doi.org/10.51891/rease.v11i7.20433

MONITORAMENTO E CONTROLE DE SISTEMAS HIDRÁULICOS UTILIZANDO SENSORES DE VAZÃO, COMUNICAÇÃO RF E INTERFACE WEB

MONITORING AND CONTROL OF HYDRAULIC SYSTEMS THROUGH FLOW SENSORS, RF COMMUNICATION, AND WEB-BASED INTERFACE

Pedro Leonardo Silva da Cruz[1]
Alex Franco Ferreira[2]

RESUMO: Este artigo apresenta o desenvolvimento de um sistema de monitoramento e controle de sistemas hidráulicos com base no uso de sensores de vazão, comunicação sem fio por rádio frequência (nRF24L01) e interface web embarcada. A proposta visa oferecer uma alternativa de baixo custo e fácil implementação para aplicações residenciais ou rurais, com foco na automação do abastecimento de reservatórios de água. O sistema é composto por dois nós: o *Node_A*, baseado em Arduino, realiza a leitura dos sensores de nível e de vazão e controla o acionamento de uma bomba d'água; e o *Node_B*, com um ESP32, recebe os dados via nRF24L01 em cada nó de comunicação e disponibiliza as informações em tempo real através de um *Webserver* local. Os testes demonstraram a confiabilidade da comunicação entre os dispositivos, a precisão na medição da vazão e a eficiência do controle automático do sistema. O trabalho contribui com soluções práticas para o uso racional da água, podendo ser expandido para cenários de *IoT* e agricultura inteligente.

Palavras-chave: Arduino. ESP32. nRF20L01, *Webserver*.

ABSTRACT: This article presents the development of a hydraulic monitoring and control system using flow sensors, wireless RF communication (nRF24L01), and an embedded web interface. The proposed system aims to provide a low-cost and easily deployable alternative for residential or rural water-supply automation. The system consists of two nodes: Node_A, based on Arduino, reads level and flow sensors and controls a water pump; and Node_B, based on ESP32, receives data via nRF24L01 and displays real-time information through a local Webserver. Tests demonstrated reliable communication, accurate flow readings, and effective automatic control. The solution contributes to sustainable water use and can be expanded to IoT and smart-farming scenarios.

Keywords: Arduino. ESP32. nRF20L01, *Webserver*.

[1]Engenheiro eletricista, autor, discente do curso de pós-graduação em automação industrial da Universidade de Vassouras.
[2]Engenheiro eletricista, orientador, docente do curso de pós-graduação em automação industrial da Universidade de Vassouras.

INTRODUÇÃO

A crescente demanda por sistemas inteligentes de automação hídrica tem impulsionado o desenvolvimento de soluções acessíveis e eficientes, especialmente voltadas ao controle de abastecimento de água, irrigação e aplicações residenciais. Neste cenário, este trabalho propõe a implementação de um sistema distribuído que integra sensores de nível tipo chave boia e sensores de vazão com microcontroladores, permitindo o controle de uma bomba hidráulica e o monitoramento em tempo real por meio de uma interface *Web*. O nó de controle é baseado na plataforma Arduino, responsável por acionar a bomba de acordo com os estados dos sensores de nível e pela medição da vazão da água, enquanto o nó de supervisão utiliza o ESP32, encarregado de receber os dados via comunicação sem fio e exibi-los em um *Webserver* local.

A comunicação entre os nós é realizada por módulos nRF24L01+, operando na frequência de 2.4 GHz, cuja eficiência, robustez e baixo consumo energético têm sido comprovados em aplicações industriais e biomédicas, inclusive em projetos com alimentação via *PoF* (*Power over Fiber*), conforme demonstrado por Souza et al. (2023). A adoção de sistemas distribuídos com comunicação sem fio tem ganhado destaque em cenários industriais e rurais, por oferecerem baixo custo, alta confiabilidade e integração simplificada com supervisórios locais. Segundo Silva e Ferreira (2023), a comunicação remota contribui diretamente para a produtividade e tomada de decisões nos processos, o que se reflete também em sistemas residenciais como o aqui proposto, capaz de operar eficientemente mesmo em locais com infraestrutura limitada.

O presente trabalho propõe o desenvolvimento de um sistema distribuído de automação voltado para o controle de bomba hidráulica, com aplicação em ambientes residenciais e rurais. A proposta visa oferecer uma solução de baixo custo, de fácil implementação e com infraestrutura mínima, capaz de automatizar o abastecimento de reservatórios de água a partir de sensores de nível e de vazão. Busca-se também estabelecer um modelo funcional que possa ser replicado em locais onde não há acesso a tecnologias de automação complexas ou rede elétrica convencional confiável, promovendo maior eficiência no uso da água e possibilitando o monitoramento em tempo real do sistema através de dispositivos conectados.

Nesse contexto, estratégias de automação baseadas em sistemas distribuídos têm se mostrado eficazes mesmo em cenários com recursos limitados, especialmente quando utilizam arquiteturas energicamente autônomas. Embora o foco deste projeto seja hidráulico, observa-se a tendência de uso de microcontroladores como o Arduino Nano integrados ao transceptor

nRF24L01+ também em aplicações industriais e biomédicas, evidenciando a versatilidade dessa abordagem (MAHMOOD et al., 2020). O uso do módulo nRF24 em ambos os nós permite uma troca de dados sem fio confiável e com baixo consumo energético, já comprovada em sistemas embarcados de monitoramento em tempo real, contribuindo para robustez e autonomia operacional em ambientes remotos.

A metodologia adotada compreende a construção de dois nós interligados por comunicação sem fio utilizando módulos nRF24L01+PA+LNA, configurados para operar em canal específico e taxa de transmissão reduzida para maior robustez. O *Node_A*, baseado em Arduino, realiza a leitura de chaves boia para detecção do nível da cisterna e da caixa d'água, bem como a medição da vazão de água por meio de um sensor tipo Hall (YF-S201). Com base nessas leituras e no modo de operação selecionado (manual ou automático), o sistema aciona ou desliga uma bomba de água por meio de um relé. O *Node_B*, implementado com ESP32, recebe os dados enviados via rádio frequência, processa-os e os disponibiliza em uma interface web embarcada acessível por navegadores em rede local. Esse modelo elimina a necessidade de sistemas SCADA proprietários, oferecendo uma alternativa enxuta, eficiente e funcional para o monitoramento de sistemas hidráulicos.

MATERIAIS E MÉTODOS

Foram escolhidos cuidadosamente os materiais fundamentais para a construção do protótipo, os quais estão organizados em uma lista detalhada a seguir, constituindo o foco central deste estudo. A escolha de cada componente foi feita de forma estratégica, buscando maximizar a precisão controles e da medição, a eficiência no processamento dos dados e a viabilidade de se implementar um sistema embarcado de baixo custo. Na sequência, são apresentados os elementos principais que viabilizaram a concretização do projeto, abrangendo desde os microcontroladores até os sensores, a estrutura de hardware e a interface de visualização. O sistema foi dividido em dois nós de comunicação:

Node_A (Arduino Nano) – Responsável pela leitura dos sensores e controle da bomba:

- 2 Chaves seletoras para modo manual/automático e liga/desliga no modo manual;
- 1 Relé de 2 canais da marca *Mingwu Electronics* (compatível com 5 V);
- 2 Chaves boias para detecção de nível (cisterna e caixa d'água);
- 1 Sensor de vazão YF-S201 (baseado em efeito Hall);
- 1 Placa ilhada e perfurada de 18 linha e 24 colunas;

- 4 LED's (Verde, Laranja, Vermelho, Branco);
- 1 Rádio Transceptor nRF24L01+PA+LNA;
- 1 Capacitor eletrolítico de 100μF;
- 8 Bordes de conexões;
- 1 Arduino Nano.

Node_B (ESP32) – Responsável pela recepção dos dados e visualização:

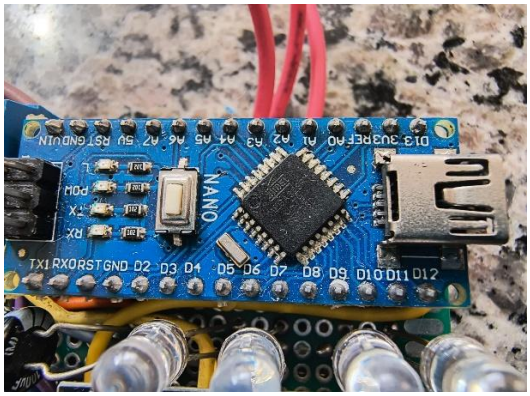
- 1 Placa ilhada e perfurada de 18 linha e 24 colunas;
- 1 Rádio Transceptor nRF24L01+PA+LNA;
- 1 Módulo ESP32 WROOM-32D.

PRINCIPAIS COMPONENTES DO PROJETO (*NODE_A e NODE_B*)

A plataforma Arduino configura-se como uma alternativa eficiente e econômica para o desenvolvimento de soluções em automação residencial. Por ser uma tecnologia de código aberto, tanto em termos de hardware quanto de software, ela permite ampla adaptabilidade e utilização por diversos desenvolvedores e aplicações (Arduino.CC, 2018).

Com base nas necessidades do presente projeto, optou-se pela adoção do modelo Arduino NANO, cuja escolha se justificou por sua reconhecida eficiência e pela presença de todos os recursos técnicos indispensáveis. Além disso, o Arduino NANO, ilustrado na Figura 1, sobressai-se por seu custo reduzido e facilidade de aquisição, aspectos fundamentais para assegurar a viabilidade prática da implementação do protótipo.

Figura 1: arduino nano



Fonte: autor, 2025

O nRF24L01+PA+LNA é a solução adotada neste projeto para realizar a comunicação entre os microcontroladores Arduino e ESP32. Trata-se de um transceptor de rádio frequência que opera na faixa de 2.4 GHz, desenvolvido pela empresa *Nordic Semiconductor*, amplamente utilizado em aplicações de Internet das Coisas (IoT) e automação.

A versão equipada com *PA (Power Amplifier)* e *LNA (Low Noise Amplifier)* oferece desempenho aprimorado, permitindo maior alcance de transmissão e sensibilidade de recepção, especialmente útil em ambientes com obstáculos físicos ou interferências eletromagnéticas. Conforme o fabricante, o módulo suporta taxas de transferência de até 2 Mbps e consumo energético reduzido, além de possibilitar a comunicação em múltiplos canais com baixa latência (NORDIC SEMICONDUCTOR, 2013). A presença de antena externa integrada e amplificadores dedicados torna essa versão particularmente vantajosa para aplicações que exigem estabilidade de sinal em ambientes rurais ou residenciais com estrutura limitada.

No contexto do sistema proposto, o uso do nRF24L01+PA+LNA foi essencial para garantir a troca de dados confiável entre os nós de controle e supervisão — *Node_A* (Arduino) e *Node_B* (ESP32). A escolha desse módulo, ilustrado na Figura 2, priorizou o equilíbrio entre custo, alcance e consumo energético em detrimento de tecnologias como LoRa ou Wi-Fi, que, embora eficientes, envolvem maior complexidade e custo. Estudos recentes reforçam essa decisão: Wu et al. (2017) demonstraram o uso bem-sucedido do nRF24L01 em sistemas industriais com transmissão bidirecional e baixo consumo, alcançando até 300 metros em campo aberto. Da mesma forma, Dialani (2023) validou sua confiabilidade em arquiteturas com controle bilateral e comunicação sincronizada entre microcontroladores, mesmo em sistemas de transferência de energia sem fio, evidenciando a robustez e versatilidade do módulo nRF24L01+ para aplicações em automação distribuída.

Figura 2: nrf24l01+PA+LNA

Fonte: AUTOR, 2025

O YF-S201 é um sensor de vazão de baixo custo amplamente utilizado em sistemas de medição de líquidos, baseado no efeito Hall, um fenômeno físico em que a presença de um campo magnético em um condutor gera uma diferença de potencial perpendicular à corrente

elétrica. O sensor possui uma hélice interna que gira com a passagem da água, acionando um pequeno ímã que, ao passar diante do sensor Hall, gera pulsos elétricos proporcionais à taxa de fluxo. Cada pulso corresponde a uma fração de volume, permitindo a conversão em litros por minuto (L/min) por meio de uma constante de calibração fornecida pelo fabricante, geralmente em torno de 7,5 pulsos por litro (COMPONENTS101, 2021). A leitura dos pulsos pode ser facilmente realizada por microcontroladores como Arduino, utilizando interrupções para maior precisão temporal.

No sistema proposto, o sensor YF-S201 foi acoplado à tubulação de saída da bomba controlada pelo *Node_A* (Arduino), permitindo o monitoramento em tempo real do fluxo de água direcionado à caixa d'água. O microcontrolador contabiliza os pulsos gerados a cada segundo e aplica a fórmula para determinar a vazão instantânea em litros por minuto. Esse valor é incluído na "*struct*" de dados enviada via rádio frequência ao *Node_B* (ESP32), que exibe as informações em uma interface *Web*. A escolha desse sensor, como ilustrado na Figura 3, foi motivada pela sua facilidade de integração, baixo custo e boa relação entre precisão e robustez, mesmo em aplicações residenciais ou rurais com limitações de infraestrutura.

Figura 3: YF-S201

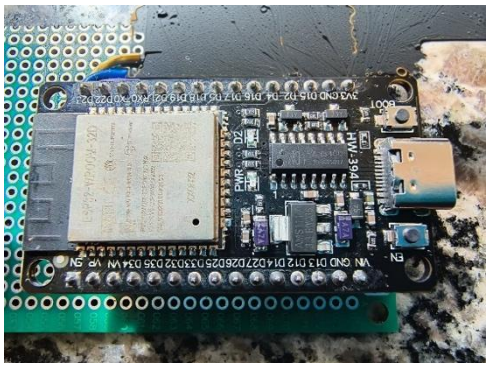
Fonte: COMPONENTS101, 2021

O ESP32 WROOM-32D é um módulo *SoC* (*System on Chip*) desenvolvido pela *Espressif Systems*, amplamente utilizado em projetos embarcados devido à sua alta integração e conectividade *Wi-Fi* e *Bluetooth*. Baseado no processador *Xtensa LX6 dual-core*, com *clock* de até 240 MHz, o ESP32 possui memória RAM de 520 KB e uma vasta gama de interfaces, incluindo *GPIOs, ADCs, DACs, SPI, I2C, UART* e *PWM*. Seu conjunto de recursos o torna ideal para aplicações em Internet das Coisas (*IoT*), automação residencial, controle remoto e monitoramento em tempo real (ESPRESSIF, 2020). A versão WROOM-32D apresenta antena

integrada com desempenho otimizado, além de certificações internacionais, o que amplia sua aplicação em soluções comerciais e industriais.

No sistema proposto, o ESP32 WROOM-32D, ilustrado na Figura 4, desempenha o papel de nó supervisório (*Node_B*), responsável por receber, via rádio frequência, os dados enviados pelo *Node_A* (Arduino) utilizando o protocolo RF24. Aproveitando sua conectividade nativa *Wi-Fi*, o ESP32 hospeda um *Webserver* local, que exibe em tempo real o estado da bomba hidráulica, o modo de operação (manual/automático), os níveis da caixa e cisterna, e o valor da vazão da água. Essa função elimina a necessidade de softwares SCADA proprietários, reduzindo os custos do projeto e facilitando o acesso remoto às informações por qualquer dispositivo conectado à mesma rede local.

Figura 4: esp32 wroom-32d



Fonte: autor, 2025

O módulo de relé de 2 canais da marca *Mingwu Electronics* é um dispositivo eletromecânico amplamente utilizado em projetos de automação com microcontroladores como Arduino, ESP32 e STM32. Cada relé do módulo opera com tensão de controle de 5 V e é capaz de comutar cargas de até 10 A em 250 V AC ou 10 A em 30 V DC, o que o torna adequado para aplicações de acionamento de dispositivos como lâmpadas, motores, bombas e válvulas. O módulo conta com transistores de acionamento, diodo de proteção e optocopladores, que garantem isolamento elétrico entre o circuito de controle e o de potência, protegendo o microcontrolador de picos de tensão ou ruídos gerados durante a comutação (COMPONENTS101, 2020).

No sistema hidráulico proposto, o relé, ilustrado na Figura 5, é controlado pelo *Node_A* (Arduino Uno) para acionar a bomba de poço conforme os sensores de nível (cisterna e caixa d'água). Em modo automático, o Arduino comanda o relé com base nas leituras das boias; em

modo manual, a comutação é feita via botão físico. O uso de um módulo de relé confiável com isolação garante segurança elétrica, permitindo que o circuito de controle opere sem interferência direta da carga da bomba.

Figura 5: relé de 2 canais

Fonte: autor, 2025

TOPOLOGIA DO SISTEMA

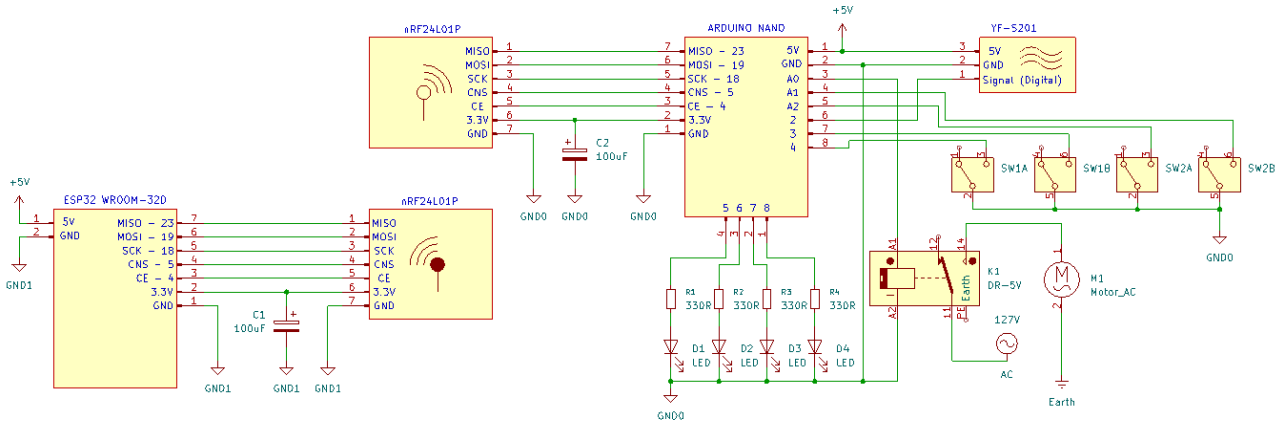
Para garantir a correta integração entre os componentes do sistema, foi necessário identificar os terminais dos microcontroladores, módulos e sensores, consultando seus respectivos *datasheets* para assegurar a compatibilidade elétrica e lógica com as plataformas Arduino e ESP32. Com base nessas especificações, foi realizada a diagramação das conexões utilizando o software KiCAD, uma suíte *open-source* amplamente adotada por engenheiros, professores e entusiastas da eletrônica. O KiCAD possibilita o desenvolvimento completo de circuitos impressos (PCBs), incluindo o diagrama esquemático, o roteamento interativo e a geração dos arquivos *Gerber*, além de ferramentas como simulação, edição de *footprints* e visualização 3D da placa. Sua aplicação neste projeto contribuiu para uma documentação técnica clara e detalhada dos dois nós (*Node_A* e *Node_B*), promovendo a replicabilidade e garantindo precisão na organização lógica do circuito (KICAD, 2023).

A estrutura de comunicação sem fio adotada foi baseada no transceptor nRF24L01+PA+LNA, que se mostrou uma solução eficiente para a troca de dados entre os nós do sistema. Trabalhos como o de Wu et al. (2017) destacam o uso desse módulo em aplicações industriais de monitoramento de temperatura com transmissão bidirecional estável. Da mesma forma, Kota et al. (2023) demonstraram sua viabilidade em sistemas de monitoramento contínuo, utilizando Arduino e nRF24L01, evidenciando que transceptores de 2.4 GHz com modulação GFSK oferecem confiabilidade e baixo consumo em aplicações críticas. Essa base

técnica fundamentou a escolha da topologia adotada no presente projeto, aliando simplicidade, robustez e economia.

O sistema funciona de forma distribuída, ilustrado na Figura 6, composto por dois nós interligados via rádio frequência utilizando módulos nRF24L01+PA+LNA. O *Node_A*, instalado junto à bomba do poço artesiano, realiza a leitura das chaves boia que indicam o nível da cisterna e da caixa d'água, além de medir a vazão instantânea da água através de um sensor YF-S201. Com base no modo de operação selecionado (manual ou automático), o Arduino aciona ou desliga a bomba por meio de um relé. A cada ciclo, o *Node_A* envia os dados encapsulados em uma "*struct*" para o *Node_B*, que é um ESP32 com Wi-Fi ativo. Este, por sua vez, recebe os dados via rádio, interpreta as informações e exibe o status do sistema em tempo real por meio de um *Webserver*, acessível por qualquer dispositivo conectado à mesma rede local. A interface web apresenta os estados dos sensores, da bomba, o modo de operação e o valor da vazão, atualizados dinamicamente a cada segundo.

Figura 6: diagrama elétrico do sistema



Fonte: AUTOR, 2025

LINGUAGEM DE PROGRAMAÇÃO DO SISTEMA (*NODE_A E NODE_B*)

Para o desenvolvimento do protótipo proposto, foi utilizada a mesma plataforma de programação para os dois nós — o *Node_A* (Arduino Nano) e o *Node_B* (ESP32 WROOM-32D) — por meio da IDE do Arduino, que permite programar dispositivos compatíveis utilizando uma linguagem baseada em C/C++ com extensões simplificadas. Essa escolha evitou a necessidade de utilizar o ambiente nativo da Espressif (ESP-IDF), uma vez que a comunidade Arduino fornece suporte ao ESP32 por meio da adição do pacote de placas oficial e da biblioteca

"Arduino.h", que emula a estrutura da API do Arduino no microcontrolador da Espressif (ARDUINO.CC, 2018). A padronização do ambiente de desenvolvimento facilitou a prototipação, integração dos módulos e depuração do sistema.

Os códigos-fonte dos dois nós não foram desenvolvidos do zero, mas a partir da adaptação de exemplos oficiais e de uso comunitário presentes nas bibliotecas disponíveis no gerenciador da IDE do Arduino. Bibliotecas fundamentais como "SPI.h" (para comunicação serial síncrona), "RF24.h" (para interface com o módulo nRF24L01), "WiFi.h" e "*WebServer*.h" (para gerenciamento de conexão e páginas web no ESP32) foram utilizadas. Essas bibliotecas são mantidas por desenvolvedores independentes e por comunidades técnicas no GitHub, onde estão documentadas e disponíveis sob licenças de código aberto, possibilitando ajustes conforme a necessidade do projeto (GITHUB, 2025). O uso de bibliotecas pré-compiladas com suporte ativo agilizou o processo de desenvolvimento e proporcionou confiabilidade na comunicação entre hardware e software.

A estrutura dos códigos seguiu o modelo padrão de programação da IDE Arduino, organizado em duas seções principais: *setup()* e *loop()*. No *Node_A*, a função *setup()* inicializa os pinos de entrada e saída, ativa interrupções para o sensor de vazão YF-S201, configura a comunicação RF e define os estados iniciais da bomba e dos *LEDs* indicadores. Já na função *loop()*, o sistema interpreta as leituras das chaves boia, calcula a vazão com base na contagem de pulsos e decide o acionamento da bomba, além de transmitir os dados por rádio a cada ciclo. No *Node_B*, também estruturado com *setup()* e *loop()*, a função *setup()* conecta o ESP32 à rede Wi-Fi, inicializa o receptor nRF24L01 e configura o *Webserver* com rotas HTTP. Na função *loop()*, o ESP32 processa a chegada dos dados RF, responde com um *ACK* e atualiza dinamicamente os valores no navegador, oferecendo ao usuário uma interface de supervisão em tempo real. Essa organização modular permitiu que cada nó desempenhasse sua função específica com clareza e independência.

CÓDIGO *NODE_A*

```
"// NODE_A (Arduino + nRF24L) Controle e medição da bomba e de nível d'agua.
#include <SPI.h>
#include "RF24.h"
RF24 radio(9, 10);
const byte addressA[6] = "NodeA";
```

```
const byte addressB[6] = "NodeB";
// Pinos
const int relayPin = A0;
const int boiaPocoPin = A1;
const int boiaCaixaPin = A2;
const int flowSensorPin = 2;
const int switchAutoManualPin = 3;
const int buttonManualPin = 4;
const int ledSend = 5;
const int ledReceive = 6;
const int ledAuto = 7;
const int ledManual = 8;
volatile uint16_t pulseCount = 0;
unsigned long lastFlowCalcTime = 0;
float flowRate = 0.0;
const float pulsesPerLiter = 7.5;
bool bombaLigada = false;
// Struct que será enviado via RF
struct StatusData {
  bool modoManual;
  bool caixaCheia;
  bool cisternaCheia;
  bool bombaLigada;
  float vazao;
};
void pulseCounter() {
  pulseCount++;
}
void setup() {
  Serial.begin(9600);
  pinMode(relayPin, OUTPUT);
  pinMode(boiaPocoPin, INPUT_PULLUP);
  pinMode(boiaCaixaPin, INPUT_PULLUP);
  pinMode(switchAutoManualPin, INPUT_PULLUP);
```

```
  pinMode(buttonManualPin, INPUT_PULLUP);
  pinMode(ledSend, OUTPUT);
  pinMode(ledReceive, OUTPUT);
  pinMode(ledAuto, OUTPUT);
  pinMode(ledManual, OUTPUT);
  pinMode(flowSensorPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(flowSensorPin), pulseCounter, RISING);
  digitalWrite(relayPin, LOW); // Bomba desligada inicialmente
  radio.begin();
  radio.setChannel(108);          // Menos interferência Wi-Fi
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);   // Mais robusto
  radio.setRetries(15, 15);
  radio.enableDynamicPayloads();
  radio.openWritingPipe(addressB);
  radio.openReadingPipe(1, addressA);
  radio.startListening();
  radio.printDetails();
  delay(3000);
}
void loop() {
  bool modoManual = (digitalRead(switchAutoManualPin) == LOW);
  // LEDs de modo
  digitalWrite(ledManual, modoManual);
  digitalWrite(ledAuto, !modoManual);
  if (modoManual) {
    if (digitalRead(buttonManualPin) == LOW) {
      bombaLigada = !bombaLigada;
      delay(300);  // debounce
    }
  } else {
    bool aguaNoPoco = (digitalRead(boiaPocoPin) == LOW);
    bool caixaCheia = (digitalRead(boiaCaixaPin) == LOW);
    bombaLigada = (aguaNoPoco && !caixaCheia);
```

```
}
digitalWrite(relayPin, bombaLigada);
// Cálculo de vazão
if (millis() - lastFlowCalcTime >= 1000) {
  noInterrupts();
  uint16_t pulses = pulseCount;
  pulseCount = 0;
  interrupts();
  flowRate = (pulses / pulsesPerLiter) * 60.0;  // L/min
  lastFlowCalcTime = millis();
  Serial.println();
  Serial.println("-------------- Inicio teste Monitor Serial --------------");
  Serial.print(F("Vazão atual: "));
  Serial.print(flowRate);
  Serial.println(F(" L/min"));
}
// Prepara os dados para envio via struct
StatusData dados;
dados.modoManual   = modoManual;
dados.caixaCheia   = (digitalRead(boiaCaixaPin) == LOW);
dados.cisternaCheia = (digitalRead(boiaPocoPin) == LOW);
dados.bombaLigada  = bombaLigada;
dados.vazao = flowRate;
radio.stopListening();
digitalWrite(ledSend, HIGH);
bool ok = radio.write(&dados, sizeof(dados));
digitalWrite(ledSend, LOW);
Serial.println(ok ? F("Struct enviada com sucesso.") : F("Erro no envio da struct."));
radio.startListening();
// Verifica resposta
unsigned long waitStart = millis();
bool respostaRecebida = false;
while (millis() - waitStart < 150) {
  if (radio.available()) {
```

```
    char reply[32];
    uint8_t len = radio.getDynamicPayloadSize();
    radio.read(&reply, len);
    Serial.print(F("Resposta: "));
    Serial.println(reply);
    Serial.println("-------------- Fim teste Monitor Serial --------------");
    Serial.println();
    digitalWrite(ledReceive, HIGH);
    respostaRecebida = true;
    break;
   }
   delayMicroseconds(50);
  }
  if (!respostaRecebida) {
   digitalWrite(ledReceive, LOW);
  }
  delay(500);
} "
```


CÓDIGO *NODE_B*

```
"//Código do NODE_B (ESP32  + nRF24L)
#include "Arduino.h"
#include <WiFi.h>
#include <WebServer.h>
#include <SPI.h>
#include "RF24.h"
// --------------------- Wi-Fi ----------------------
const char* ssid = "Edge 40 Neo";
const char* password = "poiuytrewq";
// --------------------- RF24 ----------------------
#define CE_PIN 4
#define CSN_PIN 5
RF24 radio(CE_PIN, CSN_PIN);
```

```
const byte addressA[6] = "NodeA";
const byte addressB[6] = "NodeB";
struct StatusData {
  bool modoManual;
  bool caixaCheia;
  bool cisternaCheia;
  bool bombaLigada;
  float vazao;
};
StatusData statusAtual;
// ---------------------- WebServer ----------------------
WebServer server(80);
// Página principal (HTML com JavaScript)
void handleRoot() {
  String html = R"rawliteral(
    <!DOCTYPE html>
    <html>
    <head>
      <meta charset="UTF-8">
      <title>Status Hidráulico</title>
      <style>
        body { font-family: Arial; background-color: #fofofo; padding: 20px; }
        h2 { color: #006699; }
        .info { font-size: 1.2em; margin: 10px 0; }
      </style>
      <script>
        async function atualizar() {
          const res = await fetch("/data");
          const dados = await res.json();
          document.getElementById("modo").innerText = dados.modoManual ? "Manual" :
"Automático";
          document.getElementById("caixa").innerText = dados.caixaCheia ? "Cheia" : "Vazia";
          document.getElementById("cisterna").innerText = dados.cisternaCheia ? "Cheia" :
"Vazia";
```

```
      document.getElementById("bomba").innerText    = dados.bombaLigada ? "Ligada" :
"Desligada";
      document.getElementById("vazao").innerText    = dados.vazao.toFixed(1) + " L/min";
     }
     setInterval(atualizar, 1000);
     window.onload = atualizar;
    </script>
   </head>
   <body>
    <h2>Status do Sistema Hidráulico</h2>
    <div class="info">Modo de Operação: <strong id="modo"></strong></div>
    <div class="info">Caixa d'Água: <strong id="caixa"></strong></div>
    <div class="info">Cisterna: <strong id="cisterna"></strong></div>
    <div class="info">Bomba: <strong id="bomba"></strong></div>
    <div class="info">Vazão: <strong id="vazao"></strong></div>
   </body>
   </html>
  )rawliteral";
  server.send(200, "text/html", html);
}
// Endpoint JSON para atualizar a tela
void handleData() {
  String json = "{";
  json += "\"modoManual\":" + String(statusAtual.modoManual ? "true" : "false") + ",";
  json += "\"caixaCheia\":" + String(statusAtual.caixaCheia ? "true" : "false") + ",";
  json += "\"cisternaCheia\":" + String(statusAtual.cisternaCheia ? "true" : "false") + ",";
  json += "\"bombaLigada\":" + String(statusAtual.bombaLigada ? "true" : "false") + ",";
  json += "\"vazao\":" + String(statusAtual.vazao, 1);
  json += "}";
  server.send(200, "application/json", json);
}
void setup() {
  Serial.begin(115200);
  // Conecta ao Wi-Fi
```

```
  WiFi.begin(ssid, password);
  Serial.print("Conectando ao WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado! IP: " + WiFi.localIP().toString());
  // Inicializa RF24
  radio.begin();
  radio.setChannel(108);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);
  radio.setRetries(15, 15);
  radio.enableDynamicPayloads();
  radio.openWritingPipe(addressA);
  radio.openReadingPipe(1, addressB);
  radio.startListening();
  // Webserver
  server.on("/", handleRoot);
  server.on("/data", handleData);
  server.begin();
}
void loop() {
  server.handleClient();
  if (radio.available()) {
    uint8_t len = radio.getDynamicPayloadSize();
    radio.read(&statusAtual, len);
    // Envia ACK
    radio.stopListening();
    const char reply[] = "ACK NODE_B";
    radio.write(&reply, sizeof(reply));
    radio.startListening();
  }
  delay(100);
```

} “

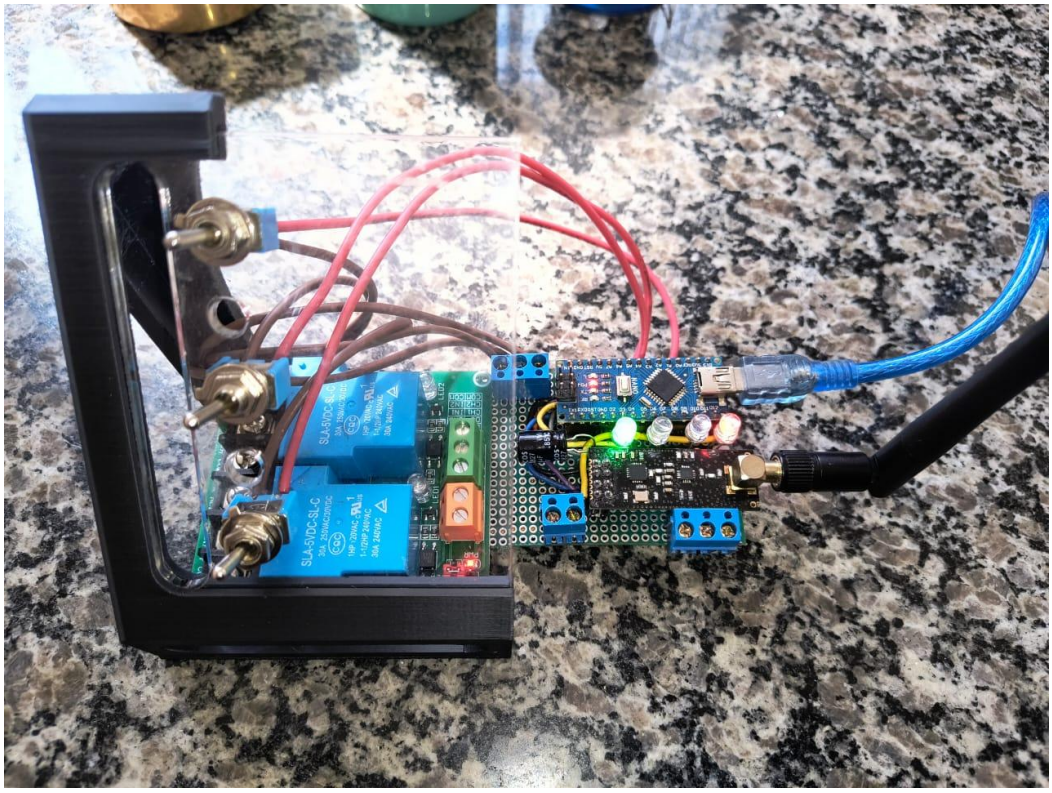
RESULTADOS E DISCUSSÃO

A estrutura de comunicação adotada neste projeto, baseada em transmissão de dados sem fio entre dois nós, reflete os benefícios apontados por Silva e Ferreira (2023), que destacam a importância da comunicação remota para o acesso a informações em tempo real, facilitando o controle, a manutenção preditiva e a eficiência operacional.

Durante os testes realizados, a comunicação sem fio entre os nós demonstrou boa estabilidade ao operar em um canal fora da faixa tradicionalmente ocupada por redes Wi-Fi, especificamente o canal 108, o que contribuiu para a redução de interferências eletromagnéticas. Além disso, a taxa de transmissão foi configurada para 250 kbps, valor recomendado para aumentar a robustez do sinal em cenários com obstáculos físicos ou ruídos ambientais. Essa abordagem técnica, aliada ao uso de módulos nRF24L01+PA+LNA, assegurou a integridade dos dados trocados no sistema hidráulico mesmo em áreas com barreiras estruturais e distâncias superiores a 30 metros entre os módulos. Os resultados indicam que, com a devida configuração de canal e taxa de dados, o sistema é capaz de manter uma comunicação confiável em ambientes residenciais e rurais com densidade moderada de dispositivos eletrônicos.

Após a finalização das ligações elétricas, e a compilação e carregamento dos códigos como ilustrado na Figura 9 e na Figura 10, verificou-se no Monitor Serial do Arduino NANO que os nós de comunicação conseguiram estabelecer, de forma satisfatória a transmissão da estrutura de dados do *NODE_A* para o *NODE_B*.

Figura 09: prototipo em funcionamento do *node_a*



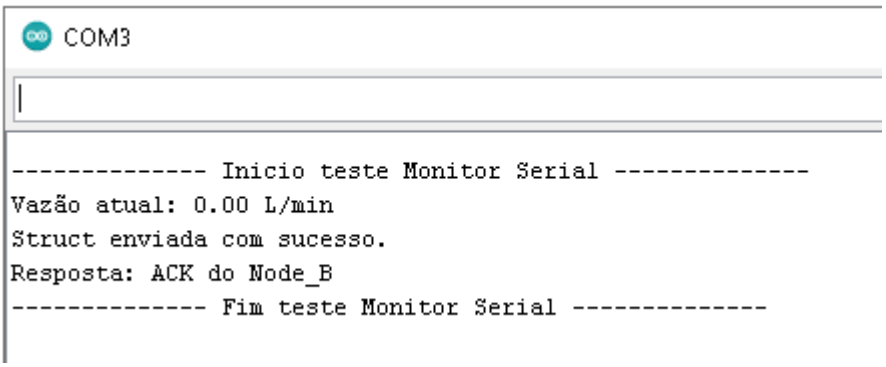
Fonte: AUTOR, 2025

Figura 10: prototipo em funcionamento do *node_b*

Fonte: AUTOR, 2025

A confirmação de recebimento retransmitida do *NODE_B* para o *NODE_A*, como ilustrado na Figura 11

Figura 11: monitor serial do arduino nano (*node_a)*



Fonte: autor, 2025

A escolha pelo uso da comunicação via nRF24L01+PA+LNA com estrutura ponto-a-ponto justifica-se pela eficiência já comprovada em estudos prévios, especialmente em aplicações onde o baixo consumo energético e a confiabilidade são essenciais. Wu et al. (2017) demonstraram que a tecnologia pode alcançar até 300 metros em linha reta sem obstáculos, mantendo a integridade dos dados em aplicações de monitoramento ambiental.

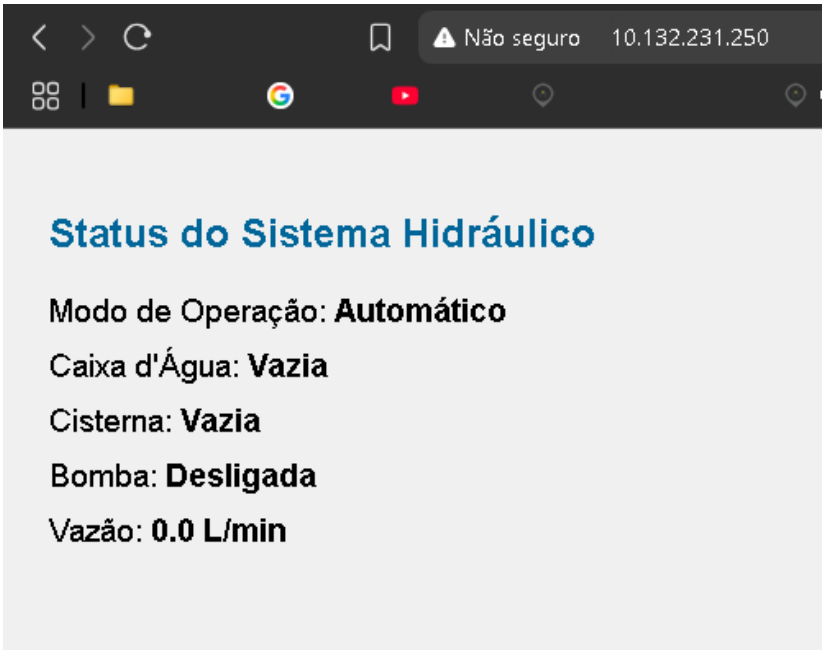
Já em contextos com interferências e barreiras físicas moderadas, como paredes e estruturas metálicas, Dialani (2023) reportou distâncias operacionais efetivas de até 100 metros, sem perda significativa de pacotes.

Embora este projeto não tenha se concentrado em testar o alcance máximo do módulo, as evidências da literatura demonstram que o nRF24L01+PA+LNA é capaz de atender com segurança às necessidades de comunicação entre os nós do sistema em ambientes rurais ou residenciais, mesmo quando a topologia envolve obstáculos.

De mesmo modo, foi testado a conexão Wi-Fi e o *Webserver* local, e como verificado anteriormente com a transmissão da estrutura de dados bem-sucedida, o ESP32 performou satisfatoriamente em carregar os dados recebidos e exibir no *Webserver* para a supervisão dos dados como ilustrado na Figura 12.

A interface web gerada pelo ESP32 permitiu acesso simultâneo por múltiplos dispositivos na rede local, com tempo de resposta satisfatório. O controle automático da bomba com base no nível dos reservatórios operou de forma eficiente, e o modo manual também foi validado.

Figura 12: *webserver* local do *node_b*



Fonte: autor, 2025

A confiabilidade da comunicação entre o *Node_A* e o *Node_B* foi garantida pelo uso de módulos nRF24L01+, estratégia alinhada ao que foi validado por Dialani (2023) em sistemas de transmissão de potência sem fio com feedback ativo e comunicação de parâmetros em tempo real.

CONSIDERAÇÕES FINAIS

O sistema proposto demonstrou sua viabilidade técnica, funcionalidade e aplicabilidade no contexto de monitoramento e controle de sistemas hidráulicos, especialmente em ambientes residenciais e rurais com infraestrutura limitada. A arquitetura baseada em dois nós — *Node_A* (Arduino Nano) e *Node_B* (ESP32) — integrou de forma eficiente sensores de nível e de vazão, acionamento automático de bomba hidráulica, comunicação sem fio por rádio frequência (nRF24L01+PA+LNA) e visualização remota dos dados via interface *web* embarcada.

Durante a fase de testes, verificou-se que a escolha por operar em um canal RF fora das faixas usualmente congestionadas (canal 108) e com taxa reduzida de 250 kbps contribuiu significativamente para a robustez da comunicação entre os nós, mesmo em cenários com obstáculos físicos e distâncias superiores a 30 metros. A estrutura ponto-a-ponto adotada mostrou-se eficaz para garantir baixa latência, alta integridade dos dados e consumo energético reduzido, características essenciais para aplicações distribuídas e de difícil acesso.

A implementação do *Webserver* local no ESP32 permitiu a visualização em tempo real do estado da bomba, do modo de operação (manual ou automático), dos níveis da cisterna e da caixa d'água, além do valor de vazão atual. A interface demonstrou bom desempenho de carregamento e atualização, podendo ser acessada simultaneamente por múltiplos dispositivos conectados à rede local.

Além de atender plenamente aos objetivos iniciais — oferecer uma solução de baixo custo, de fácil replicação e de instalação simplificada —, o sistema também evidencia a versatilidade e o potencial dos microcontroladores de código aberto e da comunicação RF em aplicações de automação distribuída. A proposta contribui ainda para práticas de uso racional da água, alinhando-se a princípios de sustentabilidade e eficiência energética.

Como trabalhos futuros, destacam-se a expansão da solução com armazenamento de dados em nuvem, permitindo histórico e análise preditiva de consumo; a implementação de um aplicativo móvel com interface gráfica amigável para visualização remota; e a integração com protocolos padronizados de Internet das Coisas, como MQTT e *Node-RED*, ampliando as possibilidades de supervisão remota e automação inteligente. Outras melhorias podem incluir a detecção de falhas no sistema hidráulico, alarmes via notificação em tempo real e até mesmo integração com assistentes virtuais ou plataformas de casa inteligente.

Portanto, o sistema desenvolvido representa um passo significativo na direção da democratização da automação hídrica acessível, com aplicações práticas que vão desde a gestão de reservatórios domiciliares até a automação de pequenos sistemas de irrigação rural, podendo também ser adaptado para contextos educacionais e experimentais em cursos técnicos e superiores na área de automação, eletrônica e *IoT*.

REFERÊNCIAS

ARDUINO, C.C. *What is Arduino?* 2018. Disponível

https://www.arduino.cc/en/Guide/Introduction. Acesso em: 13 jul. 2025.

COMPONENTS101. *5V Dual-Channel Relay Module Datasheet and Specifications*. Components101, 2020. Disponível em: https://components101.com/switches/5v-dual-channel-relay-module-pinout-features-applications-working-datasheet. Acesso em: 14 jul. 2025.

COMPONENTS101. *YF-S201 Water Flow Measurement Sensor Datasheet*. Components101, [Shenzhen, China] 2021. Disponível em: https://components101.com/sensors/yf-s201-water-flow-measurement-sensor. Acesso em: 14 jul. 2025

DIALANI, Hitesh. *Highly Efficient Dual-Side Wireless Power Transfer: Implementation of Synchronization and Wireless Communication*. *Delft University of Technology*, 2023. *Master Thesis*. Disponível em: http://repository.tudelft.nl. Acesso em: 16 jul. 2025.

ESPRESSIF SYSTEMS. *ESP32-WROOM-32D Datasheet*. Version 1.9. Espressif Systems, 2020. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Acesso em: 14 jul. 2025.

GITHUB. *GitHub – Where the world builds software*. GitHub, 2025. Disponível em: https://github.com/. Acesso em: 14 jul. 2025.

KICAD. *KiCad EDA – A Cross Platform and Open Source Electronics Design Automation Suite*. KiCad Developers Team, 2023. Disponível em: https://www.kicad.org/. Acesso em: 14 jul. 2025.

MAHMOOD, M. F.; MOHAMMED, S. L.; GHARGHAN, S. K.; AL-NAJI, A.; CHAHL, J. *Hybrid Coils-Based Wireless Power Transfer for Intelligent Sensors*. *Sensors*, v. 20, n. 9, p. 2549, 2020. DOI: 10.3390/s20092549.

NORDIC SEMICONDUCTOR. *nRF24L01+ Single Chip 2.4GHz Transceiver: Product Specification v1.0*. Oslo, Noruega, 2013. Disponível em: https://www.nordicsemi.com. Acesso em: 14 jul. 2025.

SILVA, Adalberto Oliveira da; FERREIRA, Alex Franco. O papel da comunicação remota nas organizações industriais. *Revista Ibero-Americana de Humanidades, Ciências e Educação*, São Paulo, v. 9, n. 02, p. 732–746, fev. 2023. Disponível em: https://doi.org/10.51891/rease.v9i2.8578. Acesso em: 14 jul. 2025.

SOUZA, L. C.; LIMA, E. S.; NETO, E. R.; SODRÉ JUNIOR, A. C. *Implementation of PoF-powered IoT Sensing Systems for Industry 4.0*. Journal of Microwaves, Optoelectronics and Electromagnetic Applications, v. 22, n. 1, p. 162–171, mar. 2023. DOI: https://doi.org/10.1590/2179-10742023v22i1270780

WU, Nan; LIU, Chengguo; ZHANG, Bin; ZHOU, Shuai; ZHANG, Jingwei; WU, Zhi P. Wireless bi-directional data link for smart temperature recording. *Transactions of the Institute of Measurement and Control*, v. 39, n. 4, p. 420–428, 2017. DOI: 10.1177/0142331216675402.