

# Implementação WSN

Detalhamento Técnico e Montagem

## MONTAGEM FÍSICA

- Hardware: Gateway
- Hardware: Nó Sensor
- Particularidades de Montagem

## TECNOLOGIAS

- Rádio Frequência (nRF24)
- Funcionalidades ESP32-S3
- Node-RED e MQTT

## LÓGICA DO PROJETO

- Ciclo de Vida do Sensor
- Processamento no Gateway

## CONSIDERAÇÕES FINAIS

- Limitações da Arquitetura

# Hardware: Nó Gateway

O Gateway foi montado em uma placa perfurada compacta para atuar como estação base fixa. Ele é alimentado via USB (5V) e permanece sempre ativo.

### Lista de Componentes

COMPONENTE	QTD	DETALHES
ESP32-S3 DevKit	1	Microcontrolador principal com Wi-Fi e BLE.
nRF24L01+PA+LNA	1	Módulo RF de longo alcance com antena externa SMA.

COMPONENTE	QTD	DETALHES
Placa Perfurada	1	Dimensões 5 cm x 7 cm (Ilhada).
Capacitor Eletrolítico	1	100µF soldado diretamente no VCC/GND do rádio.

## Hardware: Nó Sensor (Remoto)

O Nó Sensor é um dispositivo autônomo, desenhado para eficiência energética e montado em uma placa maior para acomodar a bateria e sensores.

### Lista de Componentes

COMPONENTE	QTD	FUNÇÃO
ESP32-S3 DevKit	1	Processamento e controle de Deep Sleep.
nRF24L01+PA+LNA	1	Transmissão de dados RF.
Baterias 18650	2	Ligadas em série (aprox. 7.4V a 8.4V).
Regulador LM2596	1	Step-Down DC-DC para converter 7.4V em 5V estáveis.
RTC DS3231	1	Relógio de Tempo Real de alta precisão (I2C).
DHT11	1	Sensor de Temperatura e Umidade.
LDR (Modificado)	1	Sensor de Luminosidade (Hackedo para saída Analógica).
Botão Push + Resistor	1	Botão de interrupção com resistor pull-up de 10kΩ.
Placa Perfurada	1	Dimensões 9 cm x 15 cm.

## Particularidades da Implementação

**CRÍTICO****Estabilidade do Rádio nRF24**

Durante os testes, observou-se falha constante no envio de pacotes ("Sem ACK"). O módulo nRF24L01 com amplificador de potência (PA+LNA) gera picos de corrente de até 115mA durante a transmissão.

- **Problema:** A indutância dos fios e a resposta do regulador interno do ESP32 causavam quedas de tensão momentâneas (voltage drops), resetando o rádio.
- **Solução:** Soldagem de um **capacitor eletrolítico de 100µF** diretamente nos pinos VCC e GND do módulo nRF24. Isso atua como um "tanque" de energia local para suprir os picos de transmissão.

**HARDWARE HACK****Sensor de Luminosidade**

O módulo LDR comercial adquirido possuía apenas saída digital (D0) baseada em comparador LM393, funcionando apenas como interruptor (claro/escuro).

- **Solução:** Foi realizado um "bypass" no circuito do módulo. Soldou-se um fio diretamente no divisor de tensão entre o fotoresistor e o resistor de referência, permitindo a leitura analógica (0-4095) pelo ADC do ESP32.

**PROTOCOLO****Configurações de RF**

Para garantir o alcance e confiabilidade na comunicação:

- **Data Rate:** 1 MBPS (Melhor sensibilidade que 2MBPS).
- **Potência:** RF24\_PA\_MAX (Máxima potência devido à alimentação externa).
- **Retries:** Configurado para (15, 15) - (Delay de 4000µs entre tentativas, 15 tentativas máximas) para lidar com interferências.

## Tecnologias: RF e ShockBurst

A comunicação utiliza o protocolo proprietário da Nordic Semiconductor, o **Enhanced ShockBurst™**.

- **ACK Automático:** O hardware do rádio lida automaticamente com a verificação de integridade (CRC) e o envio de confirmação (ACK). O microcontrolador não precisa gerenciar isso via software.
- **ACK Payload (Piggybacking):** Funcionalidade avançada usada no projeto. O Gateway insere dados (configurações de intervalo) no pacote de confirmação. Assim, o Sensor recebe comandos *imediatamente* após enviar dados, sem precisar entrar em modo de escuta (RX) separado, economizando bateria.

# Funcionalidades do ESP32-S3

---

O projeto explora recursos específicos do SoC da Espressif:

- **Deep Sleep & RTC Domain:** O processador principal é desligado. Apenas o coprocessador ULP (Ultra Low Power) e a memória RTC permanecem ativos.
- **RTC GPIO Isolation:** Uso da função `rtc_gpio_pullup_en` e `esp_sleep_pd_config` para garantir que os pinos de interrupção (Botão e RTC SQW) não flutuem durante o sono, o que causaria falsos despertares e alto consumo.
- **Wakeup Source (EXT1):** Configuração de máscara de bits para permitir que múltiplos pinos (Botão ou RTC) acordem o dispositivo.
- **RMT Driver:** Controle do LED RGB nativo (WS2812) no pino 48 para feedback visual de status no Gateway.

## Node-RED e Lógica de Nuvem

---

O backend foi implementado em Node-RED v2.0 (FlowFuse) rodando localmente.

- **Dashboard 2.0:** Uso da nova biblioteca baseada em Vue.js para interface responsiva.
- **Layout Masonry:** Para evitar problemas de alinhamento visual, os gráficos foram separados em grupos individuais, forçando um layout de duas colunas (Sensores/Controle na esquerda, Gráficos na direita).
- **Correção de Timestamp:** Como o RTC envia o horário local cru, o Node-RED aplica uma correção de fuso horário (+3h ou 10800s) antes de renderizar os gráficos, compensando a interpretação UTC padrão do Javascript.

## Lógica de Funcionamento: Nó Sensor

---

O firmware do sensor não utiliza o loop principal. Ele opera em um ciclo linear "Acordar -> Executar -> Dormir".

1. **Wake-up:** O ESP32 acorda por interrupção externa (Botão pressionado ou pino SQW do RTC indo para LOW).
2. **Leitura:** Coleta dados do DHT11, LDR e o Timestamp atual do DS3231.
3. **Transmissão:** Envia a estrutura de dados via RF.

4. **Recepção (ACK):** Verifica se o Gateway enviou algum comando no pacote de resposta. Se houver um novo intervalo de tempo, atualiza a variável na memória RTC.
5. **Agendamento:** Calcula o próximo horário de despertar e programa o alarme do chip DS3231.
6. **Hibernação:** Entra em Deep Sleep, desligando todos os periféricos não essenciais.

## Lógica de Funcionamento: Gateway

---

O Gateway opera em modo contínuo (Always On).

- **Loop Principal:** Monitora constantemente a chegada de pacotes RF e a conexão MQTT.
- **Conversão:** Ao receber dados binários do sensor, converte o Timestamp Unix em uma string de data formatada (`time.h`) e cria um JSON.
- **Publicação:** Envia o JSON para o tópico `fazenda/estufa/dados`.
- **Controle:** Ao receber um comando do Node-RED (mudança de intervalo), armazena esse valor e o coloca no buffer de transmissão do rádio (`writeAckPayload`), para que seja entregue na próxima comunicação do sensor.

## Limitações da Arquitetura

---

- **Colisões de RF:** Como a comunicação é iniciada exclusivamente pelo sensor e não há mecanismo de "Listen Before Talk" (LBT) implementado, se houver múltiplos sensores transmitindo simultaneamente, haverá perda de pacotes.
- **Segurança:** A comunicação RF e MQTT não implementa criptografia (AES/TLS), sendo vulnerável a interceptação local.
- **RTC Drift:** Embora o DS3231 seja preciso, não há sincronização automática via NTP (Network Time Protocol) implementada na versão final para economizar complexidade, exigindo ajuste manual inicial.