Acesso Remoto a Dados de Sensores com ESP32 e Ngrok, monitoramento de vazão com Servidor Web Local e Túnel Seguro

Carlos Eduardo Luyo Gonçalves[1], Eduardo De Paiva Martins Filho[2], Guilherme Machado Ribeiro França[3], Victor de Melo Lima Evangelista[4], Welliton Borges Araújo[5]

Matrícula: 20231003800255[1]
Matrícula: 20231003800042[2]
Matrícula: 20232003800040[3]
Matrícula: 20232003800032[4]
Matrícula: 20222003800060[5]

Resumo

Nesse relatório reporta-se a atividade desenvolvida, na disciplina ENG1119 Redes e Aplicações IOT, pelos discentes da PUC que consiste em controlar uma variável de processo, vazão, da Estação de Fluxo LabVolt Modelo 3502 presente nos laboratórios da instituição de ensino superior. O controle será feito pela utilização de um servidor web embarcado no ESP32. Além disso, será integrado a ferramenta ngrok para acesso remoto à interface do ESP32.

Palavras-chave: ESP32, MicroPython, Ngrok.

1 Introdução

Os sistemas de controle são fundamentais para a sociedade contemporânea. Isso é visto nos carros, aviões, elevadores e inúmeras outras tecnologias que dependem do conhecimento dessa área, sendo considerados indispensáveis para o cotidiano da população. Um sistema de controle é um conjunto de processos (ou plantas) construídos com o objetivo de se obter uma saída desejada com desempenho desejado, dada uma entrada especificada. Dessa forma, controlar um sistema significa obter uma saída que atenda às expectativas do usuário para uma determinada entrada.

Neste experimento, o controle da planta será feito por uma interface web hospedada no ESP32. O valor ajustado pelo usuário (via slider) será convertido em sinal analógico através do pino DAC, que é conectado ao terminal de entrada analógica do inversor de frequência da planta didática de vazão modelo 3502. A página web exibirá em tempo real:

- O valor DAC (0 a 255).
- A frequência aplicada (0 a 66 Hz).
- A vazão estimda (0 a 38 LPM).

1.1 Descrição da planta didática Lab - Volt Modelo 3502

A planta de vazão apresentada consiste em um sistema de escoamento de fluido (água) realizado através de tubulações de 3/4 de polegada com percurso indicado no fluxograma da figura 1.

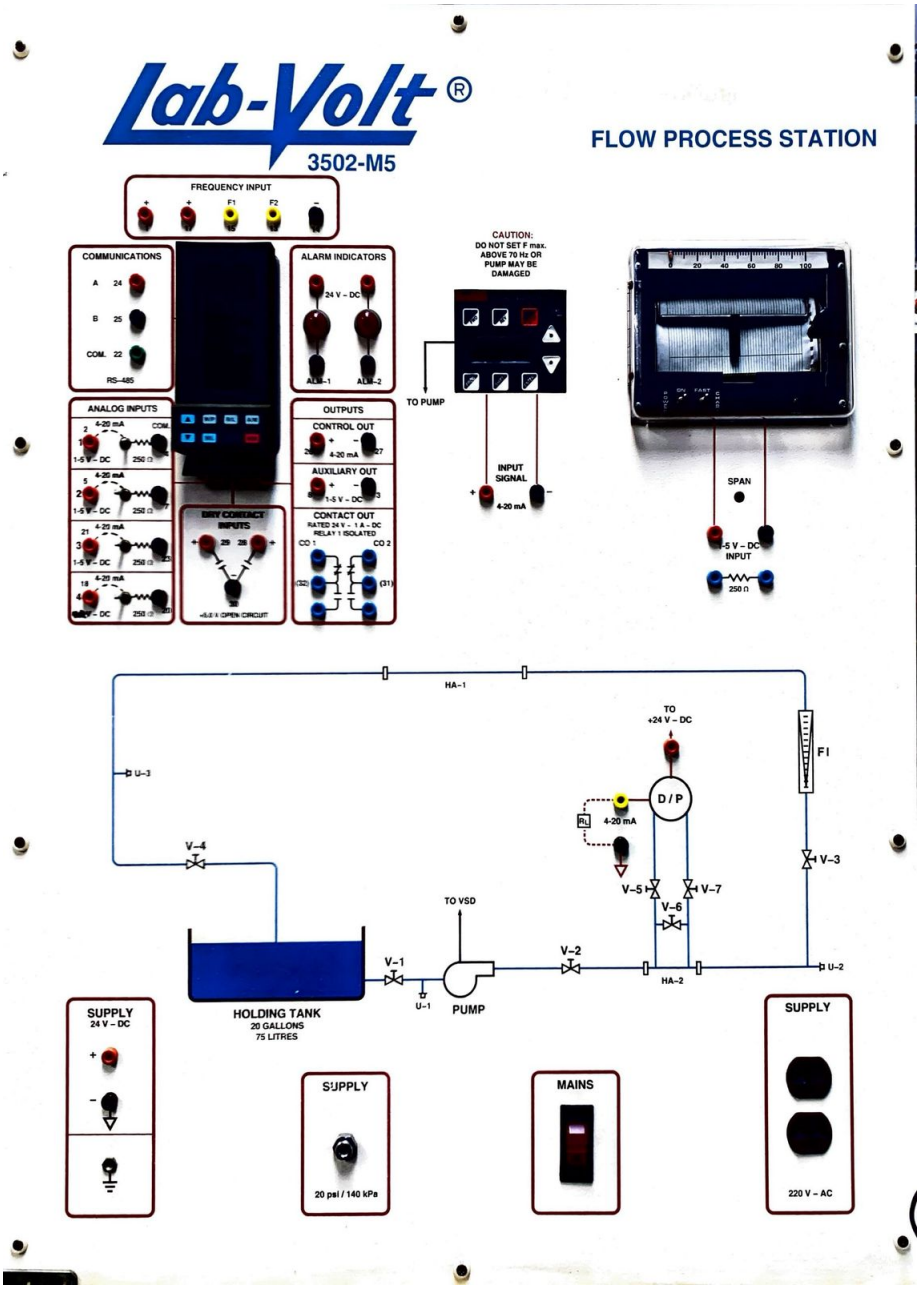

Figura 1: Fluxograma da planta de vazão usada para o experimento feito durante a aula da disciplina ELT1119 Redes IoT.

O projeto inclui componentes essenciais para o monitoramento e controle do fluxo, destacando-se:

- Sensor de Vazão tipo Tubo Venturi: Responsável pela medição precisa da vazão do fluido, aproveitando a variação de pressão para determinar a taxa de fluxo.
- Transmissor da Foxboro: Equipamento utilizado para converter as medições do sensor em sinais elétricos, transmitindo os dados para o sistema de controle.
- Inversor de Frequência: Dispositivo empregado para ajustar a velocidade da bomba ou motor, permitindo o controle da vazão de acordo com as demandas operacionais da planta.

2 Materiais e Métodos

A realização do experimento precisou dos seguintes materiais listados:

- Placa de desenvolvimento ESP32-DeviKitC-V4;
- Computador com ambiente de desenvolvimento Thonny instalado e ngrok instalado e autenticado;
- Celular com modo roteador de Wi-fi;
- Cabo USB para a comunicação entre o ESP32 e o computador;
- Planta vazão didática modelo 3502 Lab-Volt pertencente ao laboratório de instrumentação industrial da PUC Goiás;

O experimento foi desenvolvido da seguinte forma, primeiramente, deve-se conectar o ESP32 à rede Wi-Fi gerada pelo celular, utilizado como roteador. Em seguida, é necessário programar o ESP32 com o código responsável pelo controle via DAC e pelo servidor web, conforme indicado no apêndice A.

Na etapa seguinte, configura-se o serviço ngrok. Para isso, deve-se conectar um notebook à mesma rede do ESP32, abrir o terminal (prompt de comando) e executar o seguinte comando: ‘ngrok http 192.168.x.x:80‘, substituindo o endereço "x.x"pelo IP atribuído ao ESP32, o qual pode ser verificado no terminal do Thonny ou do PuTTY. Após a execução do comando, será gerada uma URL pelo ngrok, a qual deve ser copiada (por exemplo, https://abc123.ngrok.io). Essa URL permitirá o acesso remoto à interface web de controle.

Na etapa de controle da vazão, acessa-se a interface de duas maneiras: pela rede local, por meio do endereço IP (ex: http://192.168.x.x), ou remotamente pela URL do ngrok, disponível para qualquer local com acesso à internet. Dentro da interface, utiliza-se um slider (barra deslizante) para alterar o valor enviado ao DAC, variando de 0 a 255.

Por fim, deve-se observar a resposta da planta que envolve monitorar o medidor de vazão (FI) e verificar a estabilidade do nível no tanque e acompanhar o comportamento dos indicadores na interface web.

3 Resultados

Com base no código montado, apresentado no apêndice A, foi obtido, no ambiente de programação Tonny, a página HTTP presente na figura 2. Além do código montado, foi feito o Hardware ilustrado nas figuras presentes no apêndice B. Após a energização da planta 1, foi feito a rodagem do código considerando as devidas conexões de fios e placas conversoras de sinais analógicos, de (4-20mA) a (0-3,3V). O resultado obtido foi que a página criada controlava a vazão indicada pelo sensor presente na planta sem erros grosseiros. Quando era indicado a vazão de 30LPM, a frequência da bomba do motor da planta era alterada por im sinal elétrico vindo do micro-controlador ESP32, até chegar no ponto estabelecido (em inglês set-point).

Ademais, foi possível que um usuário externo ao servidor controlasse a planta de qualquer local que houvesse sinal de internet. Isso foi possível graças ao Ngrok que realizou o tunelamento para acesso e controle das variáveis presentes na página HTTP. Consequentemente, nota-se que o Ngrok permite o controle de forma remota.

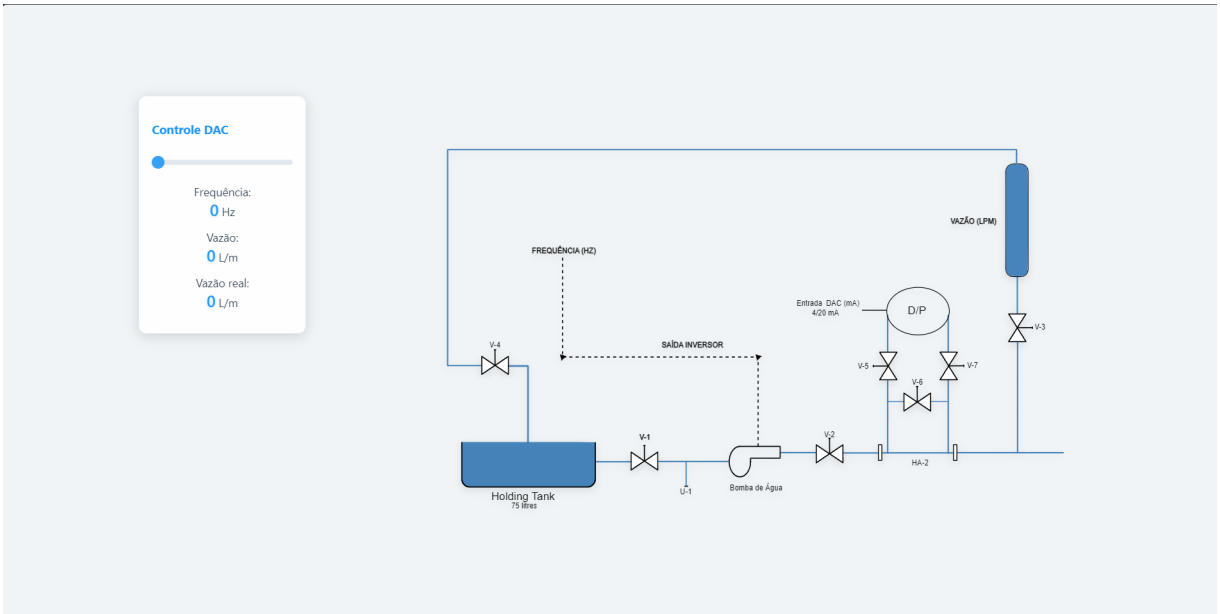

Figura 2: Visualização da página HTTP, resultado do programa do apêndice A.

4 Discussões

A resposta da planta é compatível com os valores exibidos na interface remota, ou seja, a visualização dos dados em tempo real reflete adequadamente o comportamento do sistema. Os atrasos perceptíveis na comunicação são mínimos, geralmente inferiores a dois segundos, o que mantém a usabilidade e o acompanhamento do processo de forma aceitável em aplicações não críticas.

Quanto à segurança, o controle remoto de uma planta real via web exige atenção a riscos como invasões, acesso não autorizado e falhas na integridade dos dados. Por isso, é necessário adotar medidas como autenticação segura, criptografia e redes privadas. O uso do ngrok ou de outras ferramentas de tunelamento seguro pode ser vantajoso em testes, protótipos ou aplicações de pequeno porte, onde uma solução rápida e prática é suficiente.

Em relação à modelagem da planta, a relação entre o DAC e a vazão não é completamente linear, apresentando um comportamento que pode ser analisado por meio de gráficos obtidos experimentalmente. A variação da frequência influencia diretamente a dinâmica do processo, afetando o tempo de resposta e a estabilidade do sistema, o que torna essencial um controle cuidadoso para garantir eficiência operacional.

5 Conclusão

A atividade desenvolvida foi bem-sucedida na aplicação prática de conceitos de IoT e controle de processos, integrando hardware, software e acesso remoto com o uso do ngrok. A implementação do controle de vazão por meio do ESP32 demonstrou funcionalidade satisfatória, apresentando resposta estável e confiável da planta ao comando via interface web. O sistema permitiu monitoramento em tempo real com baixo tempo de latência, além de possibilitar a atuação remota com relativa segurança. Os resultados obtidos validam a viabilidade do experimento proposto e evidenciam o potencial do uso de microcontroladores conectados à internet para controle de variáveis em sistemas físicos reais.

Apêndice A

Código usado no controle do experimento da disciplina ELT1119 - Redes IOT

Código micropython feito para o controle da planta

```
import network
import socket
import ujson
import time
from utime import sleep
from machine import Pin, DAC, ADC
from _thread import start_new_thread

vazao_adc_global = 0.0

ADC1 = ADC(Pin(32))
ADC1.atten(ADC.ATTN_11DB)
dac = DAC(Pin(25))
a = 1200

def atualizar_vazao_continuamente():
    global vazao_adc_global
    while True:
        adc1 = ADC1.read()
        if adc1 >= a:
            vazao_adc_global = (38 / (4095-a)) * (adc1 - a)
        else:
            vazao_adc_global = 0.0
        sleep(0.5)
        print(adc1)
        print("Vazão:", vazao_adc_global)

SSID = "redr"
PASSWORD = "senha123"

sta = network.WLAN(network.STA_IF)
sta.active(True)
sta.connect(SSID, PASSWORD)

print("Conectando ao Wi-Fi...")
start = time.time()
while not sta.isconnected():
    if time.time() - start > 10:
        raise RuntimeError("Erro ao conectar-se ao Wi-Fi.")
    time.sleep(0.1)

print("Conectado! IP:", sta.ifconfig()[0])

start_new_thread(atualizar_vazao_continuamente, ())

```

```
def html_page():
    return """
<!DOCTYPE html>
<html lang="pt">

<head>
    <title>Sistema de Controle Hidráulico</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
        body {
            margin: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background: #f0f4f7;
            font-family: 'Segoe UI', sans-serif;
        }

        .main-container {
            position: relative;
            width: 800px;
            height: 600px;
        }

        .system-svg {
            width: 150%;
            height: 200%;
            background: url('https://raw.githubusercontent.com/AdmiralR
            ↪  achmaninov/testingsvg/d03d23ea4dcd2a3e15048128845e1d4c0
            ↪  fa63e26/svgfinal2.svg');
            background-size: contain;
            background-repeat: no-repeat;
            filter: drop-shadow(0 4px 6px rgba(0, 0, 0, 0.1));
        }

        .control-panel {
            position: fixed;
            top: 15%;
            right: 75%;
            background: rgba(255, 255, 255, 0.95);
            padding: 20px;
            border-radius: 12px;
            box-shadow: 0 4px 20px rgba(0, 0, 0, 0.15);
            width: 220px;
            backdrop-filter: blur(5px);
            animation: panelEntry 0.8s ease-out;
        }
```

```
        .dac-slider {
            width: 100%;
            margin: 15px 0;
            -webkit-appearance: none;
            height: 8px;
            border-radius: 4px;
            background: #e0e5ec;
            outline: none;
            opacity: 0.9;
            transition: opacity 0.2s;
        }

        .dac-slider::-webkit-slider-thumb {
            -webkit-appearance: none;
            width: 20px;
            height: 20px;
            border-radius: 50%;
            background: #2196F3;
            cursor: pointer;
            transition: transform 0.2s;
        }

        .value-display {
            text-align: center;
            margin: 15px 0;
            font-size: 1.1em;
            color: #2c3e50;
        }

        .value-number {
            font-size: 1.4em;
            font-weight: 600;
            color: #2196F3;
            text-shadow: 0 2px 4px rgba(33, 150, 243, 0.2);
        }

        @keyframes panelEntry {
            from {
                transform: translateY(20px);
                opacity: 0;
            }

            to {
                transform: translateY(0);
                opacity: 1;
            }
        }
```

```
        .status-led {
            width: 12px;
            height: 12px;
            border-radius: 50%;
            display: inline-block;
            margin-left: 10px;
            animation: pulse 1.5s infinite;
        }

        @keyframes pulse {
            0% {
                box-shadow: 0 0 0 0 rgba(33, 150, 243, 0.5);
            }

            70% {
                box-shadow: 0 0 0 8px rgba(33, 150, 243, 0);
            }

            100% {
                box-shadow: 0 0 0 0 rgba(33, 150, 243, 0);
            }
        }
        <script>
            function atualizarVazaoReal() {
                fetch('/vazao2')
                    .then(response => response.json())
                    .then(data => {
                        document.getElementById('vazao_adc_global').tex
                        ↪  tContent =
                        ↪  data.vazao_real.toFixed(1);
                    })
                    .catch(error => {
                        console.error("Erro ao obter vazão real:",
                        ↪  error);
                    });
            }

            setInterval(atualizarVazaoReal, 1000); // Atualiza a cada 1
            ↪  segundo
        </script>

        </script>
        <style>
    /* ... (outros estilos) ... */

    /* ========================================== */
    /* === AJUSTES ESPECÍFICOS PARA MODO RETRATO === */
```

```
    @media (orientation: portrait) {
        .main-container {
            width: 100vw!important;
            height: 100vh!important;
            padding: 0!important;
        }

        .system-svg {
            width: 100%!important;
            height: 60vh!important;  /* Reduz altura da imagem */
            position: absolute;
            top: 5vh;
        }

        .control-panel {
            width: 85%!important;
            right: 50%!important;
            left: auto!important;
            top: 65vh!important;  /* Posiciona abaixo da imagem */
            transform: translateX(50%)!important;  /* Centraliza */
            padding: 15px!important;
        }

        /* Aumenta elementos para toque */
        .dac-slider {
            height: 10px!important;
            margin: 25px 0!important;
        }

        .dac-slider::-webkit-slider-thumb {
            width: 28px!important;
            height: 28px!important;
        }

        .value-display {
            font-size: 1.1em!important;
            margin: 20px 0!important;
        }
    }
     @media (max-width: 600px) {
        .main-container {
            width: 100%!important;
            height: auto!important;
            padding: 10px;
        }

        .control-panel {
            width: 90%!important;
```

```
            right: 5%!important;
            left: auto!important;
            top: 10%!important;
            transform: none!important;
        }

        .system-svg {
            width: 100%!important;
            height: auto!important;
        }

        .value-number {
            font-size: 1.2em!important;
        }
    }
    </style>
</head>

<body>
    <div class="main-container">
        <div class="system-svg"></div>

        <!-- Painel de Controle Flutuante -->
        <div class="control-panel">
            <h3 style="color: #2196F3; margin-bottom: 20px;">
                Controle DAC
                <span class="status-led"></span>
            </h3>

            <input type="range" min="0" max="255" value="0"
             ↪  class="dac-slider" id="dacSlider"
                oninput="updateDAC(this.value)">

            <div class="value-display">
                <div>Frequência:</div>
                <span class="value-number" id="freq">0</span> Hz
            </div>

            <div class="value-display">
                <div>Vazão:</div>
                <span class="value-number" id="vazao">0</span> L/m
            </div>
            <div class="value-display">
                <div>Vazão real:</div>
                <span class="value-number"
                 ↪  id="vazao_adc_global">0</span> L/m
            </div>
        </div>
```

```
    </div>

    <script>
        function atualizarVazaoReal() {
                fetch('/vazao2')
                    .then(response => response.json())
                    .then(data => {
                        if (data.vazao_real !== undefined) {
                            document.getElementById('vazao_adc_global')
                             ↪  .textContent =
                             ↪  data.vazao_real.toFixed(1);
                        } else {
                            console.error("Vazão real não encontrada.");
                        }
                    })
                    .catch(error => {
                        console.error("Erro ao obter vazão real:",
                         ↪  error);
                    });
            }

            setInterval(atualizarVazaoReal, 1000); // Atualiza a cada 1
             ↪  segundo

        let updateTimeout;
        const statusLed = document.querySelector('.status-led');

        function updateDAC(value) {
            // Feedback visual imediato
            statusLed.style.background = '#ff9800';
            document.getElementById('freq').textContent = (value / 255 *
             ↪  60).toFixed(1);
            document.getElementById('vazao').textContent = (value / 255
             ↪  * 38).toFixed(1);

            // Debounce para evitar flood de requisições
            clearTimeout(updateTimeout);
            updateTimeout = setTimeout(() => {
                const xhr = new XMLHttpRequest();
                xhr.open("GET", "/update?dac=" + value, true);

                xhr.onload = function () {
                    statusLed.style.background = '#4CAF50';
                    const response = JSON.parse(xhr.responseText);
                    document.getElementById('freq').textContent =
                     ↪  response.freq.toFixed(1);
                    document.getElementById('vazao').textContent =
                     ↪  response.vazao.toFixed(1);
```

```
                };

                xhr.onerror = function () {
                    statusLed.style.background = '#f44336';
                };

                xhr.send();
            }, 150);
        }
    </script>
</body>

</html>
"""
addr = socket.getaddrinfo("0.0.0.0", 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
print("Servidor HTTP iniciado em:", addr)

while True:
    try:
        conn, addr = s.accept()
        request = conn.recv(1024).decode()
        if "/vazao2" in request:
            try:
                # A variável vazao_adc_global já está sendo atualizada
                ↪  no thread
                resp_json = ujson.dumps({"vazao_real":
                ↪  vazao_adc_global})
                conn.send("HTTP/1.1 200 OK\r\nContent-Type:
                ↪  application/json\r\n\r\n")
                conn.send(resp_json)
                conn.close()
                continue
            except Exception as e:
                print("Erro ao enviar vazão real:", e)
                conn.send("HTTP/1.1 500 Internal Server Error\r\n\r\n")
                conn.close()
                continue

        if "/update?dac=" in request:
            try:
                val = int(request.split("/update?dac=")[1].split("
                ↪  ")[0])
                val = max(0, min(255, val))
                dac.write(val)
                freq = (val / 255) * 60
```

```
                vazao = (val / 255) * 38
                resp_json = ujson.dumps({"freq": freq, "vazao": vazao})
                conn.send("HTTP/1.1 200 OK\r\nContent-Type:
                 ↪  application/json\r\n\r\n")
                conn.send(resp_json)
                conn.close()
                continue
            except Exception as e:
                print("Erro no update:", e)
        header = "HTTP/1.1 200 OK\r\nContent-Type: text/html;
         ↪  charset=UTF-8\r\n\r\n"
        response = header + html_page()
        conn.send(response.encode('utf-8'))
        conn.close()

    except Exception as e:
        print("Erro geral:", e)
        conn.close()
```

Apêndice B

Hardware feito em sala de aula para avaliação da disciplina de ELT1119 - Redes IOT.

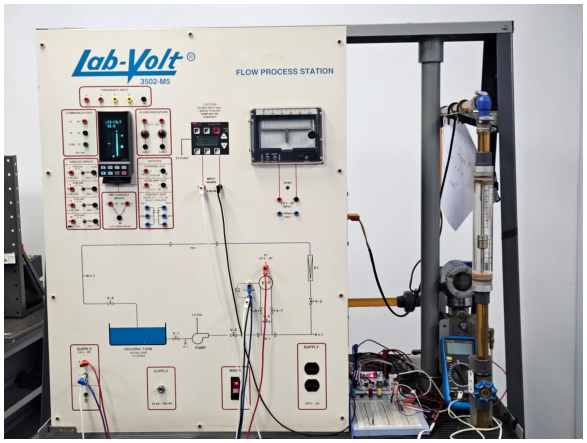

Figura 3: MOntagem do hardware feito em sala de aula.

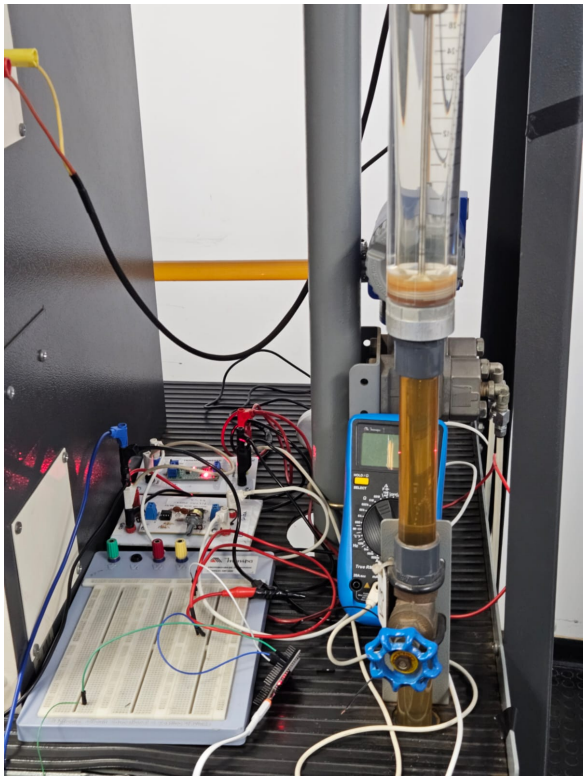

Figura 4: Montagem do hardware feita em sala de aula.