programa Sofware devolopment kit (SDK) fornecido pela própria desenvolvedora do microcontrolador ou então pelo Arduíno IDE. Por apresentar uma estruturada conectividade sem fio é possível realizar a programação remota, utilizando por exemplo o wi-fi.

3.5 ScadaBR

O ScadaBR é um software gratuito e de código aberto para o desenvolvimento de aplicações de automação, aquisição de dados e controle supervisório. O ScadaBR destina-se a fornecer todas as funcionalidades dos sistemas SCADA tradicionais.

Esse tipo de software (Supervisory Control and Data Acquisition, sigla em inglês) existe desde o final da década de 1960 e é o alicerce de todos os tipos de aplicações computadorizadas envolvendo máquinas, Controladores Lógico-Programáveis (CLPs), acionamentos eletrônicos e sensores. O SCADA pode ser descrito como o "sistema nervoso central" de um sistema automatizado, monitorando todos os dispositivos e fornecendo acesso organizado aos seus controles e parâmetros.

O público-alvo do ScadaBR são engenheiros de automação de todos os tamanhos que precisam controlar uma variedade de máquinas a partir de seus computadores, executar lógicas de automação ou simplesmente visualizar dados de sensores, ambientes e processos industriais.

A Interface Homem-Máquina (IHM) desempenha um papel crucial na consolidação de projetos industriais e de automação, pois é responsável por fornecer uma interface visual intuitiva e interativa para monitorar e controlar sistemas automatizados. Uma IHM bem projetada no ScadaBR pode trazer diversos benefícios para o sucesso do projeto.

Segundo Groover e Zimmers (2018), a IHM é o ponto de contato entre os operadores e o sistema automatizado, permitindo que informações importantes sejam apresentadas de forma clara e acessível. Por meio de uma IHM bem desenvolvida no SCADA BR, os operadores podem acompanhar o desempenho do sistema em tempo real, visualizar dados de sensores, status de equipamentos e indicadores de processo. Essas informações fornecem uma visão abrangente do funcionamento do sistema, permitindo uma tomada de decisão mais eficaz.

Além disso, a IHM no ScadaBR oferece recursos avançados de controle de processos. Conforme Marques, Afonso e Pimenta (2017) destacam, os operadores podem interagir com o sistema por meio da IHM, executando ações como iniciar ou parar equipamentos, ajustar configurações e definir pontos de controle. Essa

capacidade de controle em tempo real proporciona flexibilidade e capacidade de resposta para otimizar o desempenho do sistema de acordo com as necessidades do momento.

A análise e o histórico de dados também são aspectos importantes que uma IHM no ScadaBR pode oferecer. Conforme Silva e Mendonça (2019) enfatizam, a IHM permite o registro e o armazenamento de dados, possibilitando o acompanhamento de tendências e a análise de padrões ao longo do tempo. Com base nos dados históricos, os operadores podem identificar padrões, falhas recorrentes e oportunidades de melhoria, embasando a tomada de decisões com informações precisas.

Por fim, a IHM no ScadaBR pode ser configurada para fornecer alertas e notificações em tempo real sobre eventos críticos ou condições anormais. Essa funcionalidade, conforme Marques, Afonso e Pimenta (2017) mencionam, permite uma resposta rápida a emergências, minimizando o tempo de inatividade e aumentando a segurança operacional.

Portanto, uma IHM bem projetada no ScadaBR desempenha um papel fundamental na consolidação de projetos industriais e de automação, pois proporciona uma interface amigável e eficiente para monitorar e controlar sistemas automatizados. Ao fornecer informações em tempo real, capacidade de controle, análise de dados e alertas, a IHM no ScadaBR contribui para a otimização do desempenho do sistema e a tomada de decisões embasadas em dados concretos.

Figura 8 - Símbolo do ScadaBr

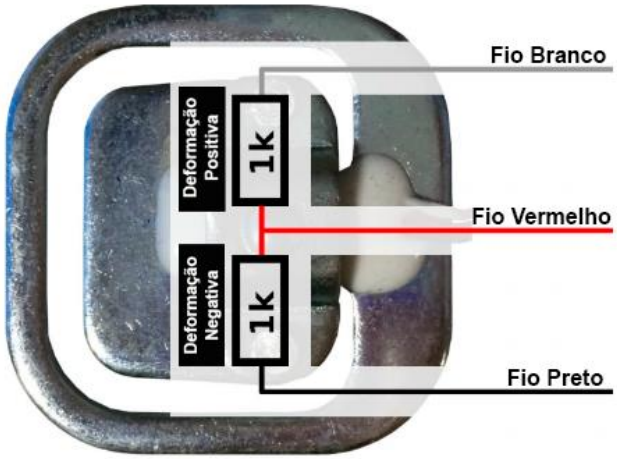
Fonte: github.com/ScadaBR

4 Metodologia

4.1 Montagem do Sistema

Para começar o projeto, primeiro foi preciso entender como as células de carga funcionam e como fazer suas ligações corretamente. A parte importante é identificar o fio do meio, o que pode ser feito com um multímetro simples. O valor de resistência entre os fios externos é o dobro da resistência entre o fio do meio e os fios externos. No meu caso, a resistência entre os fios branco e preto é de 2k, e entre branco e vermelho ou preto e vermelho é de 1k. Não é necessário saber qual fio é para tensão positiva ou negativa. Mas todos os sensores de carga devem ser iguais.

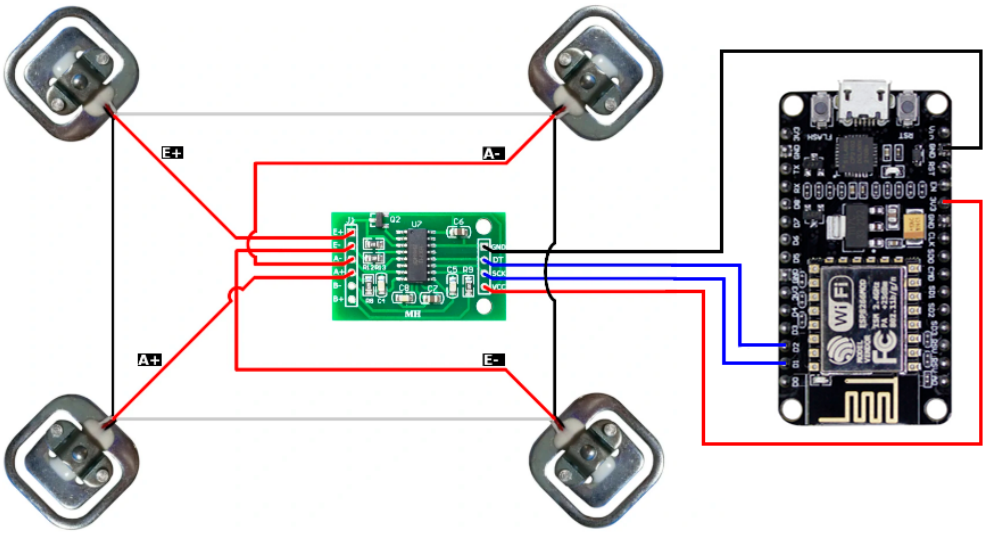
Figura 9 - Identificação das Células de Carga



Fonte: O Autor (2023)

Após a identificação de cada um dos fios, foi necessário aplicar a ponte completa de Wheatstone em associação com o módulo HX711 para a amplificação de sinal. Para isso, produzi um esquemático das ligações incluindo o microcontrolador ESP8266.

Figura 10 - Esquemático das Ligações do Sistema

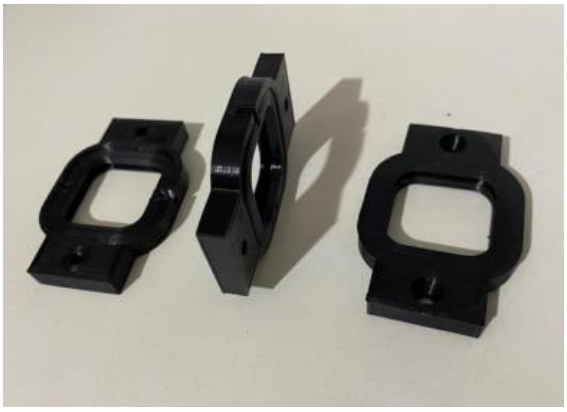


Fonte: O Autor (2023)

Neste esquemático foram associados os fios A+, E+, A-, E- em uma ponte de Wheatstone completa, como descrito no tópico 3.2 do Referencial Teórico. A ponte de Wheatstone completa foi necessária pois além do sistema de balança suportar mais peso, sua precisão e estabilidade aumentam. Foi utilizado quatro células de carga que suportam 50kg de peso cada. Com a associação o peso suportado sobe para 200kg. Os esquemas de ligações estão descritos nas tabelas 1 e 2.

Como a intenção é fazer uma gôndola com um sistema de balança, fez-se necessário fixar as células de carga em cada ponta de uma tábua de madeira que serve de prateleira. A fixação deveria ser feita de forma que cada uma das células ficasse imóvel na madeira, porém com a parte inferior livre para que a deformação seja aplicada e o peso medido. Para isso, a impressão 3D de um suporte para as células desenvolvido pelo usuário “Patrick3345” no site “Thingiverse” foi feito, sendo assim, possível a fixação sem maiores problemas.

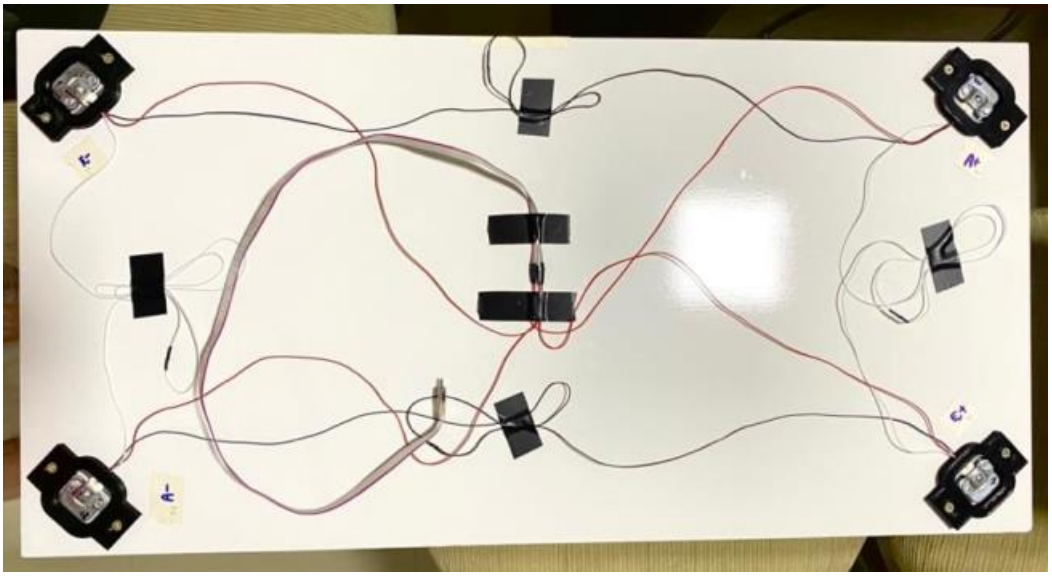
Figura 11 - Impressão 3D dos Suportes de Células de Carga



Fonte: O Autor (2023)

Com todas as peças em mãos, a montagem foi feita. Primeiramente as células de carga foram colocadas no suporte e fixadas na madeira. Depois todas as ligações com os fios foram feitas, inclusive foi usada a técnica de brasagem para conectar os fios das células (como descrito na Figura 10 e tabela 1). A montagem pode ser vista na Figura 12.

Figura 12 - Sistema da prateleira montado com Ponte de Wheatstone completa



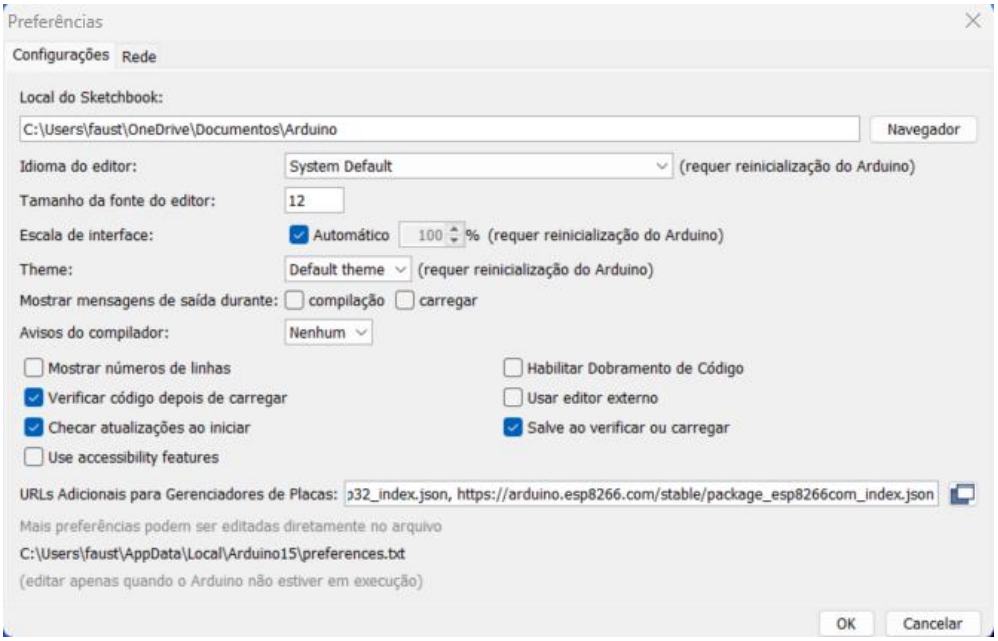
Fonte: O Autor (2023)

O microcontrolador ESP8266 e o módulo amplificador HX711 foram colocados em uma Protoboard para fazer os testes de balança.

4.2 Testes com Células de Carga

Para os testes, foi utilizado o IDE do Arduino. A princípio, realizou-se a instalação das bibliotecas requeridas para o funcionamento do módulo amplificador HX711. Após as instalações, procedeu-se à configuração o IDE do Arduino seguindo alguns passos simples. Primeiramente, abriu-se o IDE do Arduino e acessou-se a opção "Preferências" no menu "Arquivo". Em seguida, foi adicionado uma URL específica no campo "URLs adicionais para Gerenciador de Placas", como ilustrado na Figura 13. Essa URL permitiu a instalação do pacote do ESP8266 e ESP32.

Figura 13 - Instalação do Pacote do ESP8266 no IDE Arduino

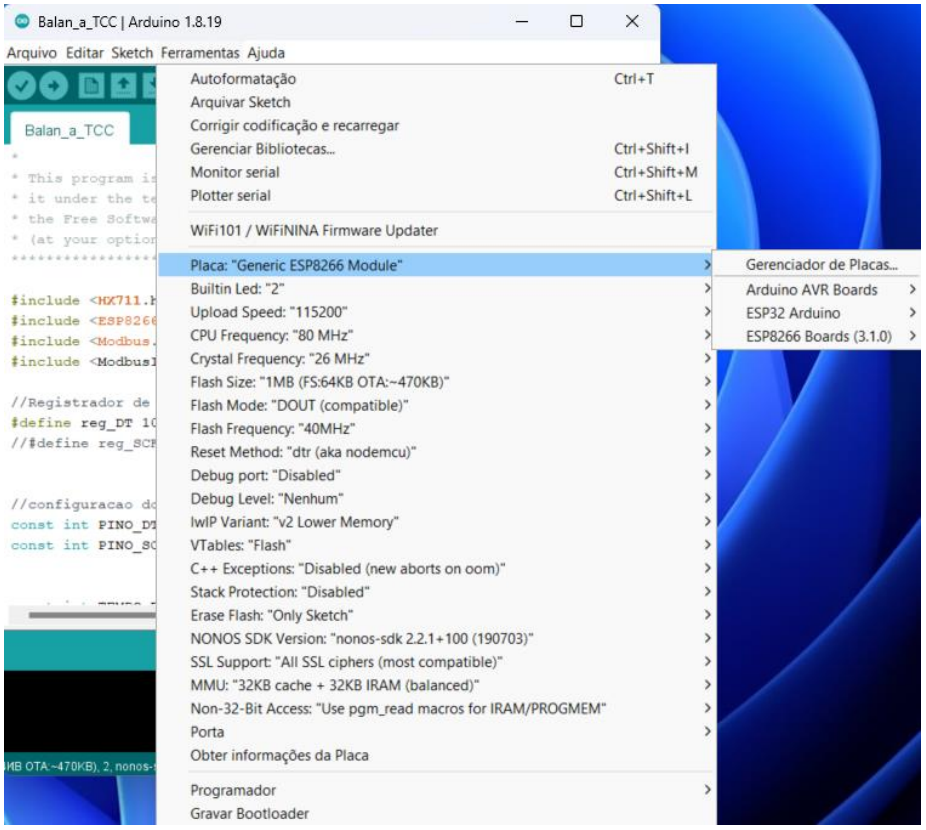


Fonte: O Autor (2023)

Em seguida, dirigiu-se ao menu "Ferramentas" e a opção "Placa" foi selecionado. A partir disso, foi escolhido o "Gerenciador de Placas". Na barra de busca do Gerenciador de Placas, digitou-se "ESP8266" e o pacote "esp8266 by ESP8266 Community" foi instalado.

Após a instalação do pacote, retornou-se ao menu "Ferramentas" e novamente a opção "Placa" foi selecionada. Dessa vez, optou-se pela placa ESP8266 adequada para o módulo em questão, que seria "Generic ESP8266 Module". Em seguida, configurou-se a porta serial correta em "Ferramentas" -> "Porta" e definiu-se a velocidade de comunicação em "Ferramentas" -> "Velocidade da Porta".

Figura 14 - Seleção do Microcontrolador ESP8266 no IDE Arduino

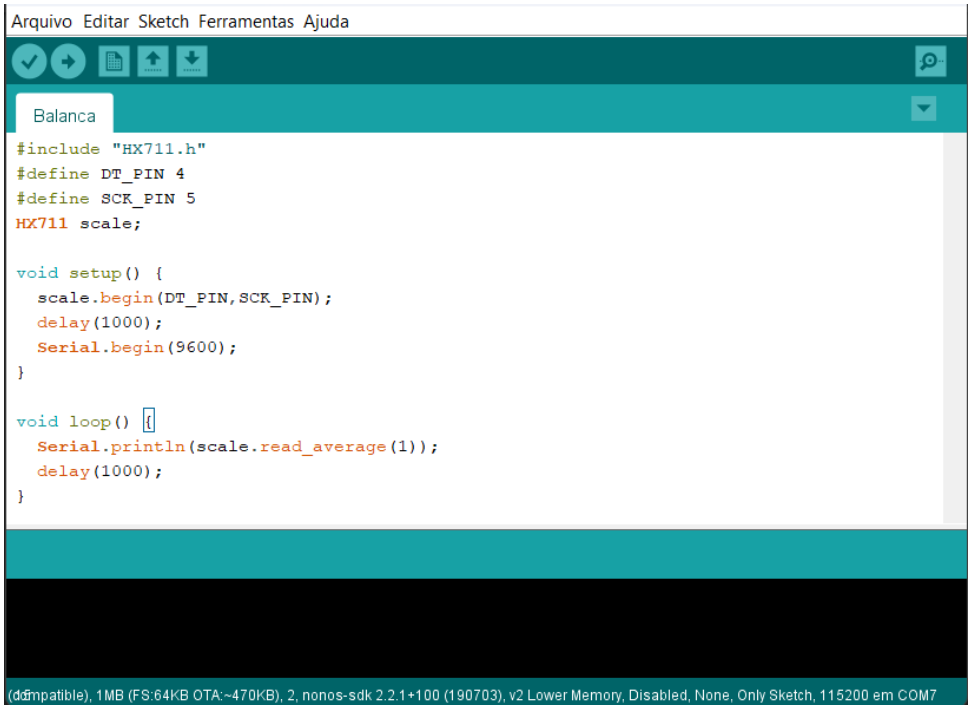


Fonte: O Autor (2023)

Com as configurações devidamente ajustadas, iniciou-se o desenvolvimento do código. Para lidar com a comunicação Wi-Fi, foi necessário incluir a biblioteca “ESP8266WiFi.h”. A partir desse ponto, era possível programar o módulo ESP8266 para realizar várias tarefas no projeto.

Conectou-se o módulo ESP8266 ao computador por meio de uma conexão USB e verificou-se se todas as configurações estavam corretas. Em seguida, elaborou-se um código simples para verificar se as células de carga estavam gerando sinais, como ilustrado na Figura 15. Posteriormente, o código foi compilado e enviado à ESP8266. Durante o processo de upload, observou-se a saída no console do IDE a fim de verificar se não havia erros e se o upload foi bem-sucedido.

Figura 15 - Programação de teste no IDE Arduino



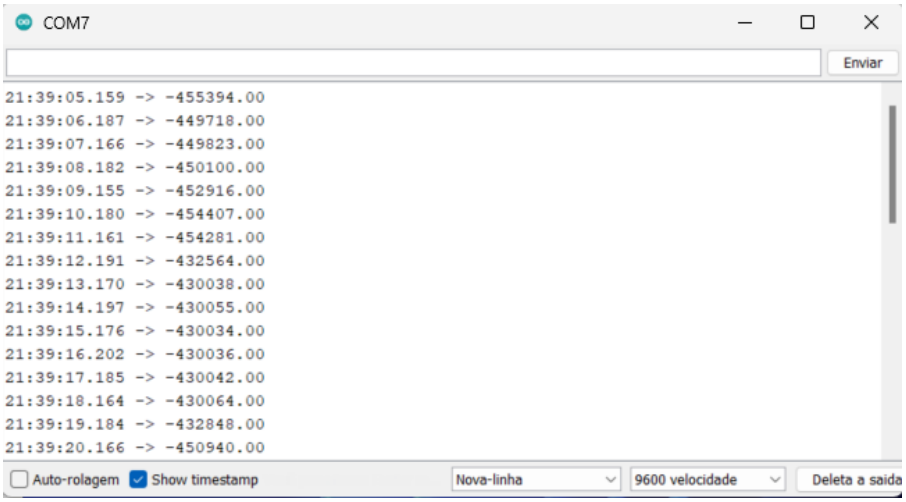
Fonte: O Autor (2023)

O código obteve êxito, porém ainda não era o resultado desejado. Ao observar a porta Serial, foi possível verificar os valores dos sinais enviados pelas células de carga e confirmar seu funcionamento correto. Este código no Arduino utiliza a biblioteca HX711 para ler dados do módulo HX711. Uma das funções mais importantes é a função ‘Scale’.

A função `Serial.println(scale.read_average(1))` lê a média de uma leitura do módulo HX711 utilizando a função `read_average(1)` da biblioteca HX711. Essa função realiza uma leitura do sensor de carga e retorna o valor médio das leituras. O valor médio é então impresso no monitor serial usando a função `println()`.

Portanto, o código lê continuamente a média das leituras do sensor de carga conectado ao módulo HX711 e imprime esse valor no monitor serial a cada segundo. Essas leituras podem ser interpretadas como o peso medido pelo sensor de carga, de acordo com a calibração realizada anteriormente. No entanto, é importante ressaltar que a balança ainda não foi calibrada, sendo esse o próximo passo a ser executado nessa etapa do processo.

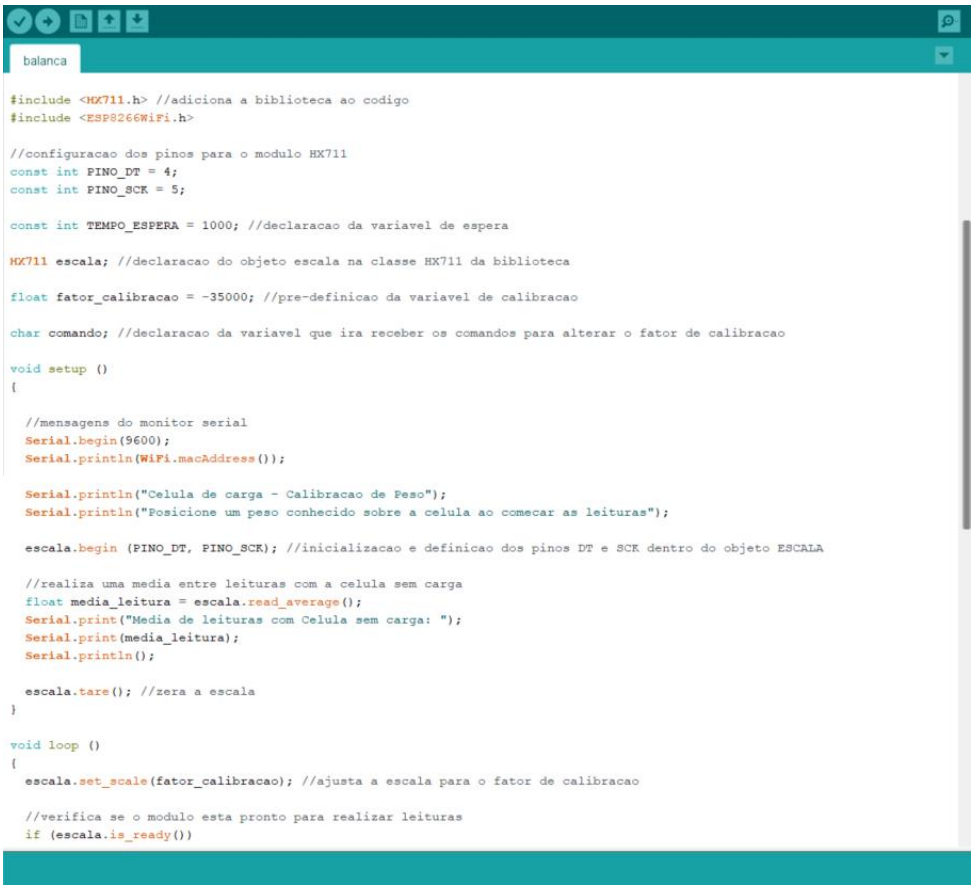
Figura 16 - Resultado na porta Serial da primeira programação



Fonte: O Autor (2023)

Para fazer a calibração do sistema, era preciso descobrir o “Fator de Calibração”. Para isso, produziu-se o código ilustrado na Figura 17 e Figura 18, o qual será descrito a seguir.

Figura 17 - Código do fator de calibração parte 1



```
balanca

#include <HX711.h> //adiciona a biblioteca ao codigo
#include <ESP8266WiFi.h>

//configuracao dos pinos para o modulo HX711
const int PINO_DT = 4;
const int PINO_SCK = 5;

const int TEMPO_ESPERA = 1000; //declaracao da variavel de espera

HX711 escala; //declaracao do objeto escala na classe HX711 da biblioteca

float fator_calibracao = -35000; //pre-definicao da variavel de calibracao

char comando; //declaracao da variavel que ira receber os comandos para alterar o fator de calibracao

void setup ()
{

  //mensagens do monitor serial
  Serial.begin(9600);
  Serial.println(WiFi.macAddress());

  Serial.println("Celula de carga - Calibracao de Peso");
  Serial.println("Posicione um peso conhecido sobre a celula ao comecar as leituras");

  escala.begin (PINO_DT, PINO_SCK); //inicializacao e definicao dos pinos DT e SCK dentro do objeto ESCALA

  //realiza uma media entre leituras com a celula sem carga
  float media_leitura = escala.read_average();
  Serial.print("Media de leituras com Celula sem carga: ");
  Serial.print(media_leitura);
  Serial.println();

  escala.tare(); //zera a escala
}

void loop ()
{
  escala.set_scale(fator_calibracao); //ajusta a escala para o fator de calibracao

  //verifica se o modulo esta pronto para realizar leituras
  if (escala.is_ready())
```

Fonte: O Autor (2023)

Figura 18 - Código do fator de calibração parte 2

```
balanca
if (escala.is_ready())
{
  //mensagens de leitura no monitor serial
  Serial.print("Leitura: ");
  Serial.print(escala.get_units(), 1); //retorna a leitura da variavel escala com a unidade quilogramas
  Serial.print(" kg");
  Serial.print(" \t Fator de Calibracao = ");
  Serial.print(fator_calibracao);
  Serial.println();

//alteracao do fator de calibracao
  if(Serial.available())
    {
      comando = Serial.read();
      switch (comando)
      {
        case 'x':
        fator_calibracao = fator_calibracao - 100;
        break;
        case 'c':
        fator_calibracao = fator_calibracao + 100;
        break;
        case 'v':
        fator_calibracao = fator_calibracao - 10;
        break;
        case 'b':
        fator_calibracao = fator_calibracao + 10;
        break;
        case 'n':
        fator_calibracao = fator_calibracao - 1;
        break;
        case 'm':
        fator_calibracao = fator_calibracao + 1;
        break;
      }
    }
  }
  else
  {
    Serial.print("HX-711 ocupado");
  }
delay(TEMPO_ESPERA);
}
```

Fonte: O Autor (2023)

O código em questão utiliza o módulo HX711 para realizar leituras de uma célula de carga (balança) no Arduino. Ele começa importando a biblioteca `HX711.h` para permitir o uso das funções do módulo. Em seguida, são definidos os pinos utilizados para a comunicação com o módulo HX711. O pino DT é definido como o pino 4 e o pino SCK como o pino 5.

A constante `TEMPO_ESPERA` é declarada com o valor de 1000, que será usado para determinar o tempo de espera entre cada leitura. Em seguida, é declarado um objeto chamado `escala` da classe HX711 para realizar as operações com o módulo.

A variável `fator_calibracao` é pré-definida com o valor médio da leitura do código anterior (-43000), que será utilizado para ajustar a escala do módulo. A variável `comando` é declarada para receber os comandos enviados pelo monitor serial para alterar o fator de calibração.

Na função `setup()`, as mensagens iniciais são impressas no monitor serial para orientação. O objeto `escala` é inicializado com os pinos DT e SCK. Em seguida, é realizada uma leitura média da célula sem carga e essa média é impressa no monitor serial. Por fim, a função `tare()` é chamada para zerar a escala.

Na função `loop()`, o fator de calibração é ajustado chamando a função `set_scale()` do objeto `escala`. É verificado se o módulo está pronto para realizar leituras utilizando a função `is_ready()`.

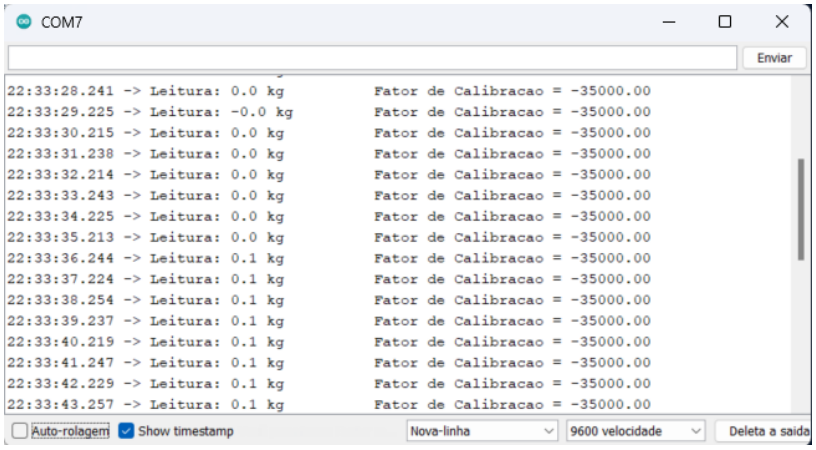
Se o módulo estiver pronto, a leitura atual é obtida usando a função `get_units()`. Essa leitura é impressa no monitor serial junto com a unidade de medida (quilogramas) e o valor do fator de calibração.

O código também permite a alteração do fator de calibração através dos comandos enviados pelo monitor serial (teclas x, c, v, b, n e m). Se algum comando estiver disponível, ele é lido e um switch-case é usado para ajustar o fator de calibração de acordo com o comando recebido.

Se o módulo HX711 não estiver pronto para realizar leituras, a mensagem "HX-711 ocupado" é impressa no monitor serial. Após cada iteração do loop, há um atraso definido pela variável `TEMPO_ESPERA` antes de iniciar a próxima leitura.

Para que a calibração fosse feita de maneira precisa, era necessário ter algum objeto de peso conhecido para ajustar este fator de calibração até que o mesmo atingisse a leitura em kg correta, que neste caso, era 0.1kg. Ao final, pela porta serial, o fator de calibração que havia começado em -43000 subiu para -35000 por meio dos comandos descritos anteriormente.

Figura 19 - Resultado na porta Serial para descoberta do fator de calibração



Fonte: O Autor (2023)

Desta forma, nossa balança estava funcionando corretamente e os testes com as células de carga foram finalizados. Agora poderíamos iniciar a próxima etapa do projeto.

Uma pequena observação: Para cada superfície que a prateleira está, a deformação da célula de carga é diferente. Sendo necessário, assim, um novo fator de calibração.

4.1 Implementação da IHM

Para a implementação da IHM, que será desenvolvida no ScadaBR foi necessário utilizar o protocolo Modbus TCP no ScadaBR em conjunto com o microcontrolador ESP8266 e o módulo HX711. Portanto, foi preciso fazer algumas adaptações no código anterior e incluir as funções necessárias para habilitar a comunicação Modbus TCP.

Para o desenvolvimento do código adaptado apresentado na Figura 20 e Figura . Foram incluídas as bibliotecas necessárias para a comunicação Modbus TCP, como o "ModbusIP_ESP8266.h". Além disso, foi necessário adicionar a biblioteca "ESP8266WiFi.h" para configurar a conexão Wi-Fi.

Os pinos utilizados no ESP8266 (GPIO0 e GPIO2) correspondem aos pinos 4 e 5, respectivamente. Essa informação é importante porque o GPIO0 é usado como pino de dados (DT) e o GPIO2 é usado como pino de clock (SCK) para o módulo HX711. Esses pinos são definidos no código como constantes "PINO_DT" e "PINO_SCK".

No início do código, é realizada a configuração da conexão Wi-Fi, utilizando o nome da rede (SSID) e a senha (password). É possível ajustar esses parâmetros para se conectarem à rede Wi-Fi correta.

Em seguida, há a configuração dos registradores Modbus, usando a função "mb.addIreg(reg_DT)" para adicionar o registrador Modbus analógico que será utilizado para enviar a leitura do HX711. No código desenvolvido, o registrador escolhido é o "reg_DT", que é definido como 10.

No loop principal, a função "mb.task()" é chamada para lidar com as operações de comunicação Modbus TCP. Essa função deve ser chamada regularmente para garantir o funcionamento correto da comunicação.

A leitura da balança usando o módulo HX711 é realizada como no código anterior. Após obter a leitura em gramas, ela é atribuída à variável "DT". Em seguida, a função "mb.Ireg(reg_DT, DT)" é usada para copiar o valor da leitura para o registrador Modbus DT, permitindo que o ScadaBR tenha acesso a esse valor.

No monitor serial, é exibida a leitura atual, o fator de calibração e o valor do registrador "reg_DT" para fins de depuração. Por fim, há um pequeno atraso de 1 segundo no loop para evitar uma alta taxa de atualização dos valores.

É importante mencionar que a utilização dos pinos GPIO0 e GPIO2 no ESP8266 influencia o protocolo Modbus TCP, pois esses pinos são reservados para a comunicação serial interna do módulo. Portanto, é fundamental evitar o uso desses pinos para outras finalidades por causa de possíveis interferências.

Figura 20 - Código Adaptado para comunicação Modbus TCP parte 1

Balan_a_TCC

```
#include <HX711.h> //adiciona a biblioteca ao codigo
#include <ESP8266WiFi.h>
#include <Modbus.h>
#include <ModbusIP_ESP8266.h>

//Registrador de Offsets
#define reg_DT 10
//#define reg_SCK 11


//configuracao dos pinos para o modulo HX711
const int PINO_DT = 4; //GPIO0=D2
const int PINO_SCK = 5; //GPIO2=D1


const int TEMPO_ESPERA = 1000; //declaracao da variavel de espera

HX711 escala; //declaracao do objeto escala na classe HX711 da biblioteca

float fator_calibracao = -21500; //pre-definicao da variavel de calibracao
int DT;

char comando; //declaracao da variavel que ira receber os comandos para alterar o fator de calibracao

//ModbusIP object
ModbusIP mb;

long tempo;

void setup ()
{

  //mensagens do monitor serial
  Serial.begin(9600);

  // Configuração WIFI

   //mb.config("Fausto Algar", "[redacted]"); //coloque aqui os parâmetros de sua rede local
 // mb.config("Fausto Area", "[redacted]"); //coloque aqui os parâmetros de sua rede local
   mb.config("iPhone do Fausto", "[redacted]"); //coloque aqui os parâmetros de sua rede local
  // mb.config("INDUSTRIA", "[redacted]"); //coloque aqui os parâmetros de sua rede local
  //   mb.config("Domotica", "[redacted]"); //coloque aqui os parâmetros de sua rede local


    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("Endereço IP: ");
    Serial.println(WiFi.localIP());

    //Calibração

  Serial.println("Celula de carga - Calibracao de Peso");
  Serial.println("Posicione um peso conhecido sobre a celula ao comecar as leituras");
```

Fonte: O Autor (2023)

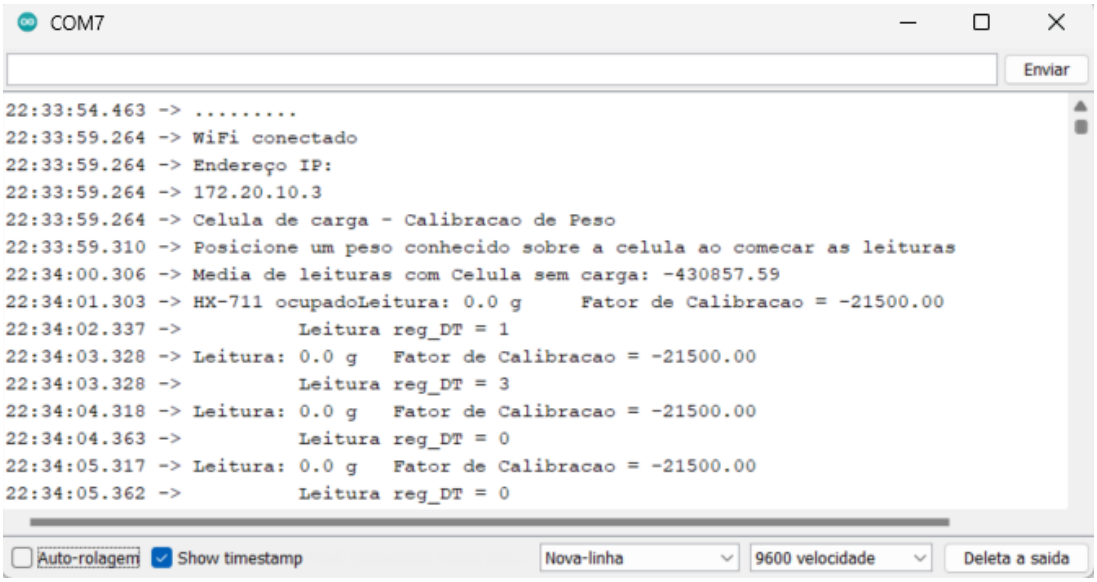
Figura 21 - Código Adaptado para comunicação Modbus TCP parte 2

```
Balan_a_TCC

    //Calibração

  Serial.println("Celula de carga - Calibracao de Peso");
  Serial.println("Posicione um peso conhecido sobre a celula ao comecar as leituras");

  escala.begin (PINO_DT, PINO_SCK); //inicializacao e definicao dos pinos DT e SCK dentro do objeto ESCALA

  //realiza uma media entre leituras com a celula sem carga
  float media_leitura = escala.read_average();
  Serial.print("Media de leituras com Celula sem carga: ");
  Serial.print(media_leitura);
  Serial.println();

  escala.tare(); //zera a escala



    //+-- Configura os registradores Modbus analógicos (2 bytes)
    mb.addIreg(reg_DT);
    tempo = millis();
}

void loop ()
{

    mb.task();

  escala.set_scale(fator_calibracao); //ajusta a escala para o fator de calibracao

  //verifica se o modulo esta pronto para realizar leituras
  if (escala.is_ready())
  {
    //mensagens de leitura no monitor serial
    Serial.print("Leitura: ");
    Serial.print(escala.get_units(), 1); //retorna a leitura da variavel escala com a unidade quilogramas
    DT = (escala.get_units())*1000;
    if (DT < 0)
    {
      DT = 0;
    }
    Serial.print(" g");
    Serial.print(" \t Fator de Calibracao = ");
    Serial.print(fator_calibracao);
    Serial.println();

      mb.Ireg(reg_DT,DT); //+- Copia o valor do AD para o registrador Modbus DT

    Serial.print(" \t Leitura reg_DT = ");
    Serial.print(DT);
    Serial.println();
    }
    else
    {
      Serial.print("HX-711 ocupado");
    }
  delay(1000);

}
```

Fonte: O Autor (2023)

O código foi compilado e carregado para a placa ESP8266. Verificou-se se a comunicação Modbus TCP estava funcionando corretamente. Por meio do código, pela porta serial, obtive o endereço IP que deveria utilizar no ScadaBR, como mostra a Figura 22.

Figura 22 - Leitura da porta serial para captura de IP



Fonte: O Autor (2023)

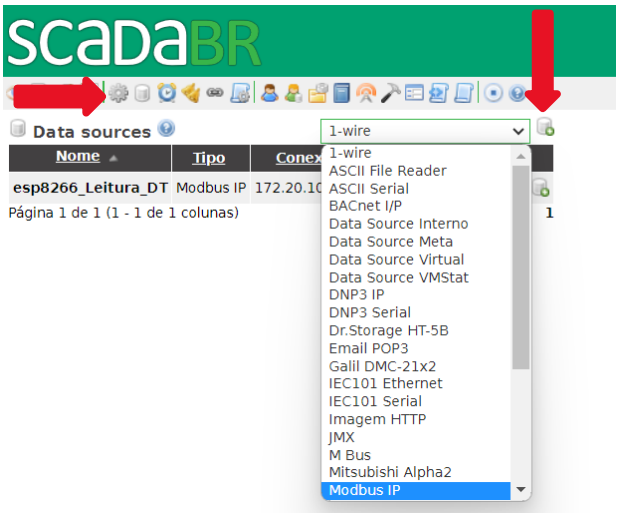
Com tudo funcionando corretamente, agora era hora de criar propriamente a Interface Homem Máquina com o ScadaBR.

Ao abrir o projeto no ScadaBR, o ambiente de trabalho é apresentado. Para prosseguir com essa etapa, a primeira ação necessária foi adicionar um Data Source para receber o sinal enviado pela ESP8266 por meio do protocolo Modbus TCP.

Para adicionar um Data Source Modbus IP, foi necessário clicar na opção “Data Sources” na barra lateral esquerda. Isso abriria a página de configuração dos dispositivos de entrada de dados. Na página “Data Sources”, foi selecionada a opção “Adicionar Novo” para adicionar um novo dispositivo de entrada de dados ao projeto.

Uma janela de configuração foi exibida. Nessa janela, deve-se selecionar “Modbus IP” como o tipo de conexão, pois era desejado fazer uma comunicação com um dispositivo Modbus via TCP/IP.

Figura 23 - Como adicionar um Data Source

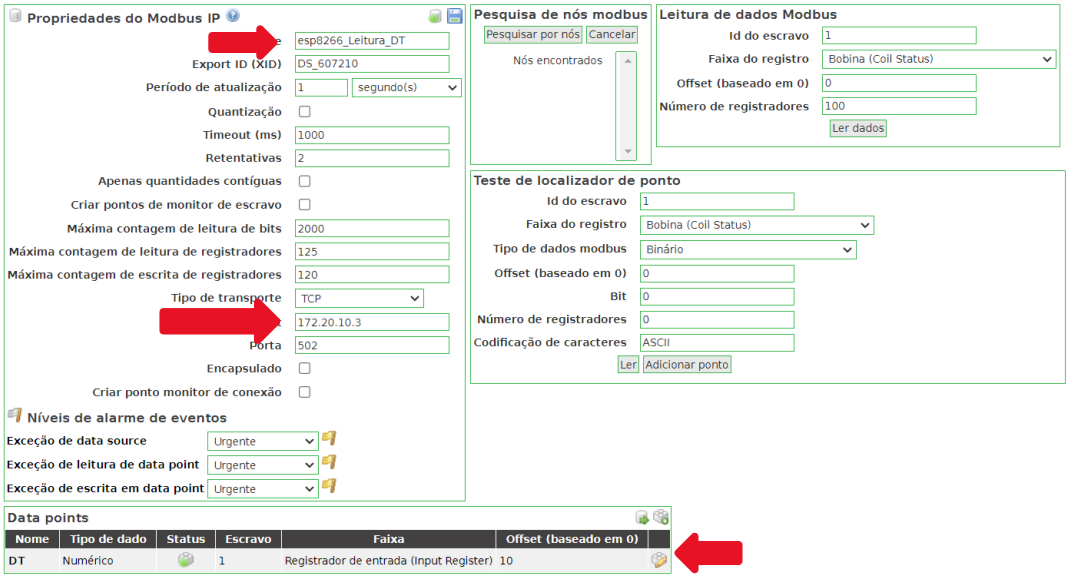


Fonte: O Autor (2023)

Agora, era preciso preencher as informações necessárias na janela de configuração do Data Source Modbus IP. Isso inclui o nome do Data Source, o período de atualização (neste caso de 1 em 1 segundo), quanto é o tempo de Timeout e quantas retentativas serão feitas, o endereço IP do dispositivo Modbus, que foi obtido por meio do código anterior, e a porta utilizada para a comunicação (geralmente a porta padrão é 502 para Modbus TCP).

Depois de preencher as informações, era necessário clicar em “Salvar” para adicionar o Data Source Modbus IP ao projeto.

Figura 24 - Configuração do Data Source



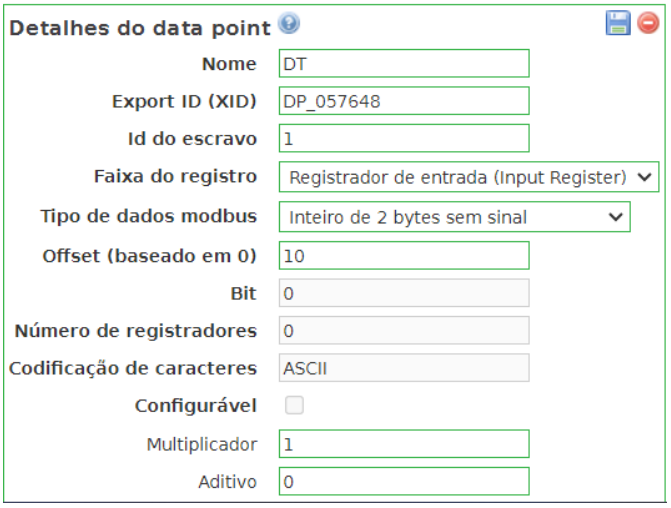
Fonte: O Autor (2023)

Em seguida, foi preciso configurar os Data Points, que representarão os dados desejados que serão recebidos pelo dispositivo Modbus. Na página de configuração do Data Source Modbus IP, na parte inferior (como indicado na Figura 23), foi possível acessar a configuração de Data Point.

O botão “Adicionar Data Point” foi selecionado para criar um Data Point. As informações necessárias foram preenchidas, como nome, endereço do registrador no dispositivo Modbus (Input Register) e o tipo de dado (neste caso um inteiro de 2 bytes).

Após adicionar o Data Point, o botão “Salvar” doi selecionado para confirmar as configurações.

Figura 25 - Configuração do Data Point



Fonte: O Autor (2023)

Neste momento, a criação das telas e elementos gráficos para visualizar os dados recebidos do dispositivo Modbus poderiam ser realizadas. Foram utilizadas as ferramentas de design do ScadaBR para criar telas personalizadas e adicionar elementos como gráficos, medidores, alarmes, etc.

Determinou-se o total de quatro telas para o Supervisório:

1- Uma tela para seleção de gôndola com a planta de um supermercado;

2- Uma tela com a gôndola específica que levará à nossa prateleira testada;

3- Uma tela com um gráfico que mostrará a variação de peso medido pela prateleira;

4- Uma tela que contará com alarmes acionados.

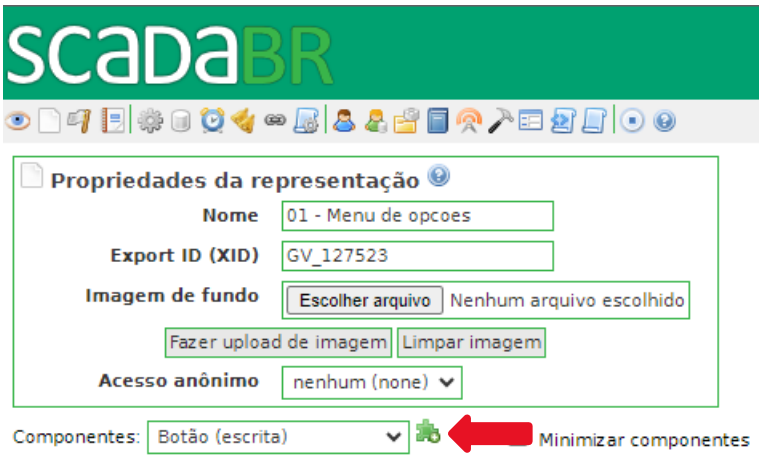
Começando pela primeira tela, foi necessário clicar em “Representação Gráfica” no canto esquerdo e a seguir adicionar uma nova tela clicando em “Nova Representação”, como ilustrado na Figura 26.

Figura 26 - Como adicionar nova tela no ScadaBR

Fonte: O Autor (2023)

Em seguida, abriu-se uma caixa de opções e foi escolhida o nome da nova tela, assim como a imagem de fundo de Layout de supermercado utilizada. Nesta tela também, ao clicar em “Componentes” e em seguida adicioná-lo (como ilustrado na Figura 27), foi possível colocar botões para mudar de tela. Neste caso foi adicionado um botão em uma das gôndolas para poder mudar para a tela 2.

Figura 27 - Configurando a nova tela

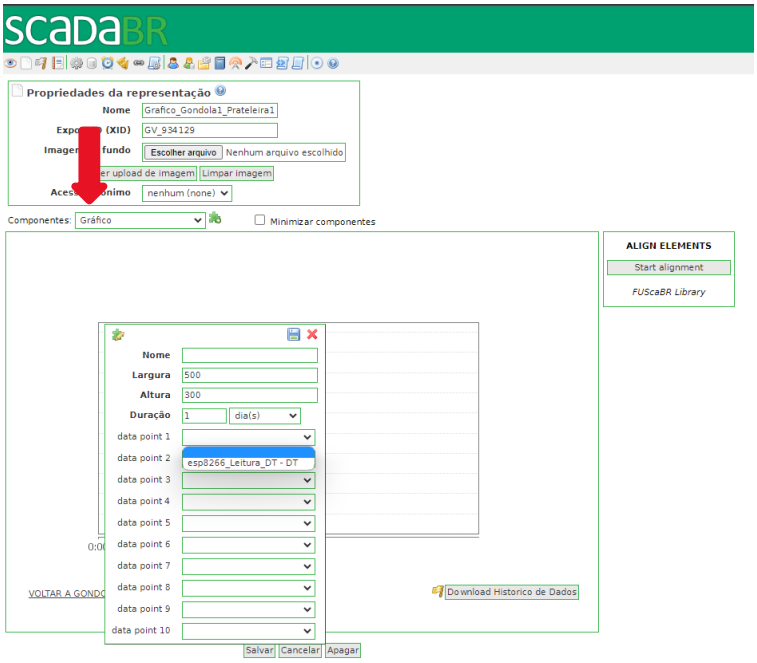


Fonte: O Autor (2023)

Para configurar a segunda tela o mesmo procedimento foi utilizado. Foi escolhido o nome, uma imagem de fundo e também adicionados alguns elementos a esta imagem. No caso foram adicionados botões para acessar as telas 3 e 4 e também para retornar à tela 1. Além disso, também foram colocados 2 elementos visuais para enxergar em tempo real a variação do sinal enviado pela ESP8266.

Para a próxima tela, foi utilizada a mesma lógica, porém desta vez não há nenhuma imagem de fundo. Ao invés dela, um componente “Gráfico” foi adicionado e seus parâmetros configurados, conforme a Figura 28. Também foi adicionado um botão para voltar à tela anterior.

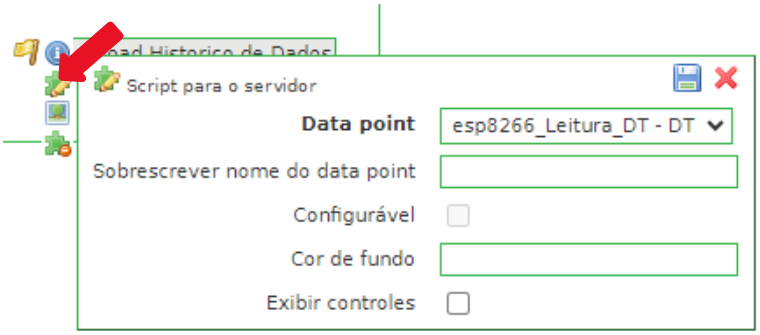
Figura 28 - Configuração da Tela de Gráficos



Fonte: O Autor (2023)

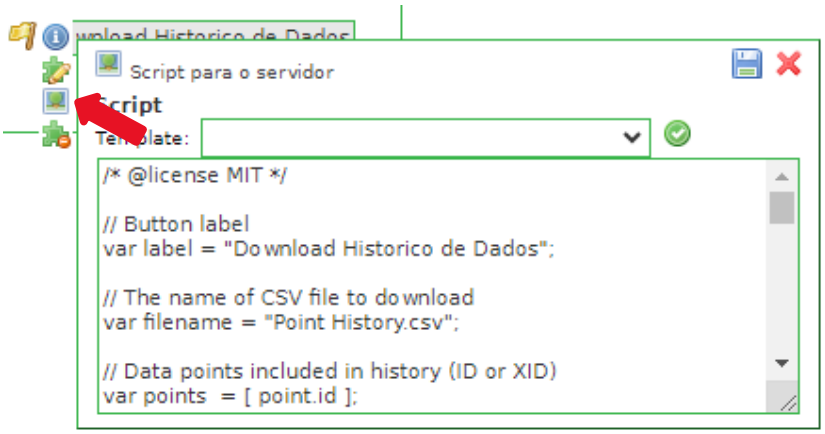
Ainda nesta tela, também foi adicionado um Componente “Script para o servidor” para criar um botão de download dos dados registrados pelo gráfico. Nele, como observado na Figura 29, foi necessário selecionar o Data Point e pela Figura 30 foi importado um Script para que o Download seja possível no formato Excel.

Figura 29 - Seleção do Data Point no Script para o servidor



Fonte: O Autor (2023)

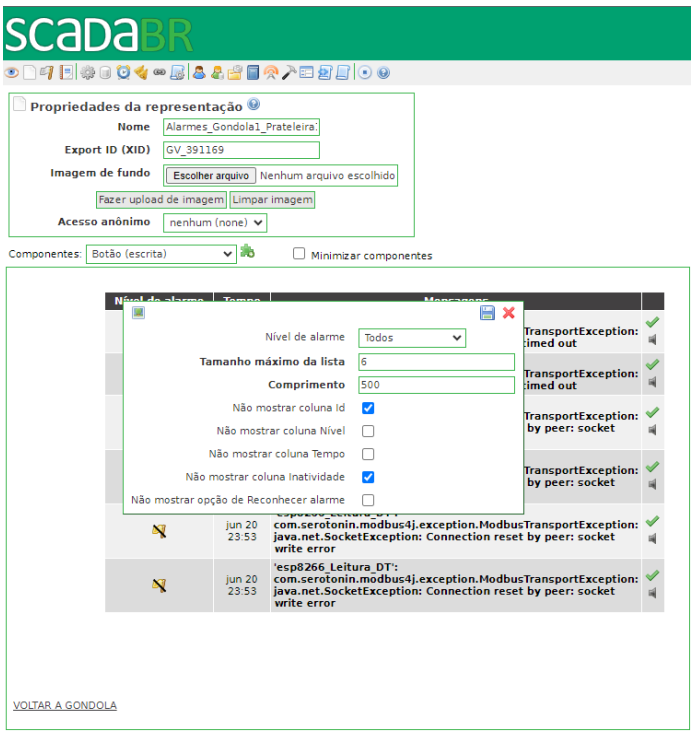
Figura 30 - Importação do Script para o servidor



Fonte: O Autor (2023)

Para a criação da quarta e última tela, a de Alarmes, o mesmo procedimento anterior foi adotado, porém ao invés de adicionar um gráfico como componente, uma lista de alarmes foi adicionada. Nela, as configurações necessárias foram feitas, como ilustrado na Figura 31. Também, assim como nas outras telas, foi adicionado um botão para voltar à tela anterior.

Figura 31 - Configuração da tela de Alarmes

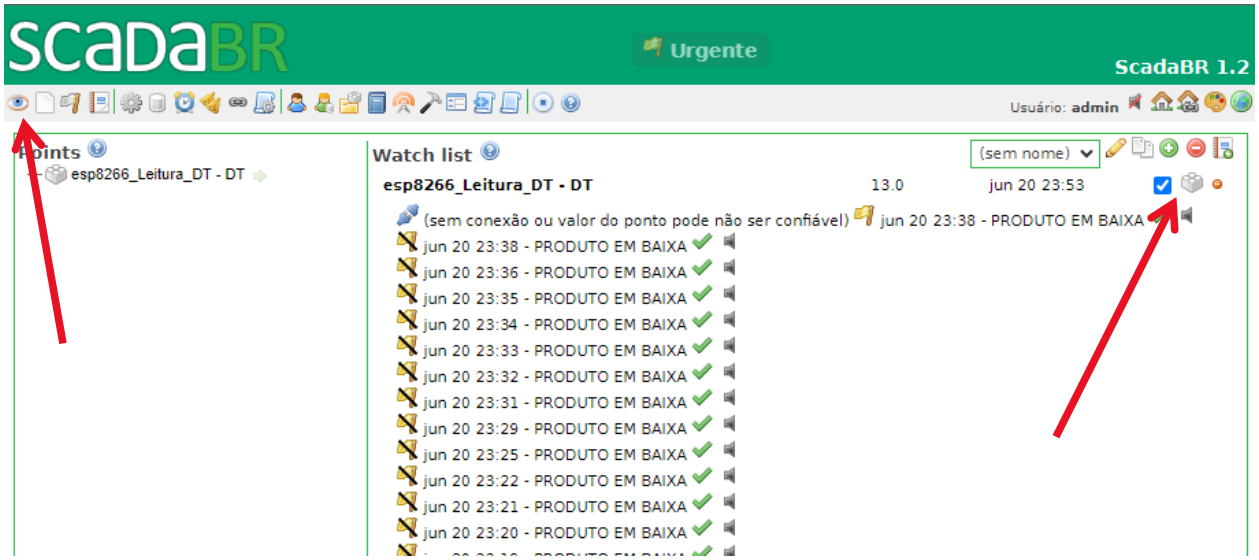


Fonte: O Autor (2023)

Com a tela de Alarmes criada, fez-se necessário o desenvolvimento dos alarmes em si. Para isso, a tela de Watch List, pelo ícone do “olho” no canto superior esquerdo (indicado na Figura 32), foi acessado. Nesta tela, foi preciso clicar nos detalhes do Data Point no canto direito (também indicado na Figura 32). Dentro desta opção, as propriedades do Data Point foram navegadas (indicado na Figura 33).

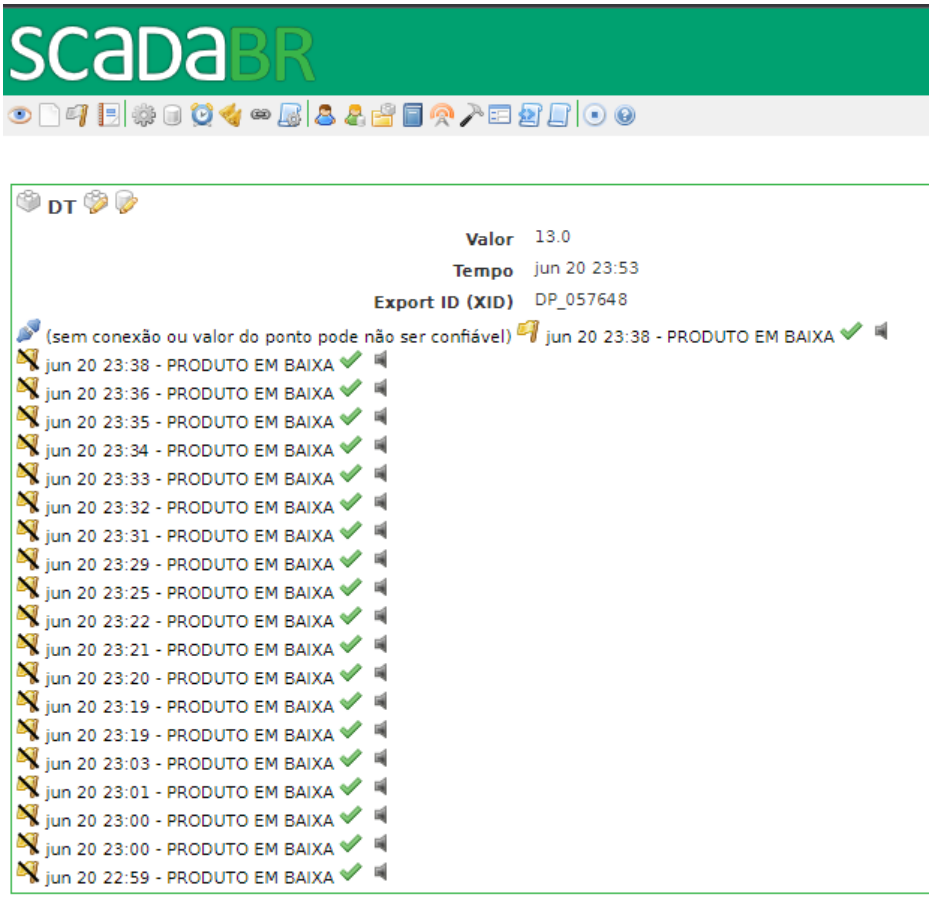
Nessas propriedades, no canto direito da tela, foi possível a criação dos alarmes. Primeiramente, identificou-se o tipo de alarme (que neste caso seria um detector de limite inferior), o nome do alarme, nível de alarme, o limite mínimo de peso necessário para que ele dispare e, por fim, o tempo que este alarme se mantém acionado. Estas configurações podem ser observadas na Figura 34.

Figura 32 - Acessando a Watch List



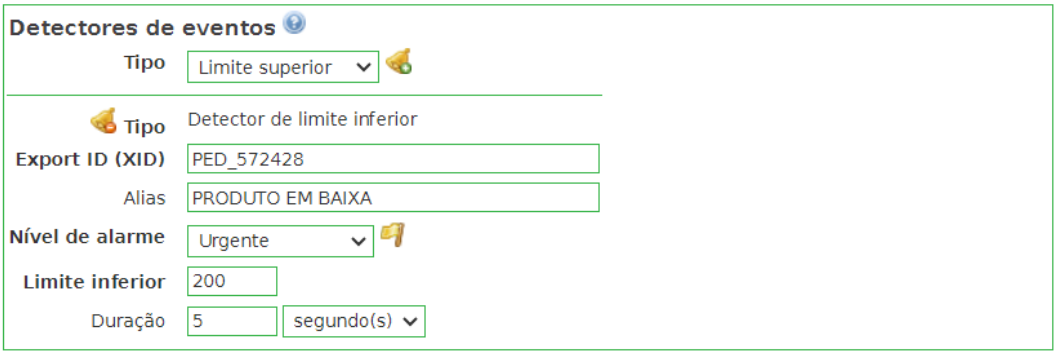
Fonte: O Autor (2023)

Figura 33 - Acessando Propriedades do Data Point



Fonte: O Autor (2023)

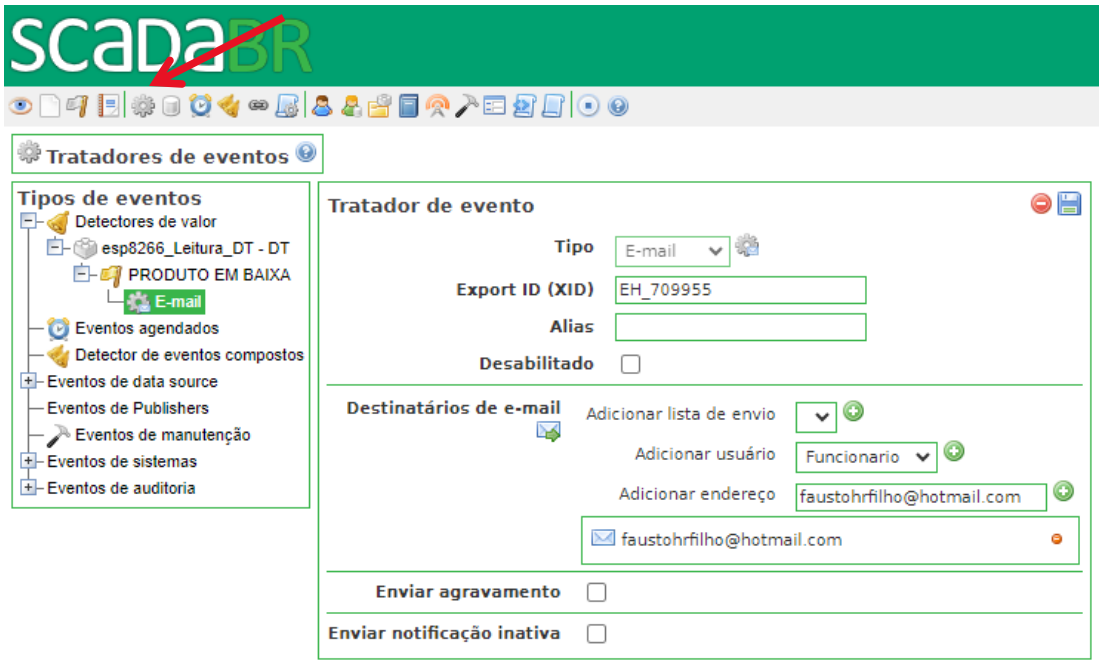
Figura 34 - Definindo os Alarmes



Fonte: O Autor (2023)

Outra função interessante que se pode configurar em “Tratadores de Eventos” (localizado no canto superior esquerdo, conforme a Figura 35) era a de mandar um e-mail a alguém para avisar que este alarme foi disparado.

Figura 35 - Configuração de Tratador de Eventos



Fonte: O Autor (2022)

Por fim, o projeto foi salvo e o ScadaBR executado para a visualização e monitoramento dos dados do dispositivo Modbus IP.

5 Resultados e Discussões

Com as configurações e implementações do capítulo anterior, foi estabelecido a comunicação Modbus TCP entre a placa ESP8266 e o SCADA BR. Pode-se monitorar e controlar os valores dos registradores Modbus no SCADA BR, obtendo uma integração eficiente entre o hardware ESP8266 e o software SCADA BR para o meu projeto.

Primeiramente código usado para a criação da balança e a comunicação Modbus com o SCADA BR se provou muito eficiente. Por meio dele foi possível configurar todo o sistema de balança, bem como seu fator de calibração independente da superfície em que está. Interessante também é notar que a balança é precisa e sofre variação de poucos gramas, tornando-a assim, confiável.

A comunicação via Modbus TCP também foi bem precisa, visto que enquanto o microcontrolador ESP8266 estiver conectado a uma fonte de energia, ele sempre enviará os dados para o ScadaBR. Foi observado um problema com relação ao envio de dados em alguns momentos, porém este tema será tratado mais tarde, haja vista ser um erro de Hardware, não de Software.

Neste momento serão descritos os resultados obtidos nas telas da IHM, que por sua vez se provaram funcionais e objetivas.

Um dos objetivos deste projeto era fazer a leitura de quantos produtos estão em uma prateleira de varejo, como um supermercado, a fim de melhorar o gerenciamento de estoque de pequenas e médias empresas. Este objetivo não conseguiu ser atingido (mensurar a quantidade de produtos), porém pôde-se chegar próximo a um protótipo que atenda essa exigência.

Para a primeira tela, representado na Figura 36, é possível observar o Layout do supermercado, no qual é permitido selecionar a gôndola com produtos desejados. Os botões foram configurados para representar cada gôndola por números. Assim, é possibilitado a varredura de todo o supermercado para saber qual a situação de cada produto. Quando se clica no número 1, por exemplo, a Tela 2 é aberta, que por sua vez faz parte da seção de “Cereais e Farináceos”. Com isso, podemos concluir que essa tela dá ao operador o poder de enxergar a quantidade de produtos de todo o supermercado sem a necessidade de ir fisicamente conferir.

Figura 36 - Tela 1: Layout do Supermercado

Fonte: O Autor (2023)

Para a segunda tela, representado na Figura 37, é possível observar a prateleira programada para medir o peso total aplicado sobre ela. Parte do objetivo do projeto era conseguir “setar” o peso unitário do produto que seria colocado na prateleira. Com isso, o valor de peso total recebido via protocolo Modbus TCP deveria ser dividido pelo peso unitário, mostrando assim, a quantidade real de produtos da prateleira, não o peso total.

Contudo, apesar desta etapa não ter sido completa, é palpável a ideia de como o sistema funcionaria em um ambiente verdadeiro. Observando a variação do peso total em tempo real, é possível checar se o estoque que existe no sistema do supermercado coincide com o sistema da Gôndola Automatizada. Caso haja alguma divergência, existem algumas possíveis opções:

1- O produto específico está no carrinho do cliente;

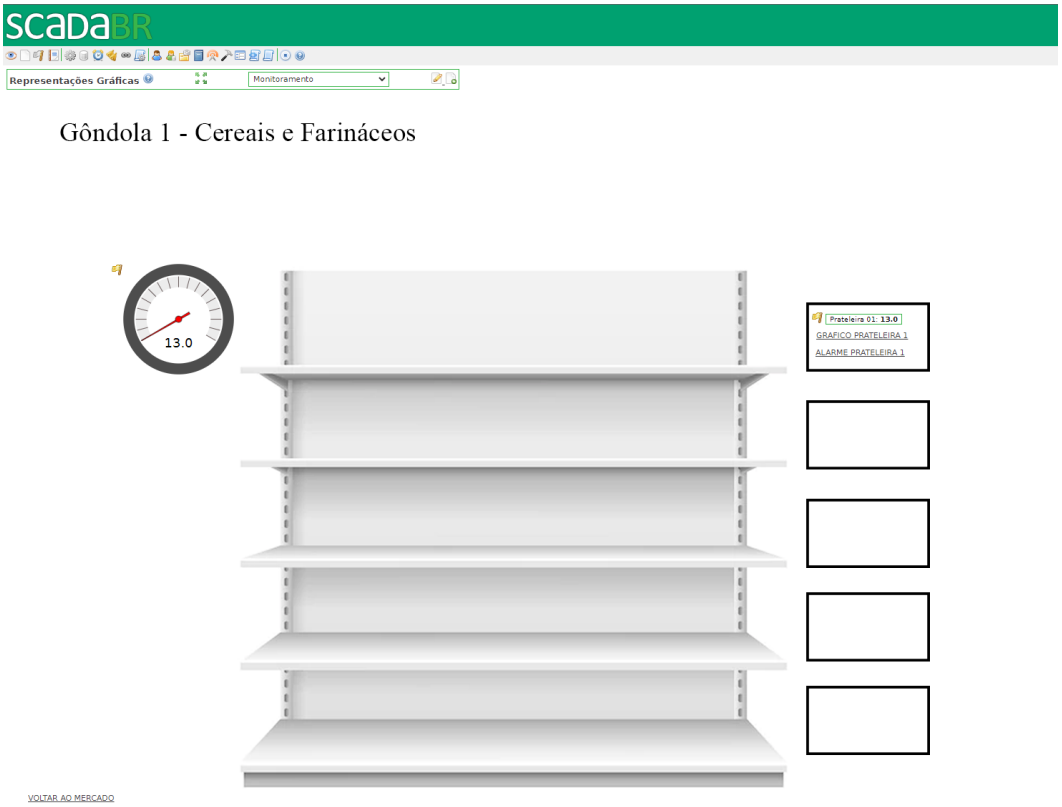
2- O produto foi colocado em algum lugar errado e, consequentemente, também será possível a identificação, pois alguma das outras prateleiras também estará errada;

3- Furto de produtos.

Este sistema visa identificar estas perdas de furtos para os empreendimentos, pois muitas vezes há muita demora para a sua identificação (quando ela existe) e consequentemente em uma demora ainda maior para formar um plano de ação que combata esse problema. Assim é gerado mais perda pois elas não são identificadas com rapidez e a empresa entra em um ciclo de perdas e muitas vezes nem se dá conta disso. Portanto, esta tela seria a melhor amiga do gestor de estoque de varejo.

Nesta tela também é possível voltar ao layout total do mercado pelo canto inferior esquerdo. Há a possibilidade de acessar o “Gráfico da Prateleira” (Tela 3) e os “Alarmes de Prateleira” (Tela 4) pela lateral direita da representação da gôndola.

Figura 37 - Tela 2: Supervisório das Prateleiras da Gôndola



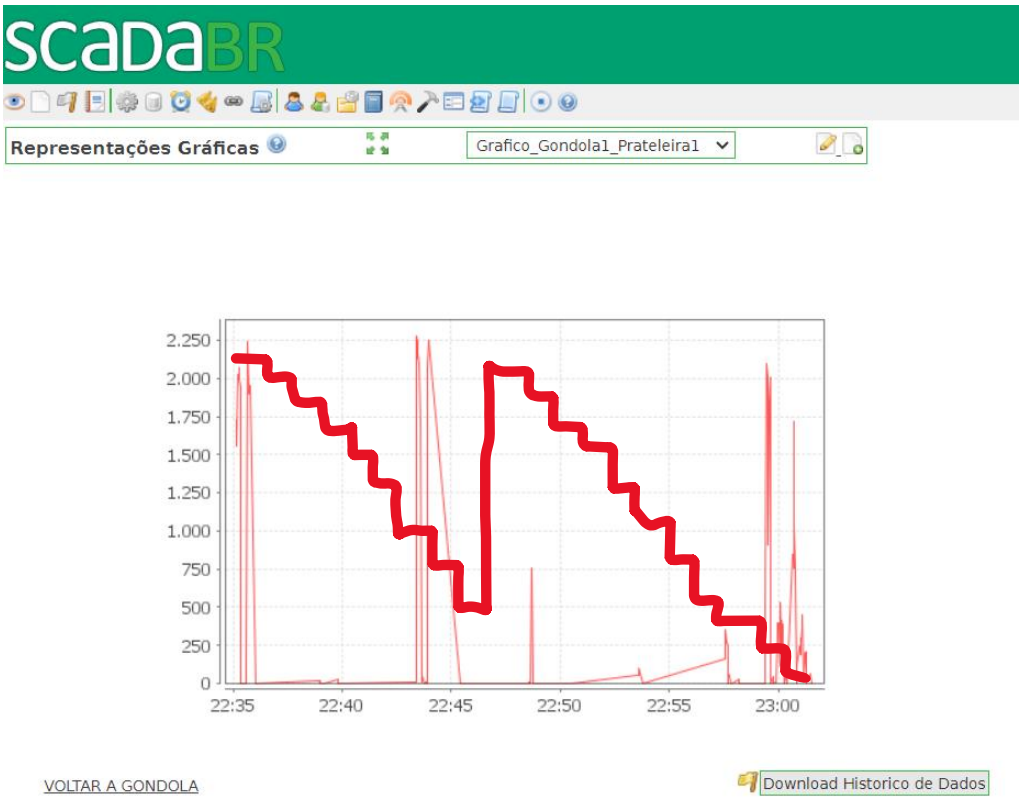
Fonte: O Autor (2023)

Na terceira tela, observa-se a variação do produto na prateleira pelo tempo. Novamente, o ideal para o projeto seria que o valor fosse a quantidade de produtos, não o peso total. Mas da mesma forma que é possível observar bem o funcionamento do sistema na tela anterior, também é possível nesta.

Nesta tela é possível ver toda a variação de produtos que são colocados na balança. A forma normal esperada deste gráfico seria uma curva quadrática, ficando muito tempo constante e então desce uma unidade (normalmente) devagar ao longo do tempo, pois os clientes vão retirando os produtos e, de repente, sobe bastante quando o repositor recoloca os produtos na gôndola. Este gráfico não pode ser observado pois não foi possível o teste em ambiente real. A variação também está bem nítida devido ao erro que será explicado posteriormente. Podemos observar o valor medido e como deveria ser pela Figura 38.

Outra função interessante para o gráfico é a análise de dados visuais que é possível ser feita. Com ele, a observação de quais produtos mais tem saída e quais não tem tanto pode ser feita. Dessa forma, o entendimento acerca da logística da empresa melhora, alimentando a equipe comercial com informações valiosas a respeito de seus ativos.

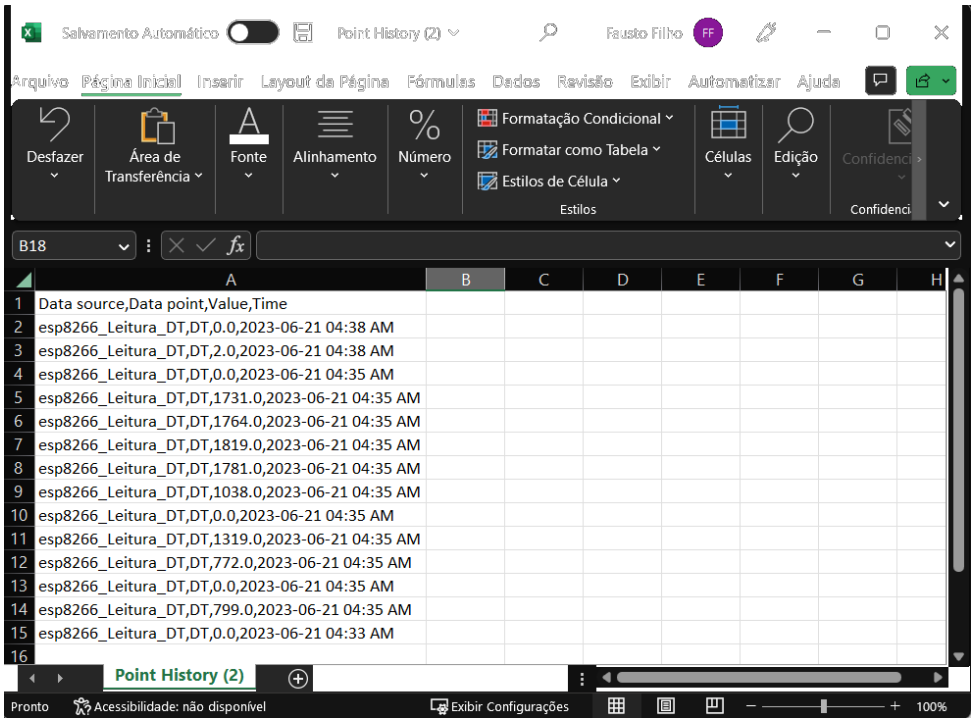
Figura 38 - Tela 3: Gráfico de variação de peso medido na prateleira



Fonte: O Autor (2023)

Na tela 3, o operador também consegue gerar um relatório a cerca dos dados obtidos via excel, como ilustrado na Figura 39. Desta forma é possível levar os dados para os gerentes de maneira mais eficiente a respeito do que está ocorrendo na empresa com relação aos produtos.

Figura 39 - Dados exportados no Excel



Fonte: O Autor (2023)

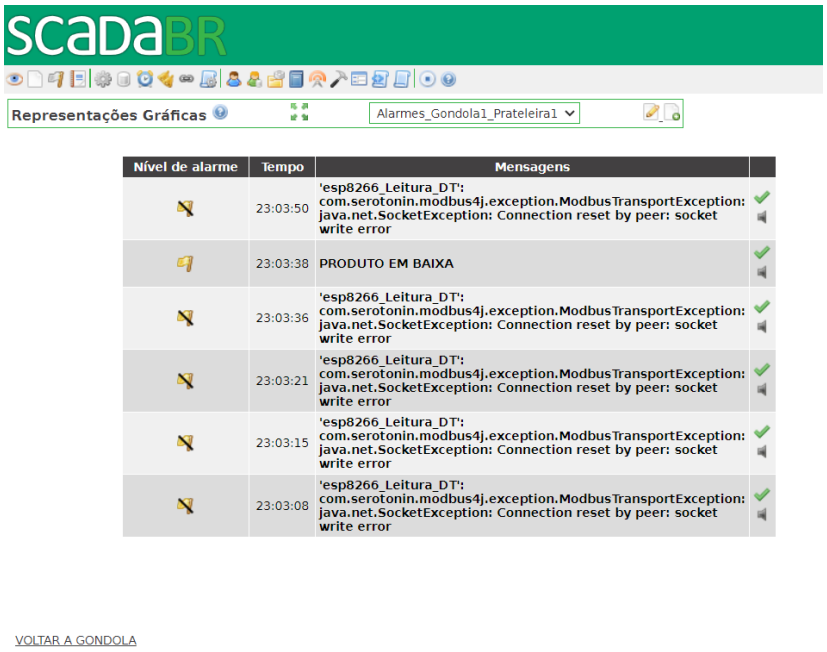
Na quarta tela, um resumo dos últimos 6 eventos que merecem a atenção do operador é apresentado. Porém, o dado mais importante obtido por meio dele é com relação a falta de produto na prateleira. Quando há um produto em baixa (vide Figura 40), um alarme do tipo “Urgente” é disparado informando ao operador que o mesmo precisa ser reposto na gôndola.

Este tipo de informação é importante, pois assim é possível gerenciar melhor a equipe com relação as prioridades. Muitas vezes em varejos não existe uma lógica (na maioria das vezes por falta de treinamento) por trás dos colaboradores que são responsáveis pela atividade de reposição. Com essa ferramenta é possível trazer organização e controle por parte dos líderes de equipe.

Outra ferramenta que pode ajudar na agilidade da reposição do item em baixa na gôndola é a função do e-mail automático. Geralmente, em um supermercado, existe um colaborador responsável por sessão ou corredor, porém esta pessoa transita muito

pelo espaço de trabalho impossibilitando que ela esteja ciente de tudo que acontece no lugar de sua responsabilidade. Então, com esta ferramenta, é possível informar esta pessoa em tempo real, por meio do seu celular, que algo de seu encargo necessita de atenção, que seria no caso, a reposição de um produto. Podemos observar este e-mail automático por meio da Figura 41.

Figura 40 - Tela 4: Lista de Alarmes disparados no teste



Fonte: O Autor (2023)

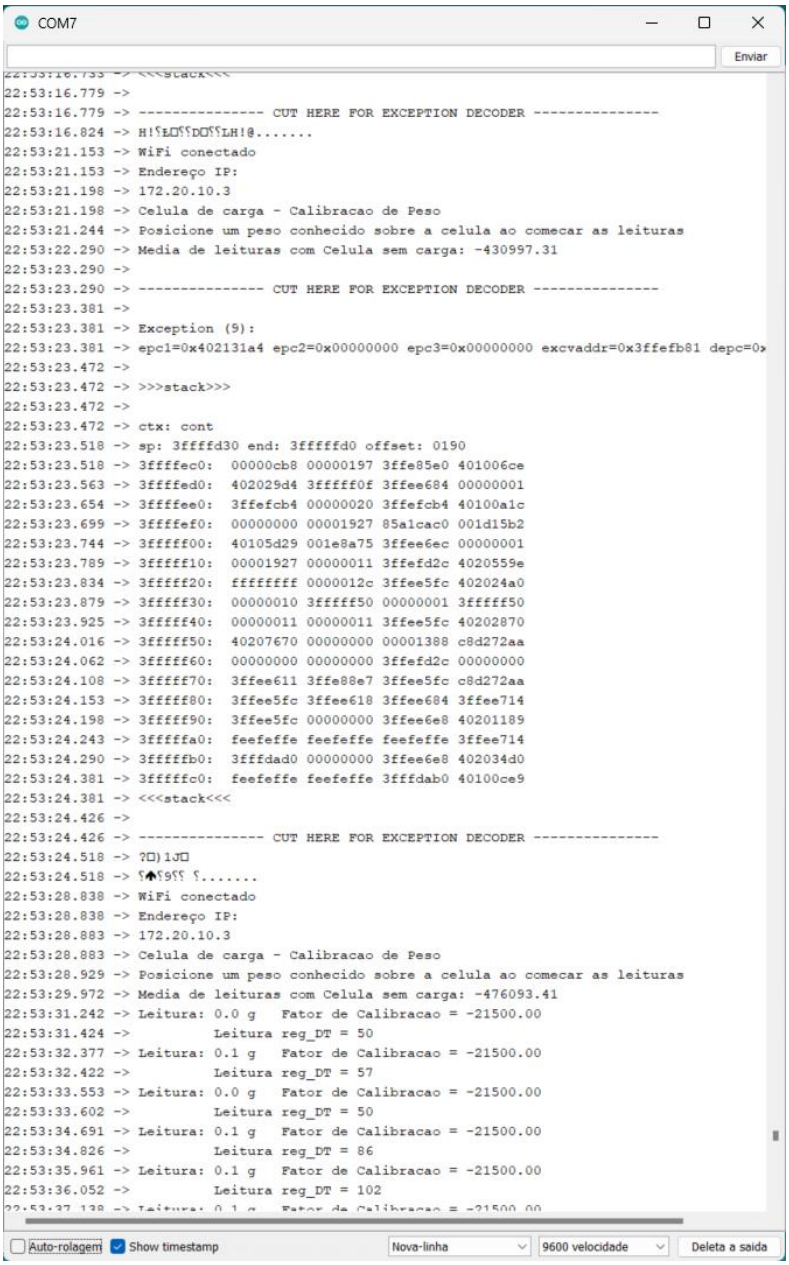
Figura 41 - E-mail automático de alarme do ScadaBR

Fonte: O Autor (2023)

O sistema funciona muito bem, apesar de não apresentar os parâmetros propostos no início do projeto. Contudo, foram obtidos resultados inesperados ao longo dos testes.

Em quase todos os testes, o código escrito no ESP8266 era enviado via protocolo Modbus TCP para o SCADA BR e este o recebia sem problemas. Entretanto, algumas vezes uma perda repentina de sinal enviado pelo microcontrolador acontecia, fazendo com o que o Data Point estacionasse em 0 (zero). Foi tentado localizar o erro checando a Serial Port do código no IDE do Arduino e o seguinte erro foi ilustrado na Figura a seguir.

Figura 42 - Erro identificado na porta serial do Arduino IDE



Fonte: O Autor (2023)

O que acontece é que, às vezes, quando o dispositivo tenta fazer as leituras das células de carga, este erro acontece até que o código seja resetado. Assim, ele se conecta ao Wi-Fi, retorna o endereço IP, e ao tentar ler novamente os sinais o erro acontece mais uma vez. Isso acontece algumas vezes em loop, até que, em certo momento ele consegue inesperadamente realizar as leituras como se nada houvesse acontecido. Porém mais tarde, com a leitura contínua, o erro persiste. Supostamente não há nada específico que desencadeie este erro, aparenta ser aleatório.

Ao que tudo indica, isto pode ser um erro de exceção ocorrido no código. O rastreamento do erro aponta que a exceção ocorreu no endereço de programa (epc) 0x4000df64. O restante do rastreamento mostra a pilha de execução do programa no momento do erro. Identificar a causa exata do erro com base apenas no rastreamento era impossível. No entanto, há alguns motivos possíveis.

As explicações para este erro não podem ser vindas provenientes da escrita do código, atualização de bibliotecas/firmware ou erro de configuração dos parâmetros do Modbus TCP. Porque o código está funcional, todas as configurações no ScadaBR estão de acordo com o protocolo Modbus e todas as bibliotecas e firmwares foram atualizadas.

O primeiro motivo para tal erro pode ser por causa de interferências e/ou ruídos. A montagem do microcontrolador ESP8266 e módulo amplificador HX711 foram feitas em uma protoboard. Estes equipamentos são muito suscetíveis a falhas devido a inúmeros fatores. Conexões soltas, resistências de contato que podem gerar quedas de tensão, interferências de sinal (eletromagnéticas e radiofrequência), acoplamento capacitivo e tantos outros.

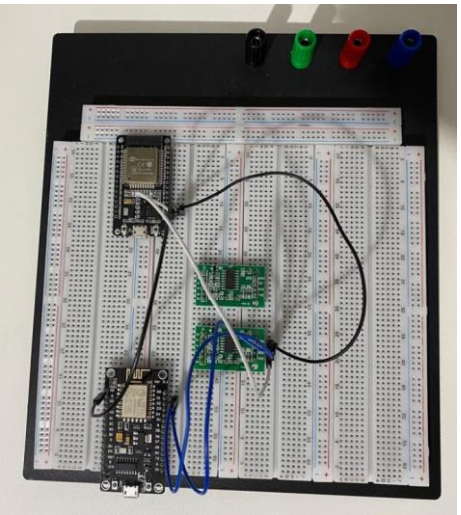
Essas interferências da protoboard podem ser descartadas, pois apenas o código que contém a comunicação Modbus TCP sofre deste erro. Se o problema realmente fosse ela, todos os outros códigos sofreriam da mesma interferência, o que de fato não ocorre. Além disso, como ilustrado na Figura 5.7, depois de um tempo neste loop de erros, as leituras voltam a funcionar sem motivo aparente.

Segundo pesquisas, a explicação mais plausível é que uma possível falta de energia no módulo HX711 possa causar esse erro. O ESP8266 pode consumir picos de energia significativos ao enviar informações via Wi-Fi, o que pode resultar em uma queda de tensão e falta de energia para o amplificador HX711.

Para tentar provar esta hipótese, foi acoplado um microcontrolador ESP32 na protoboard exclusivamente para fornecer energia ao módulo amplificador por meio do pino 3v3, como pode ser observado na Figura 43. Assim, o código foi compilado novamente e este erro reduziu drasticamente a quantidade de vezes que ele aparece.

Devido ao não consumo de energia para enviar os dados via Wi-Fi, a alimentação do módulo HX711 se tornou menos oscilante. Contudo, o mesmo erro ainda acontece, porém em menor frequência.

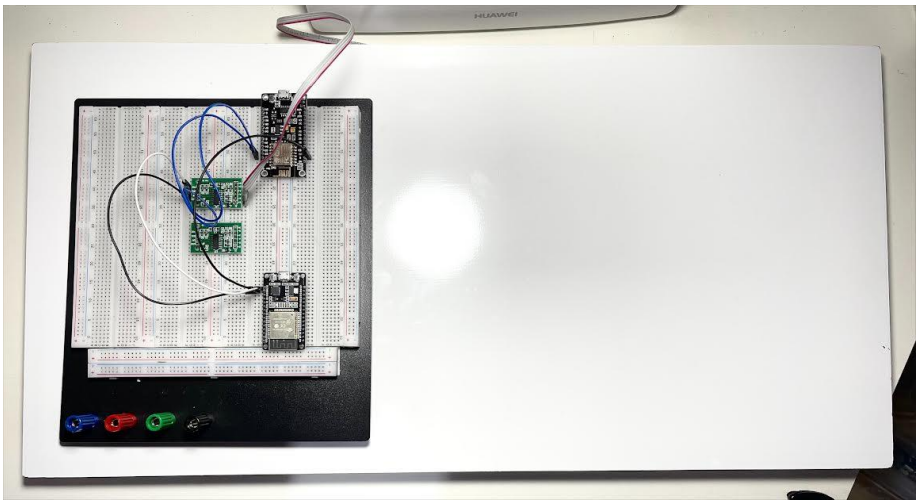
Figura 43 - ESP32 sendo utilizada exclusivamente como fonte para HX711



Fonte: O Autor (2023)

Voltando ao DataSheet do fabricante (Avia Semiconductor), notou-se que a fonte ideal para o módulo HX711 seria uma externa de 5V. Como a ESP32 fornece apenas 3,3V era de se esperar que esta falta de energia para a leitura correta dos sinais das células de carga voltasse a ocorrer em algum momento. Portanto, a melhoria deste sistema fica para trabalhos futuros.

Figura 44 - Prateleira confeccionada para o projeto



Fonte: O Autor (2023)

6 Conclusão e Trabalhos Futuros

A conclusão deste trabalho de conclusão de curso demonstrou o desenvolvimento de uma gôndola automatizada para varejo, que visa otimizar o gerenciamento de estoque, prevenir perdas e agilizar a reposição de produtos em pequenas e médias empresas do setor. Por meio da implementação de uma prateleira equipada com células de carga, foi possível obter medições precisas do peso total dos produtos na prateleira. Além disso, a integração do microcontrolador ESP8266 e do módulo amplificador HX711, utilizando o protocolo Modbus TCP, permitiu a comunicação eficiente com o SCADA BR, fornecendo um ambiente de supervisão adequado para atingir os objetivos propostos.

Durante o desenvolvimento do projeto, identificaram-se alguns erros e oportunidades de melhoria. Um dos principais problemas foi a falta de energia fornecida ao módulo HX711, o que resultou em leituras incorretas. Para aprimorar o hardware, seria recomendável o desenvolvimento de uma fonte de alimentação externa de 5V exclusiva para o módulo, com filtros capacitivos, garantindo um fornecimento de energia estável e evitando interferências. Além disso, a substituição da protoboard por conexões mais confiáveis, como soldagem dos dispositivos, contribuiria para uma montagem mais robusta e livre de problemas de conexão.

Outro objetivo que não foi alcançado foi a configuração no ScadaBR para permitir o ajuste do peso unitário dos produtos na prateleira. Essa funcionalidade possibilitaria a exibição da quantidade de produtos, em vez do peso total, no ambiente de supervisão. Para melhorar essa questão, seria necessário propor uma melhoria no ScadaBR, implementando a capacidade de configurar o peso unitário dos produtos, para que essa informação pudesse ser impressa no supervisorio.

No que diz respeito a futuros trabalhos, é possível destacar diversas áreas de aprimoramento. Primeiramente, os erros identificados, como a falta de energia no módulo HX711 e as conexões instáveis, devem ser corrigidos para garantir um funcionamento confiável do sistema. Além disso, é importante explorar outras funcionalidades no ScadaBR, como a implementação de alarmes mais sofisticados, por exemplo, alertando não apenas a baixa quantidade de produtos, mas também outras condições relevantes para o gerenciamento de estoque.

Outra possível melhoria seria a implementação de um sistema de controle de temperatura e umidade na gôndola, permitindo monitorar e manter as condições ideais para produtos sensíveis. Além disso, poderiam ser exploradas outras tecnologias de comunicação, como RFID, para rastreamento de produtos individualmente.

Quanto ao aprendizado adquirido, este projeto permitiu uma compreensão mais

aprofundada dos princípios de controle e automação, além de fornecer habilidades práticas no desenvolvimento de sistemas embarcados integração de diferentes componentes. Foi uma oportunidade valiosa e enriquecedora para aplicar conceitos teóricos em um contexto real e entender as complexidades envolvidas na criação de soluções tecnológicas para o setor varejista.

Por fim, é fundamental ressaltar o impacto positivo que essa gôndola automatizada pode trazer para a vida dos varejistas. Com a implementação das melhorias sugeridas e a continuidade do desenvolvimento, a gôndola automatizada pode se tornar uma solução valiosa e amplamente adotada no setor de varejo. Ao oferecer um gerenciamento mais eficiente de estoque, prevenção de perdas e agilidade na reposição de produtos, essa solução pode ajudar a otimizar as operações de pequenas e médias empresas, aumentando sua produtividade e melhorando a experiência dos clientes.

Referências

Groover, M. P., & Zimmers, E. W. (2018). Automation, Production Systems, and Computer-Integrated Manufacturing. Pearson.

Marques, L., Afonso, J., & Pimenta, A. (2017). Introduction to SCADA Systems: SCADA Systems Architecture and SCADA Protocol Standards. In Communications in Computer and Information Science (Vol. 755, pp. 307-318). Springer.

Silva, J. C., & Mendonça, J. P. (2019). SCADA systems in critical infrastructures: A survey on security issues. IEEE Communications Surveys & Tutorials, 21(3), 2276-2297.

Célula de Carga | OMEGA Engineering Brasil. Disponível em: <https://br.omega.com/prodinfo/celulas-de-carga.html>.

ROBOCORE. Célula de Carga 50kg. Disponível em: <https://www.robocore.net/sensor-robo/celula-de-carga-50kg >.

LIMA, Thiago. Ponte de Wheatstone. Disponível em: <http://www.embarcados.com.br/ponte-de-wheatstone.

NATIONAL INSTRUMENTS. Medições de deformação com Strain Gages. Disponível em: < http://www.ni.com/white-paper/3642/pt>.

MARCELINO, M. L.; MASSA, A.; FLORIAN, F.; BITTENCOURT, J. Aplicação da ponte de Wheatstone com extensómetros elétricos: Instrumentação de um link para medições das forças de tração e compressão. Universidade de Araraquara, 2018.

LOCATELLI, Caroline. Balança com célula de carga e HX711; 2019. Disponível em: <http://www.curtocircuito.com.br/blog/balanca-com-celula-de-carga-ehx711>.

AVIA SEMICONDUCTOR 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales DESCRIPTION. [s.l: s.n.]. Disponível em: <https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf.

KOLBAN, Neil. Kolban’s book on ESP32. [S.l.]: Leanpub, 2018.

CALIBRAÇÃO E COMPENSAÇÃO DE SENSORES DE PRESSÃO PIEZORRESISTIVOS. [s.l: s.n.]. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3140/tde-08092010-121153/publico/Dissertacao_Alex_Fukunaga_Gomes.pdf>.