Dispositivo IoT para monitorar indicador de pressão através de processamento de imagem

Gabriel Cavalheiro Francisco [*] **Heitor Medeiros Florencio** [**] **Jefferson Doolan Fernandes** [***]

[*] *Universidade Federal do Rio Grande do Norte, RN (e-mail: gcavalheirof@gmail.com)*
[**] *Instituto Metrópole Digital, Universidade Federal do Rio Grande do Norte, RN (e-mail: heitorm@imd.ufrn.br)*
[***] *Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, RN (e-mail: jefferson.fernandes@escolar.ifrn.edu.br)*

Abstract: Applications based on Internet of Things (*"Internet of Things"* - IoT) devices allow consuming and monitoring data in real time and integrating different processes. This integration of data and processes is one of the main advantages of using IoT technology in Industry 4.0. The growth in the computational power of these IoT devices has enabled their use in different applications, such as the measurement of process variables by image processing. This work aims to develop an application with an IoT device, based on ESP32 with an integrated camera, to capture images from an analog manometer of a water treatment plant in a pharmaceutical industry and generate the measured pressure value. This data will be made available on the factory supervisory for real-time monitoring through the MQTT protocol. The pressure values were made available on the factory supervisory to allow the visualization of the data in real time.

Resumo: As aplicações baseadas em dispositivos de Internet das Coisas (*"Internet of Things"* - IoT) permitem consumir e monitorar dados em tempo real e integrar diferentes processos. Essa integração de dados processos é uma das principais vantagens do uso da tecnologia IoT na Indústria 4.0. O crescimento do poder computacional desses dispositivos IoT vem possibilitando o uso em diferentes aplicações, como a medição de variáveis de processos por processamento de imagem. Este trabalho tem como objetivo desenvolver uma aplicação com dispositivo IoT, baseado no ESP32 com câmera integrada, para capturar imagens de um manômetro analógico de uma planta de tratamento de água de uma indústria farmacêutica e gerar o valor de pressão medido. Esse dado será disponibilizado no supervisório da fábrica para monitoramento em tempo real através do protocolo MQTT. Os valores de pressão foram disponibilizados no supervisório da fábrica para permitir a visualização dos dados em tempo real.

Keywords: IoT; Industry 4.0; Image processing; MQTT; Node-Red.
Palavras-chaves: IoT; Indústria 4.0; Processamento de imagem; MQTT; Node-Red.

1. INTRODUÇÃO

O foco do movimento da Industria 4.0 é na produção inteligente, que tem o objetivo de integrar os dados dos diversos processos de uma fábrica. Essa integração permite correlacionar as variáveis de cada etapa e definir indicadores, como indicadores de manutenção preventiva de máquinas, com o intuito de aumentar a eficiência da fábrica (Kagermann et al., 2013). Entretanto, substituir equipamentos antigos nem sempre é viável por exigir alto investimento financeiro, tempo de transição da nova tecnologia e validação do processo.

Além disso, a atualização (*retrofit*) de máquinas ou sistemas em um ambiente industrial gera impacto não somente no custo da nova tecnologia, mas também no período de parada de funcionamento do processo para troca dos equipamentos da planta e, em alguns casos, na curva de aprendizagem de manutenção dos sistemas por parte da equipe de engenharia e manutenção.

Neste trabalho é desenvolvido uma aplicação com um dispositivo de Internet das Coisas (IoT) para realizar a aquisição da informação de pressão de um manômetro analógico de forma não invasiva, ou melhor, sem a necessidade de alteração na instalação do sistema atual. O dispositivo IoT é desenvolvido para instalação em uma planta de tratamento de água purificada dentro de uma indústria farmacêutica. Essa unidade de produção de água purificada abastece toda a indústria farmacêutica.

Existem vários processos de tratamento para que a água atinja os parâmetros ideais de produção. Com isso, existem várias tubulações, bombas e válvulas para o controle de fluxo da água. Existem mais de 15 manômetros analógicos instalados na planta, que não disponibilizam esses dados

de pressão para o supervisório da fábrica, impossibilitando o monitoramento em tempo.

A aplicação desenvolvida neste trabalho torna viável o monitoramento em tempo real desses valores de pressão da planta. Ao invés de realizar a troca dos manômetros analógicos, é possível instalar os dispositivos IoT, que atuam de forma não invasiva, evitando a necessidade de trocar instrumentos e partes estruturais da planta. Outra vantagem do dispositivo IoT está em sua flexibilidade, sendo possível utilizar o mesmo em conjunto para outras soluções com novos sensores.

A proposta deste trabalho é que o dispositivo IoT esteja integrado com uma câmera, realizando a aquisição da imagem do ponteiro do manômetro constantemente. Após essa coleta da imagem, é realizado uma manipulação da imagem, afim de isolar o ponteiro e determinar assim a pressão em tempo real.

2. REFERENCIAL TEÓRICO

2.1 IoT

O conceito de internet das coisas (IoT) se refere a conectar vários objetos em uma só rede, possibilitando com que os mesmos troquem informações entre eles (Gokhale et al., 2018).

A ideia fica mais clara quando olhamos para a automação residencial. Hoje é muito comum controlar luzes, câmeras, ar-condicionado e até mesmo robô aspirador por um aplicativo no *smartphone* ou assistente virtual. A ideia de coletar dados desses dispositivos em tempo real e a possibilidade de ligar/desligar algum aparelho com base nessas informações é o que faz a tecnologia de internet das coisas algo tão atrativo.

A implantação de dispositivos IoT gerou um aumento na conectividade entre diversas coisas em diversos ambientes, em tempo real, com uma velocidade de crescimento sem precedentes (Kassab e Darabkh, 2020). A indústria, por sua vez, já permite o monitoramento em tempo real de diversos parâmetros. A aplicação de IoT nos sistemas industriais gera troca de informação que estão fora dos sistemas de controle e permite uma melhor rastreabilidade dos processos e coisas.

Para a indústria, isso reflete diretamente em eficiência e na produção inteligente. Tanto que nasceu uma subcategoria, chamada de IIoT ("*Industrial internet of things*") específica para o chão de fábrica, com foco na comunicação entre máquinas e na coleta contínua de dados, dando uma visão mais clara de como está atuando cada processo.

2.2 Processamento de imagem

A imagem digital é uma matriz de vários *pixels*, cada um representando uma cor específica. Um processamento de imagem é a transformação sucessiva de uma imagem digital com o intuito de se obter informações específicas (Persechino e Albuquerque, 2015). Não existe um modelo exato de processamento de imagens pois o objetivo final depende de cada aplicação. Existe uma enorme variedade de aplicações de processamento de imagem, por exemplo no processamento de imagens de ultrassom e tomografias.

A definição das funções utilizadas no processamento da imagem depende da cada aplicação (Câmara et al., 1996). No entanto, em geral as etapas podem ser classificadas como:

- pré-processamento;
- realce;
- classificação;

O pré-processamento engloba desde a captura da imagem até o uso de técnicas para redução de ruídos, correção de distorções, entre outras. Os procedimentos de posicionamento de câmera e de iluminação do ambiente podem aumentar consideravelmente a eficiência do modelo de processamento da imagem.

Na etapa de realce a imagem sofre alterações para que fique nítido os objetos, ficando assim fácil de identificá-los.

Por último, a classificação é o momento em que esses objetos são parametrizados, atribuindo dados importantes a cada forma geométrica identificada na imagem e fazendo uso desses dados no fim da aplicação.

3. METODOLOGIA

Este artigo apresenta o desenvolvimento de um dispositivo IoT para captura e envio de uma imagem de um manômetro com o intuito de realizar um processamento da imagem para gerar o valor de pressão indicado no manômetro. A Figura 1 apresenta a arquitetura do sistema desenvolvido neste trabalho.

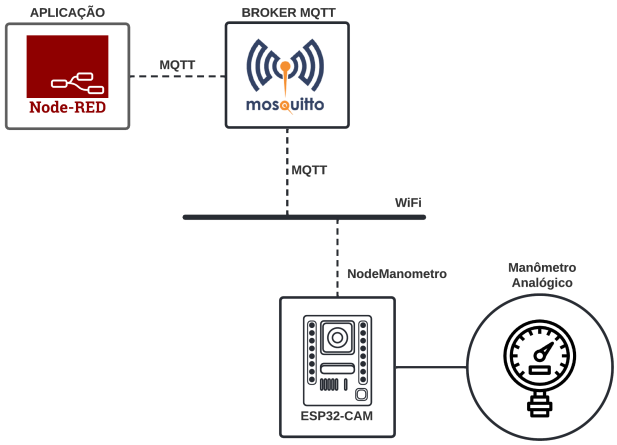


Figura 1. Arquitetura do sistema

O NodeManometro é o dispositivo IoT responsável pela aquisição da imagem do visor do manômetro e envio do dado (imagem) via o protocolo MQTT para o *broker* da aplicação. O dado é consumido por uma aplicação desenvolvida em Node.js no NodeRED. Essa arquitetura IoT permite que vários dispositivos IoT sejam instalados nos diversos manômetros analógicos para geração dos diversos valores de pressão com o mesmo programa de processamento de imagem na *middleware* desenvolvido no NodeRED.

Um *broker* MQTT *open-source* Mosquitto foi instalado para manter a troca de dados entre os dispositivos IoT do processo a aplicação de tratamento dos dados.

A aplicação foi desenvolvida utilizado o Node-RED devido sua facilidade na integração com dispositivos IoT, principalmente utilizando comunicação MQTT. Foi implementado um bloco do Node-RED que executa o programa que realiza o processamento da imagem. Esse programa foi desenvolvido na linguagem Python e realiza o pré-processamento da imagem e, em seguida, calcula o valor da pressão a partir do ângulo do ponteiro da imagem do manômetro.

Um requisito essencial desse projeto é o desenvolvimento de soluções para a indústria com o uso de dispositivos IoT de baixo custo e não invasivo, evitando a necessidade de atualização dos sistemas de automação das máquinas e plantas envolvidas. COm isso, o dispositivo escolhido para desenvolver o NodeManometro foi o ESP32-CAM.

3.1 Dispositivo IoT NodeManometro: ESP32-CAM

O ESP32-CAM é contém um microcontrolador ESP32 com câmera integrada, módulo WiFi, suporte para cartão SD e com tamanho reduzido de 40x27mm. Devido ao tamanho reduzido e pode computacional dessa placa ESP32-CAM, é possível utilizá-la em diversas aplicações IoT, como reconhecimento facial de pessoas em uma operação, leitura de QRCode, detecção de movimento de produtos, entre outras.

Figura 2. Placa ESP32-CAM.

Existem dois pinos de alimentação, um pino de 3.3V e outro de 5V. Porém, o próprio fabricante recomenda que se utilize a alimentação de 5V para captura de imagens para evitar efeitos ruídos indesejados na imagem. A placa possui um regulador de tensão AMS1117 que suporta até 15V, ou seja, é possível conectar pilhas em série, tornando assim o dispositivo totalmente remoto, independente de um espaço físico.

Diferente do ESP32 e de seu antecessor, o ESP8266, a placa ESP32-CAM não possui um conector USB, mantendo assim um tamanho bem reduzido. Foi utilizado o conversor USB-TTL FT232RL para permitir a gravação do código no microcontrolador.

Como existem vários modelos dessa placa no mercado, o primeiro passo no código é de fazer um mapeamento dos pinos. Em seguida, são definidas as propriedades da imagem, como o formato, tamanho e saturação. Como a imagem vai passar por um recorte e vários processamentos, a resolução definida foi 320x240 *pixels* e o formato JPEG, por ser comum nos diversos sistemas operacionais. Por fim, também é necessário adicionar uma conexão com a rede WiFi do ambiente e uma lógica para tentar se reconectar a cada 500ms, caso seja desconectado.

Com isso, o ESP32-CAM já está totalmente configurado, capturando as imagens e enviando para o *broker* Mosquitto via protocolo MQTT em uma taxa de amostragem definida pela aplicação.

3.2 Algoritmo de processamento de imagem

A aplicação, implementada no Node-RED, inicia com a aquisição da imagem através do consumo do tópico do *broker* MQTT. Em seguida, é executado um *script* em Python que realiza todo o processamento da imagem e retorna o valor de pressão.

O objetivo do *script* de processamento é gerar uma imagem com o ponteiro completamente isolado para então calcular o ângulo de inclinação do ponteiro e determinar a pressão que está sendo indicada no manômetro. Para permitir a detecção desse valor de ângulo, a imagem precisa passar por várias etapas de pré-processamento com o objetivo de isolar completamente o ponteiro.

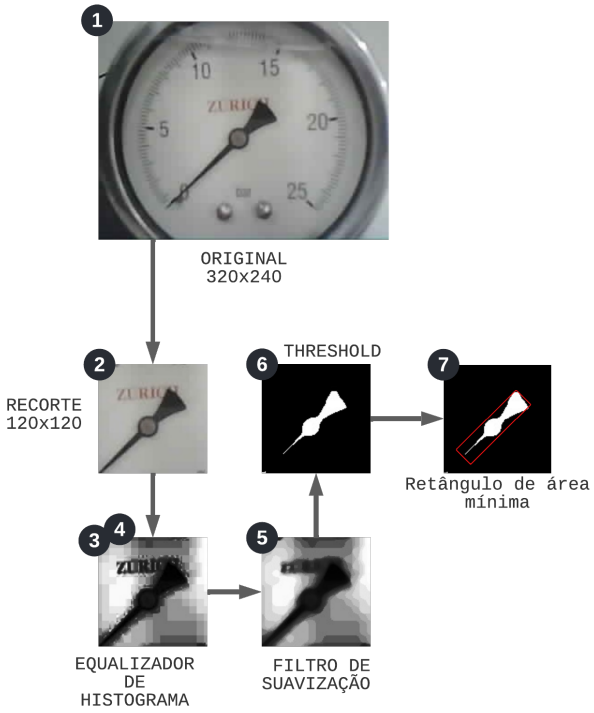


Figura 3. Fluxograma do processamento da imagem.

A Figura 3 apresenta 7 passos realizados para tratar a imagem antes do cálculo do ângulo do ponteiro. A primeira imagem do Figura 3 apresenta o dado (imagem) original com tamanho de 320x240 *pixels*. Os passos 2, 3, 4, 5 e 6 são responsável por gerar a imagem tratada, apresenta no passo 7 da Figura 3, que será utilizada pela função de cálculo do ângulo do ponteiro. O pseudocódigo 0 apresenta um resumo das etapas do *script* Python que executa o processamento da imagem e geração do valor de pressão.

A imagem capturada contém um tamanho de 320x240 *pixels*, colorida e no formato JPEG. O primeiro passo para isolar o ponteiro é de definir a região de interesse (ROI - *"Region of interest"*), esse termo define a região que interessa para aplicar o processamento de imagem, variando de aplicação para aplicação. Nesse caso, a região de interesse é o centro da foto, então é feito um recorte, excluindo as bordas do manômetro e a maior parte das

Algoritmo 1 Sequência de operações

1: **original** ← $opencv.imdecode(img, -1)$
2: **cropped_img** ← $center\_crop(original, (120, 120))$
3: **gray_img** ← $opencv.cvtColor(cropped\_img)$
4: **equal_img** ← $opencv.equalizeHist(gray\_img)$
5: **blur_img** ← $opencv.medianBlur(equal\_img, 7)$
6: **threshold_img** ← $opencv.threshold(blur\_img, sp\_th, 255, opencv.thresh\_binary\_inv)$
7: **contour** ← $opencv.findContours(threshold\_img)$
8: **base_heig, ang** ← $opencv.minAreaRect(contour)$
9: **pressao** ← $getPressure(base\_heig, ang)$

marcações de pressão, como é exibido na Figura 4b. O recorte é executado pela função *center_crop()*.

(a) Imagem original.

(b) Imagem recortada.

Figura 4. Etapa de recorte da área central.

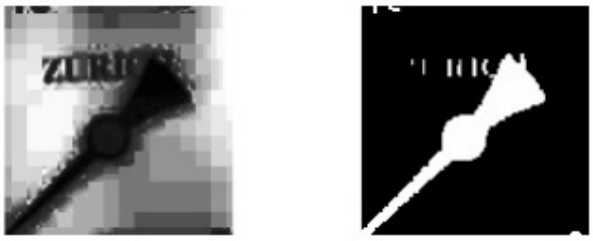
A câmera poderia ficar mais próxima do manômetro e assim evitar esse processo, porém comprometeria a visão do operador, caso fosse necessário obter o valor de pressão manualmente. Além disso, a distância também ajuda no ajuste fino da imagem para centralizar com o manômetro, já que o suporte permite que a câmera seja ajustável, tendo uma rotação de 360°.

Após o recorte, a imagem fica com um novo tamanho de 120x120 *pixels* e então é utilizado uma técnica de segmentação para retirar a cor da imagem, deixando apenas em tons de cinza. Esse processo é executado pela função *cvtColor()* do OpenCV.

A próxima etapa é realizar a equalização do histograma, atenuando o contraste da imagem, tornando a diferença entre as partes claras e escuras mais evidentes. Com esse processo, fica bastante explícito os *pixels* do ponteiro quando comparado com o fundo branco, como é possível observar na Figura 5a. A equalização é executada pela função *equalizeHist()*.

A etapa seguinte é *thresholding*, que consiste em transformar a imagem em binária. Ao final dessa etapa, os *pixels* serão completamente branco ou completamente preto. O algoritmo define um valor constante de intensidade que será comparado aos valores de todos os *pixels* da imagem. Caso a intensidade do pixel seja maior, então ele se torna branco, caso contrário, passará a ser preto. O resultado dessa etapa é apresentado na Figura 5b.

Ainda existem partes brancas resultantes de letras ou marcações da escala do manômetro e não é possível ignorar esse problema, pois vai acabar alterando o valor final da

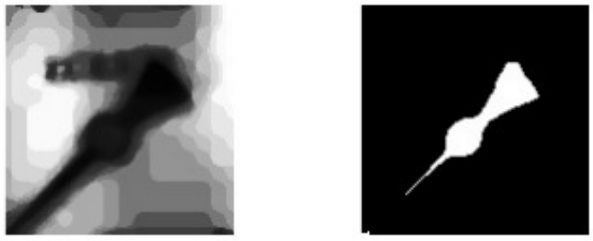


(a) Imagem após equalização.

(b) Imagem após thresholding.

Figura 5. Etapas de equalização e *thresholding*.

pressão. A melhor solução para resolver esse problema é utilizar um filtro linear na imagem antes de realizar o *thresholding*, suavizando as bordas como mostra a Figura 6a. Como as letras e as marcações são muito finas, no momento que ocorre a suavização das bordas, elas acabam perdendo toda a intensidade do preto, ficando assim maior que o valor constante definido no *threshold*. A função *medianBlur()* é responsável por aplicar o filtro.



(a) Imagem após suavização.

(b) Imagem após thresholding.

Figura 6. Etapas de suavização e *thresholding*.

A Figura 6b apresenta o resultado da aplicação do processo de *thresholding* após o filtro de suavização. É possível observar que o ponteiro está completamente isolado na nova imagem. Mesmo a ponta indicadora ficando mais fina, por causa da suavização das bordas, ainda fica claro a inclinação do ponteiro.

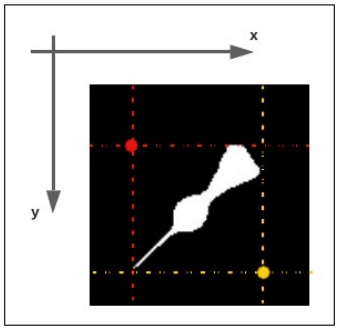


Figura 7. Pontos inicial e final do retângulo de interesse.

Por fim, é definido uma função para calcular o ângulo do ponteiro. Essa função inicia com a criação de um retângulo com uma área mínima que contemple toda a área branca do ponteiro da imagem. O primeiro passo é

definir o contorno da área branca a partir de um ponto inicial (vértice superior esquerdo do retângulo), que é o pixel branco com menor valor para x e y, e um ponto final (vértice inferior direito do retângulo), que é o ponto com maior valor para x e y. Na Figura 7 é possível ver esses dois pontos e como é gerado um retângulo a partir deles.

Figura 8. Retângulo de área mínima.

Com os dois pontos e o retângulo formado, pode-se usar a função *minAreaRect()* da biblioteca OpenCV. Essa função cria um novo retângulo com base na imagem recebida que engloba todo o ponteiro em uma área mínima, como mostra a Figura 8.

Independente do posicionamento do ponteiro, a função sempre calcula o ângulo com base na aresta que gera um ângulo igual ou menor que 90º com o eixo X. A Figura 9 apresenta dois exemplos de retângulos criados pela função *minAreaRect()* e os ângulos calculados pela função.

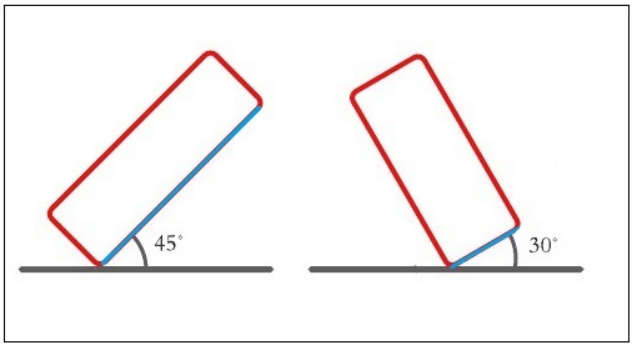


Figura 9. Ângulo do retângulo de área mínima.

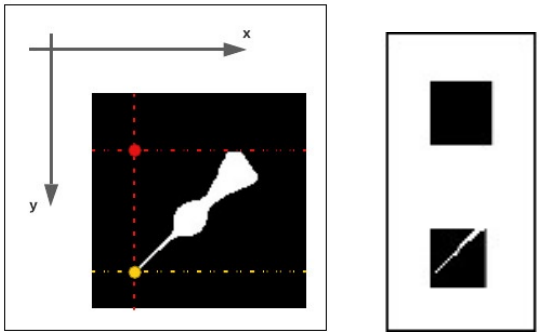
Observa-se na Figura 9 que o retorno da função *minAreaRect()* será um valor de ângulo igual ou menor que 90º. Com isso, é necessário determinar qual quadrante o ponteiro está direcionado.

A próxima etapa do algoritmo é identificar qual o quadrante que o ponteiro está apontando, quadrante 1, 2, 3 ou 4.

A estratégia utilizada inicia com o cálculo do valor do vértice inferior esquerdo do retângulo original. Esse vértice é calculado com o deslocamento do ponto final (vértice inferior direito) para o menor valor do eixo X. A Figura 10a apresenta os dois pontos utilizados na função de identificação da direção do ponteiro.

De acordo com o posicionamento dos dois pontos, é possível afirmar que pelo menos um deles estará exatamente em cima de uma das extremidades do ponteiro. Assim, a próxima etapa realiza um pequeno recorte de 25x25 *pixels* nesses dois pontos para contar a quantidade de *pixels* brancos nessa área. Segundo a norma NBR 14105, a largura da extremidade indicadora do ponteiro não pode ser maior que a divisão de escala, ou seja, ela deve ser bem mais fina, comparada com a outra extremidade.

A Figura 10b apresenta o resultado dos recortes para a Figura 10a. É possível observar que o ponteiro não está em cima do ponto do vértice superior esquerdo, assim sua área está completamente preta. Por outro lado, o ponto do vértice inferior esquerdo apresenta uma parte com *pixels* brancos que representa a presença de uma das extremidades do ponteiro no vértice.



(a) Pontos vértices utilizados para identificar a direção do ponteiro.

(b) Recorte dos dois vértices.

Figura 10. Recortes da estratégia de determinação de quadrante.

O próximo passo é realizar uma contagem de *pixels* brancos dentro da área que apresentou uma extremidade do ponteiro. De acordo com a Figura 10, a extremidade do ponteiro detectada no recorte apresenta cerca de 5% de *pixels* brancos dentro da área de 25x25 *pixels*. Um outro teste de recorte foi realizado com a outra extremidade do ponteiro e foi detectado um valor de 50% de *pixels* brancos dentro da área de 25x25 *pixels*.

O limiar definido para indicar se foi detectado a extremidade mais fina ou mais grossa do ponteiro foi 18%. Ou seja, se existe um recorte com porcentagem maior que 0% e menor que 18% de *pixels* branco, ele é classificado como o vértice que contém a extremidade mais fina. Se o valor de porcentagem de *pixels* brancos for maior que 18% é detectado a extremidade mais grossa do ponteiro.

Com isso, é possível determinar o ângulo do ponteiro e em qual quadrante ele está apontando. O próximo passo é calcular a pressão a partir desse ângulo.

O manômetro usado nesse experimento faz medição de 0 a 25bar dentro de um ângulo de 270°. O cálculo de pressão deve considerar um passo de quantização de 270º dividido por 25 bar, ou seja, um passo de 10.8º/bar. A equação 1 apresenta o cálculo para gerar o valor da pressão a partir do valor do ângulo calculado pela função *minAreaRect()*.

$$pressaoQ1 = (angulo - 45) * 25/270 \quad (1)$$

È possível observar na equação 1 que existe um *offset* de 45º no valor do ângulo. Esse *offset* ocorre porque o valor

de 0 bar no manômetro está defasado 45º do x. Existe um valor de *offset* para cada quadrante (Q1, Q2, Q3 e Q4), como mostram as equações 2, 3 e 4.

$$pressaoQ2 = (angulo + 45) * 25/270 \quad (2)$$
$$pressaoQ3 = (angulo + 135) * 25/270 \quad (3)$$
$$pressaoQ4 = (angulo + 225) * 25/270 \quad (4)$$

Todas essas etapas de cálculo do valor de pressão a partir do valor de ângulo resultante da função *minAreaRect()* foram implementados na função *getPressure()*.

4. RESULTADOS

O dispositivo IoT NodeManometro foi instalado em um manômetro de uma unidade de tratamento de água de uma indústria farmacêutica. Foram coletados dados durante o desenvolvimento do código para teste e durante a instalação física do dispositivo para testar o suporte desenvolvido neste trabalho. Foram coletadas cerca de 500 amostras por dia durante 5 dias.

Com o objetivo de coletar as imagens direto no processo, foi produzido um suporte para o dispositivo NodeManometro, impresso em 3D e específico para os manômetros da planta industrial do laboratório farmacêutico.

A seguir serão apresentados os resultados do processamento de imagem para geração do valor de pressão e do suporte produzido para o dispositivo.

4.1 Geração dos valores de pressão a partir das imagens

Na tabela 1 são apresentados alguns exemplos de posições do ponteiro, assim como a porcentagem de área branca em cada ponto inicial e final (vértice superior e inferior), seguido pelo ângulo calculado pela função *minAreaRect()*. Com esses dados, são empregadas as equações dos quadrantes e definida a pressão indicada pelo ponteiro.

Tabela 1. Posições do ponteiro.

Imagem	Ponto inicial e final	Área branca	Ângulo	Pressão
		0%	45°	0 bar
		4.6%		
		7.8%	48.6°	8.6 bar
		0%		
		7.5%	1.3°	13.1 bar
		53.7%		
		63.8%	2.5°	21.9 bar
		56.9%		

Na terceira imagem da Figura 1, quando o ponteiro está completamente na posição vertical, fica evidente a diferença entre as duas extremidades e a área branca de cada uma.

A última imagem da Figura 1 apresenta o ponteiro completamente na posição horizontal. Nesse caso os dois pontos ficam muito próximos e como a porcentagem do recorte é acima de 18% é possível identificar a extremidade mais grossa do ponteiro.

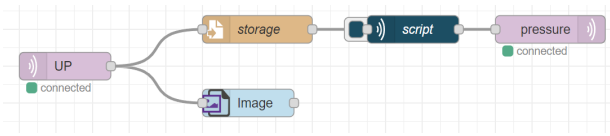


Figura 11. Blocos do Node-Red.

A Figura 11 apresenta a sequência de processamento entre os nós da plataforma Node-Red. Primeiro o bloco de *subscribe* do MQTT para receber a imagem original, enviando para um bloco de visualização da imagem e um bloco de salvar imagem para futuras análises. Então a imagem entra na execução do algoritmo em python, enviando o valor de pressão para o bloco de *publish* em MQTT.

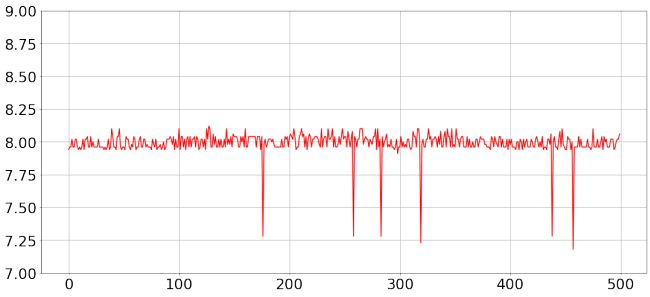


Figura 12. Resultado da amostragem.

O gráfico da Figura 12 apresenta 500 amostras feitas durante um dia nesse mesmo manômetro. Existem 6 pontos que geraram um valor com um erro de quase 0,75 bar, mas que no geral, o valor médio ficou muito próximo de 8 bar.

Tabela 2. Análise de amostragem do manômetro.

Qntd. Amostras	500
Valor Médio	7,9896
Valor de referência	8
Desvio padrão	0,0921

Na tabela 2 é feito uma análise desses dados, onde é apresentado uma média de 7,98 bar dessas amostras. Nessa etapa, o ponto de referência para a pressão é de 8 bar. Mesmo sem o tratamento para esses *outliers* o desvio padrão ainda ficou com valor aproximado de 0,09 bar.

4.2 Suporte para o NodeManometro

O dispositivo desenvolvido deve realizar a medição do valor de pressão de forma não invasiva. Ou seja, a captura da imagem do manômetro não deve influencia na visão dos operadores do processo. Além disso, o dispositivo deve

conseguir monitorar outros manômetros e sua instalação deve ser flexível.

Com o intuito de atender ao conceito de implantação de dispositivos IoT não invasivos, foi projetado e produzido um suporte para o dispositivo que atende aos seguintes requisitos:

- R01: Manter uma distância de 10cm para não comprometer a visão do operador.
- R02: Manter o padrão de conexão com os manômetros do processo.
- R03: Permitir o ajuste de posicionamento da câmera.

O projeto mecânico do suporte iniciou com a peça da braçadeira que irá fixar o suporte nos manômetros. Foi escolhido a braçadeira do tipo D porque garante um fácil encaixe na tubulação da base dos diversos manômetros do processo conforme descrito no requisito 02.

(a) Modelo do suporte em 3D.

(b) Manufatura aditiva do modelo.

Figura 13. Produção do suporte.

Após o projeto da braçadeira, foi projetado a haste que mantém o ESP32-CAM posicionado em frente ao manômetro. O comprimento da haste é de 12cm, garantindo assim o requisito R01. O projeto do suporte do ESP32-CAM permite rotacionar o ESP32-CAM em 360º. A Figura 13a apresentar o modelo do suporte e a Figura 13b o suporte impresso em 3D.

Figura 14. Manômetro operando no processo.

A Figura 14 apresenta o suporte instalado direto no processo com visão direta para um dos manômetros da planta. Foram realizados testes de captura de imagens com o suporte desenvolvido.

5. CONCLUSÃO

Nesse trabalho foi proposto e desenvolvido uma aplicação de um dispositivo IoT com câmera integrada para a captura de imagens de um manômetro atuando em uma indústria farmacêutica. As fotos obtidas passam por várias etapas de processamento de imagem, com o propósito de isolar o ponteiro do manômetro e assim obter a sua inclinação e qual de suas extremidades é a indicadora. Por fim, com esse dado de inclinação é possível calcular qual a pressão que está sendo indicada e enviar esse valor através do protocolo MQTT para o supervisório da empresa.

O suporte da câmera desempenhou de maneira adequada o seu papel de permitir mobilidade de ajuste para a câmera e de mantê-la fixa para as capturas de imagem. Algo a se trabalhar no futuro seria um suporte para as baterias e proteger os fios de alimentação. Já na questão de software, um dos desafios para determinar a veracidade dos dados apontados pelo algoritmo é que a pressão dos manômetros dentro da indústria praticamente não variam durante o dia, pois os processos funcionam de forma estável durante as 24h.

A análise de dados do manômetro demonstrou que não só a acurácia, mas também a precisão estava boa. Com um desvio padrão próximo de 0,09, os valores se mantiveram constantes, salvo 6 pontos *outliers* que deram um erro de 0,75 bar de diferença com a referência, é um erro de quase 10%. No caso em questão, é possível que algum evento externo tenha alterado a iluminação da imagem, como a sombra de uma pessoa por exemplo, mas é difícil de analisar sem ter a imagem da amostra específica. De toda forma, ainda existe a possibilidade de tratar esses dados, trabalhando com média móvel ou filtros por exemplo.

AGRADECIMENTOS

Ao Núcleo de Pesquisa em Alimentos e Medicamentos (NUPLAM) da UFRN pela contribuição na pesquisa.

REFERÊNCIAS

Câmara, G., Souza, R.C.M., Freitas, U.M., e Garrido, J. (1996). Spring: Integrating remote sensing and gis by object-oriented data modelling. *Computers & graphics*, 20(3), 395–403.

Gokhale, P., Bhat, O., e Bhat, S. (2018). Introduction to iot. *International Advanced Research Journal in Science, Engineering and Technology*, 5(1), 41–44.

Kagermann, H., Helbig, J., Hellinger, A., e Wahlster, W. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group.* Forschungsunion.

Kassab, W. e Darabkh, K.A. (2020). A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*, 163, 102663.

Persechino, A. e Albuquerque, M.P. (2015). Processamento digital de imagens: conceitos fundamentais. Centro Brasileiro de Pesquisas Físicas (CBPF), Rio de Janeiro, RJ.