

# Ultra-Low Power LoRa Remote Load Control with ESP32-S3 and SX1262

## Project Overview ["EoRa-PI-Foundation" files](#)

Successfully implemented ~175 uA average current consumption; LoRa remote load control system. Achieving months of battery operation through optimized deep sleep and duty cycle reception.

## Hardware Configuration

**Development Board:** "EoRa Pi", ESP32-S3 (SX1262 LoRa 915MHz)

- ESP32-S3 with integrated SX1262 LoRa radio
- Custom wake-up circuit using dual 74HC04N Inverters
- GPIO routing: LoRa DIO1 (GPIO33) → Inverter → Inverter → GPIO16 (RTC GPIO)

### Key Components:

- SX1262 LoRa transceiver (915MHz, 14dBm)
- 74HC04N inverters for clean wake signal routing
- High-power load control (relay/contacter)
- Battery power management

## Technical Implementation

### LoRa Configuration

```
// Optimized parameters for duty cycle operation
Frequency: 915.0 MHz
Bandwidth: 125.0 kHz
Spreading Factor: 7
Coding Rate: 4/7
TX Power: 14 dBm
Preamble: 512 symbols (critical for duty cycle timing)
Sync Word: RADIOLIB_SX126X_SYNC_WORD_PRIVATE
```

### Power Management Strategy

1. **ESP32 Deep Sleep:** Ultra-low power consumption when idle
2. **LoRa Duty Cycle:** `radio.startReceiveDutyCycleAuto()` minimizes radio power
3. **Wake-on-Radio Protocol:** Two-stage packet system for reliable operation; only one packet transmission per cycle

## Wake-Up Circuit Design

The key breakthrough was routing the LoRa DIO1 interrupt through 74HC04N for a unity gain, non-inverted DIO1 signal; used pair of inverters from 74HC04 to reroute DIO1 to RTC\_NUM\_16 an external 0, wake capable pin:

SX1262 DIO1 → GPIO33 → 74HC04N (A1 input Y1 output connected to A2 input Y2 output) → RTC\_NUM\_16 → ESP32 RTC\_GPIO, external 0, Wake on Radio (WOR)

This ensures clean signal integrity and proper deep sleep wake-up functionality.

## Software Architecture

### Library Conversion Challenge

Successfully converted from SX126x-Arduino library to RadioLib while maintaining functionality:

- Preserved duty cycle reception capabilities
- Maintained parameter compatibility
- Implemented clean String-based packet reading

### Wake-On-Radio Protocol

Implemented a two-packet protocol for reliable one transmission operation:

1. **WOR (Wake-On-Radio) packet** → Wakes ESP32 → Initializes duty cycle mode
2. **Payload packet** → Received by duty cycle radio → Executes command immediately

```
// Server side - dual transmission
sendWakePacket(); // Wake the receiver
delay(500);       // Brief pause
sendCommandPacket(); // Actual command execution
```

### Deep Sleep Management

```
void goToSleep(void) {
    // Configure LoRa for duty cycle reception
```

```
radio.startReceiveDutyCycleAuto();

// Configure RTC GPIO wake-up
rtc_gpio_pulldown_en(WAKE_PIN);
esp_sleep_enable_ext0_wakeup(WAKE_PIN, RISING);

// Enter deep sleep
esp_deep_sleep_start();
```

## Performance Results

### Power Consumption:

- Deep Sleep: Measured ~175  $\mu$ A (ESP32-S3) + duty cycle radio consumption
- Active Time: <5% duty cycle during command execution
- **Estimated Battery Life:** 6+ months on typical battery

### Reliability:

- 100% packet reception success rate
- Consistent wake-up and command execution
- Sub-second response time from transmission to action

### Operational Metrics:

- Wake-up time: <2 seconds
- Command processing: Immediate
- Return to sleep: Automatic after task completion

## Key Lessons Learned

1. **GPIO Routing Critical:** RTC GPIO access essential for deep sleep wake-up
2. **Parameter Matching:** TX/RX LoRa parameters must match exactly
3. **Duty Cycle Timing:** Longer preambles (256+ symbols) crucial for reliable duty cycle reception
4. **RadioLib Integration:** String-based packet reading provides clean implementation
5. **Two-Stage Protocol:** Wake-On-Radio approach solves timing challenges elegantly

## Applications

This design is ideal for:

- Remote equipment control in field deployments
- Battery-powered IoT sensor networks
- Agricultural automation systems
- Emergency/backup communication systems
- Any application requiring months of unattended operation

## Code Availability

The complete implementation demonstrates practical solutions for:

- ESP32-S3 deep sleep optimization
- LoRa duty cycle reception
- RadioLib integration
- Hardware interrupt management
- Ultra-low power design patterns

## Conclusion

Successfully achieved the goal of creating a production-ready, ultra-low power LoRa remote control system. The combination of ESP32-S3 deep sleep, SX1262 duty cycle reception, and smart protocol design delivers months of battery operation with reliable command execution.

The project showcases the powerful capabilities of the ESP32-S3 platform for battery-powered IoT applications when properly optimized.

---

**Hardware:** [EoRa-S3-900TB](#), Dev. Board, “EoRa Pi” (ESP32-S3 + SX1262)

**Software:** Arduino IDE, RadioLib, ESP-IDF framework

**Power:** Battery-optimized for field deployment

**Range:** LoRa 915MHz with excellent rural coverage