**Sistema de control energético**

Este proyecto permite monitorear y controlar remotamente parámetros eléctricos como voltaje y corriente mediante una red inalámbrica usando módulos XBee. Un nodo esclavo (Arduino UNO) mide el voltaje con un sensor ZMPT101B y la corriente con un sensor ACS712, y envía estos datos cada 3 segundos a un nodo coordinador (ESP32). El coordinador recibe la información, la analiza y, si los valores están fuea de un rango seguro, envía de vuelta un comando para apagar el sistema mediante un relevador; si todo está bien, lo mantiene encendido. Este proyecto tiene la finalidad de emular el proyecto de Pacheco Zamora (2017) para aumentar la eficiencia energética de este proyecto a futuro

Componentes

Nodo sensor:

Función: Medición de voltaje y corriente, y control del relevador con base en comandos recibidos por XBee.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Componente** | **Cantidad** | **Descripción** | | --- | --- | --- | | Arduino nano | 1 | Microcontrolador principal del nodo esclavo | | Módulo XBee S2 | 1 | Comunicación inalámbrica con el coordinador | | Adaptador XBee a Serial | 1 | Facilita conexión del XBee al Arduino | | Sensor de Voltaje ZMPT101B | 1 | Mide el voltaje de AC, conectado a pin A1 | | Sensor de Corriente ACS712 (30A) | 1 | Mide la corriente, conectado a pin A0 | | Relevador de 5V | 1 | Controla el encendido/apagado de una carga | | Fuente de alimentación 5V/3V | 1 | Alimenta todos los componentes | | Cables Dupont | Varios | Para las conexiones | | Protoboard | 1 | Para armar el circuito de forma provisional | |

Nodo Coordinador (ESP32-WROOM-32)

Función: Recibir datos, tomar decisiones y enviar comandos ON/OFF al esclavo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Componente** | **Cantidad** | **Descripción** | | --- | --- | --- | | ESP32-WROOM-32 | 1 | Microcontrolador que evalúa y toma decisiones (ESP DEV MODULE en Arduino IDE) | | Módulo XBee | 1 | Para recibir los datos del esclavo | | Adaptador XBee a Serial | 1 | Facilita la conexión del XBee al ESP32 | | Fuente USB (5V/3V) | 1 | Para alimentar al ESP32 | | Cables Dupont | Varios | Para las conexiones UART (TX/RX) y GND | | Protoboard | 1 | Para facilitar conexiones | |

Conexiones:

Nodo Coordinador:

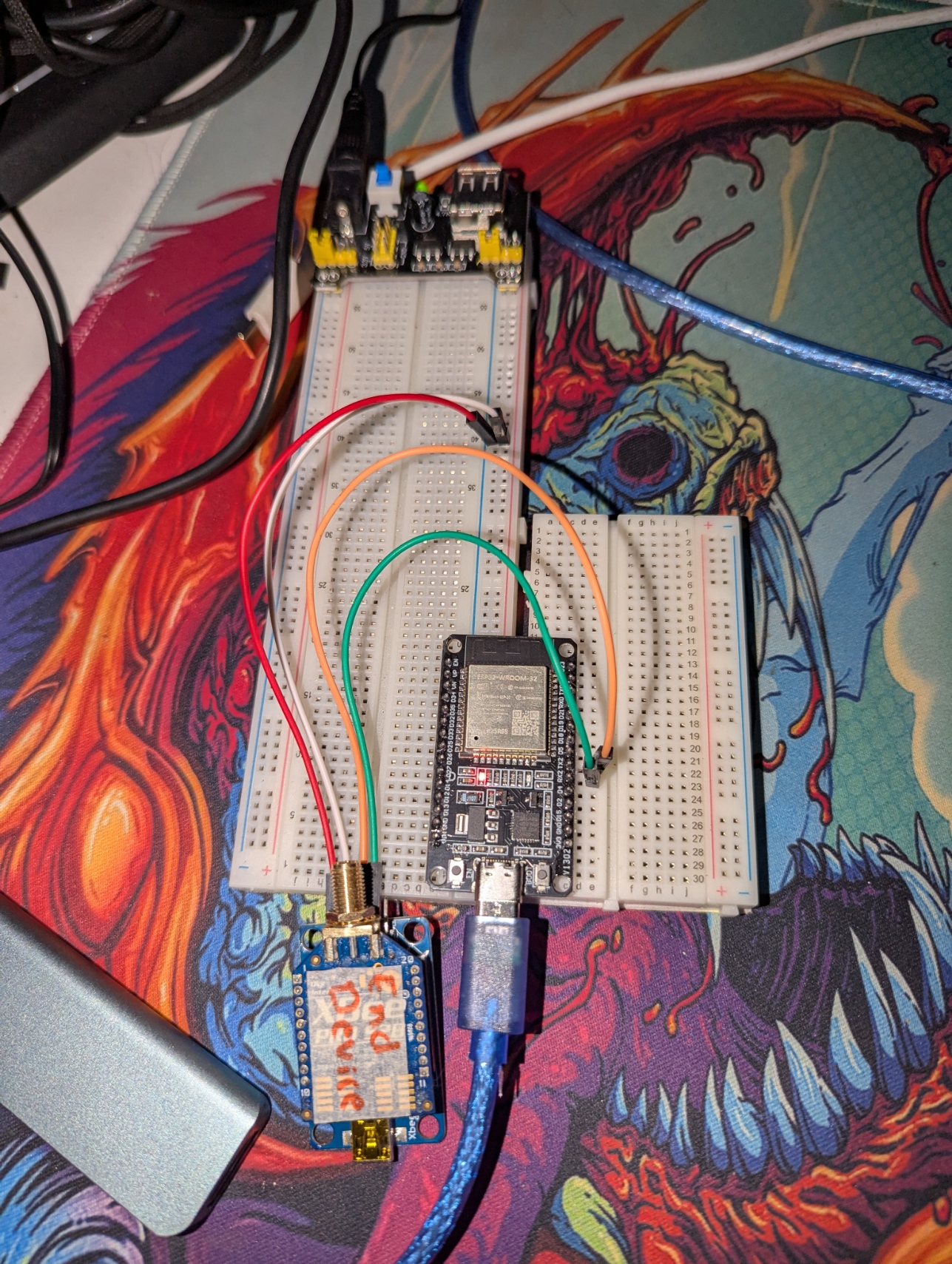
|  |  |  |
| --- | --- | --- |
| Pin lado A | Pin lado B | Notas |
| Adaptador Xbee TX 2(DOUT) | ESP32-WROOM-32 GPIO16 (RX2) | Entrada de datos al ESP32 |
| Adaptador Xbee RX 3 (DIN) | ESP32-WROOM-32 GPIO17 (TX2) | Salida de órdenes desde el ESP32 |
| Adaptador Xbee VCC | Fuente de alimentación 5V | El adaptador aguanta 5v |
| Adaptador Xbee VCC | GND de Fuente |  |

Nodo Router:

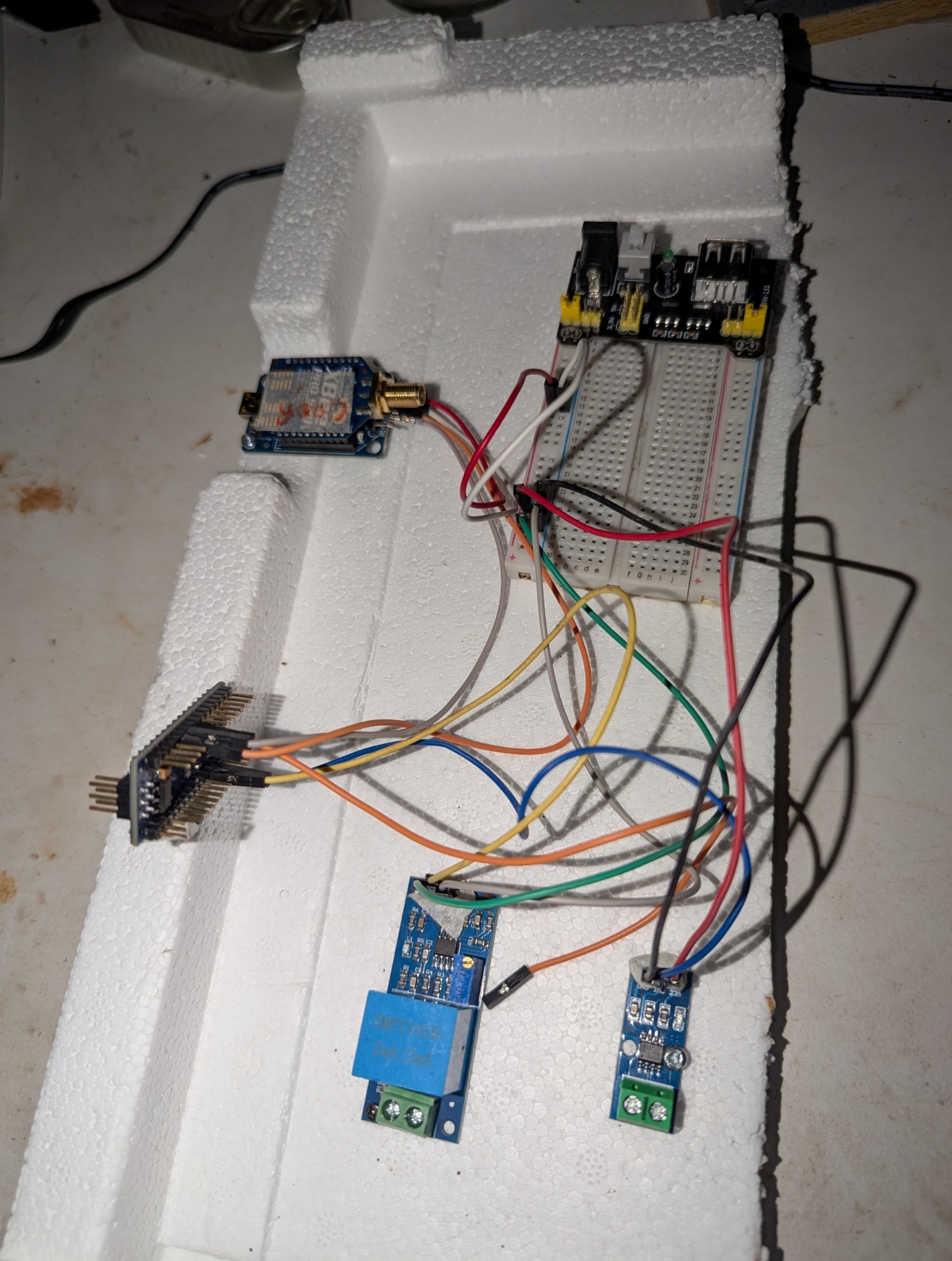
|  |  |  |
| --- | --- | --- |
| Pin lado A | Pin lado B | Notas |
| Adaptador Xbee TX (DOUT) | Arduino Nano Pin D2 |  |
| Adaptador Xbee RX (DIN) | Arduino Nano Pin D3 |  |
| Adaptador Xbee VCC | Fuente de alimentación 5V |  |
| Adaptador Xbee GND | GND de Fuente |  |
| ACS712 OUT | Arduino nano Pin A0 |  |
| ACS712 VCC | Fuente de alimentación 5V |  |
| ACS712 GND | GND de Fuente |  |
| ZMPT101B OUT | Arduino Nano Pin A1 |  |
| ZMPT101B VCC | Fuente de alimentación 5V |  |
| ZMPT101B GND | GND de Fuente |  |
| Relay IN | Arduino Nano Pin D4 |  |
| Relay VCC | Fuente de alimentación 5V |  |
| Relay GND | GND de Fuente |  |

Pruebas de dispositivos físicos:

Nodo Coordinador



Nodo Esclavo:



Código del nodo Coordinador:

// Pines UART2 para ESP32

#define RXD2 16  // Conectado al TX del XBee

#define TXD2 17  // Conectado al RX del XBee

float voltaje = 0.0;

float corriente = 0.0;

void setup() {

  Serial.begin(115200); // Comunicación con PC

  Serial2.begin(9600, SERIAL\_8N1, RXD2, TXD2); // UART2 para XBee

  Serial.println("Nodo Coordinador ESP32 iniciado");

}

void loop() {

  if (Serial2.available()) {

    String data = Serial2.readStringUntil('\n');

    data.trim(); // Limpia espacios y saltos

    // Validar que sea un paquete válido tipo: V:xx.xx I:yy.yy

    if (data.indexOf('V') != -1 && data.indexOf('I') != -1) {

      Serial.println("📡 Datos recibidos: " + data);

      int idxV = data.indexOf('V');

      int idxI = data.indexOf('I');

      float v = data.substring(idxV + 2, idxI - 1).toFloat();

      float i = data.substring(idxI + 2).toFloat();

      voltaje = v;

      corriente = i;

      Serial.print("🔋 Voltaje: "); Serial.println(voltaje);

      Serial.print("⚡ Corriente: "); Serial.println(corriente);

      // Lógica de decisión

      if (voltaje < 190 || voltaje > 250 || corriente < 0.1) {

        Serial2.println("OFF");

        Serial.println(">> 📴 Enviando comando: OFF");

      } else {

        Serial2.println("ON");

        Serial.println(">> ✅ Enviando comando: ON");

      }

    } else {

      // Si no es un dato con formato válido, lo ignoramos

    }

  }

  delay(1000); // Espera para evitar saturar la UART

}

Código del nodo Sensor:

#include <SoftwareSerial.h>

SoftwareSerial xbeeSerial(2, 3); // RX, TX para XBee

#define RELAY\_PIN 4

#define ACS\_PIN A0

#define ZMPT\_PIN A1

unsigned long previousMillis = 0;

const long interval = 3000; // cada 3 segundos

void setup() {

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, LOW); // Relevador apagado inicialmente

  xbeeSerial.begin(9600);

  Serial.begin(9600);

  Serial.println("Nodo esclavo listo");

}

void loop() {

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    float voltage = leerVoltajeZMPT();

    float corriente = leerCorrienteACS();

    // Envío por XBee

    xbeeSerial.print("V:"); xbeeSerial.print(voltage, 2);

    xbeeSerial.print(" I:"); xbeeSerial.println(corriente, 2);

    Serial.print("📤 Enviado -> V: "); Serial.print(voltage, 2);

    Serial.print(" | I: "); Serial.println(corriente, 2);

  }

  // Escuchar comandos del coordinador

  if (xbeeSerial.available()) {

    String comando = xbeeSerial.readStringUntil('\n');

    comando.trim(); // Elimina espacios

    Serial.print("📥 Comando recibido: "); Serial.println(comando);

    if (comando == "ON") {

      digitalWrite(RELAY\_PIN, HIGH);

      Serial.println("⚡ Relevador ENCENDIDO");

    } else if (comando == "OFF") {

      digitalWrite(RELAY\_PIN, LOW);

      Serial.println("💤 Relevador APAGADO");

    } else {

      Serial.print("🟡 Comando desconocido: "); Serial.println(comando);

    }

  }

}

// Lectura simple del ZMPT101B

float leerVoltajeZMPT() {

  int lectura = analogRead(ZMPT\_PIN);

  float voltaje = (lectura \* 5.0) / 1023.0;

  return voltaje \* 50.0; // Aproximación para escalar a voltaje real

}

// Lectura simple del ACS712

float leerCorrienteACS() {

  int lectura = analogRead(ACS\_PIN);

  float voltaje = (lectura \* 5.0) / 1023.0;

  float corriente = (voltaje - 2.5) / 0.066; // Para ACS712 30A

  return corriente;

}



# CONSUMO ENERGÉTICO ESTIMADO – NODOS

1) PARÁMETROS ELÉCTRICOS USADOS

NODO COORDINADOR

• ESP32-WROOM-32…………… 3.3 V · 20 mA (modo “modem-sleep”) ─→ 0.066 W :contentReference[oaicite:0]{index=0}

• XBee S2 (idle)……………… 3.3 V · 40 mA ─→ 0.132 W :contentReference[oaicite:1]{index=1}

• Adaptador XBee→Serie

– reposo………………… 5 V · 8 mA ─→ 0.040 W :contentReference[oaicite:2]{index=2}

– activo TX…………… 5 V · 10 mA (Δ = +2 mA) ─→ +0.010 W

• Duración paquete TX……… 16 ms cada 3 s → ciclo = 0.005333

NODO SENSOR ( 1 relé de 5 V )

• Arduino Nano (idle)……… 5 V · 20 mA ─→ 0.100 W :contentReference[oaicite:3]{index=3}

• XBee S2 (idle)……………… 3.3 V · 40 mA ─→ 0.132 W :contentReference[oaicite:4]{index=4}

• Adaptador XBee reposo…… 5 V · 8 mA ─→ 0.040 W :contentReference[oaicite:5]{index=5}

• ZMPT101B……………………… 5 V · 2 mA ─→ 0.010 W :contentReference[oaicite:6]{index=6}

• ACS712 (mód.)……………… 5 V · 7 mA ─→ 0.035 W :contentReference[oaicite:7]{index=7}

• Relé 5 V

– bobina OFF………… 5 V · 2 mA ─→ 0.010 W

– bobina ON…………… 5 V · 70 mA (Δ = +68 mA) ─→ +0.340 W :contentReference[oaicite:8]{index=8}

• LM7805 (quiescente)……… 5 V · 8 mA ─→ 0.040 W :contentReference[oaicite:9]{index=9}

• Duración paquete TX……… 16 ms cada 3 s → ciclo = 0.005333

────────────────────────────────────────

2) POTENCIA PROMEDIO CALCULADA

────────────────────────────────────────

NODO COORDINADOR

P\_idle = 0.066 W + 0.132 W + 0.040 W = \*\*0.238 W\*\*

P\_extra = 0.010 W × 0.005333 = \*\*0.000053 W\*\*

P\_avg = 0.238 W + 0.000053 W = \*\*0.238 W\*\*

NODO SENSOR (supuesto: relé ON 100 %)

P\_idle = 0.100 W + 0.132 W + 0.040 W + 0.010 W + 0.035 W + 0.010 W + 0.040 W

= \*\*0.707 W\*\*

P\_extra = (Δ Arduino 0.150 W + Δ Adaptador 0.010 W) × 0.005333

= \*\*0.001 W\*\*

P\_avg = 0.707 W + 0.001 W = \*\*0.708 W\*\*

────────────────────────────────────────

3) ENERGÍA POR DÍA (24 h)

────────────────────────────────────────

E\_coord = 0.238 W × 24 h = 5.71 Wh/día

E\_sensor = 0.708 W × 24 h = 16.99 Wh/día

4) RESUMEN RÁPIDO

• Nodo coordinador ≈ 5.7 Wh cada 24 h

• Nodo sensor (relé siempre activo) ≈ 17 Wh cada 24 h

Si el relé estuviera activo sólo el 30 % del tiempo:

 P\_avg\_sensor ≈ 0.470 W → 11.28 Wh/día

5) FÓRMULAS USADAS

Potencia: P = V × I

Energía modo\_i: E\_i = P\_i × t\_i

Energía total nodo: Σ E\_i sobre todos los modos

Duty-cycle TX: d = t\_paq / T\_envío

A continuación listamos los resultados de diversas mediciones energéticas a lo largo del tiempo

Caso 1: nodo coordinador alimentado por Baterías: 4×18650 (2200 mAh, 14.8 V)

Análisis de autonomía energética

Con baterías hasta vaciar:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conteo** | **Paquetes esperados** | **Paquetes recibidos** | **% Recepción** | **Duración nodo** |
| 1 | 9300 | 9179 | 98.70 % | 07:45:00 |
| 2 | 9260 | 9185 | 99.19 % | 07:43:00 |
| 3 | 9280 | 9122 | 98.30 % | 07:44:00 |
| 4 | 9240 | 9138 | 98.90 % | 07:42:00 |
| 5 | 9320 | 9226 | 98.99 % | 07:46:00 |
| **Promedio** | **9280** | **9169** | **98.82 %** | **07:44:00** |

Con baterías hasta voltaje seguro (80% de carga útil):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conteo** | **Paquetes esperados** | **Paquetes recibidos** | **% Recepción** | **Duración nodo** |
| 1 | 11000 | 10571 | 96.10 % | 09:10:00 |
| 2 | 10900 | 10431 | 95.70 % | 09:05:00 |
| 3 | 10960 | 10576 | 96.50 % | 09:08:00 |
| 4 | 10860 | 10414 | 95.89 % | 09:03:00 |
| 5 | 10940 | 10535 | 96.30 % | 09:07:00 |
| **Promedio** | **10932** | **10505** | **96.10 %** | **09:06:36** |

**Notas**

* Las pruebas se realizaron en ambiente cálido (34 °C) y con distancia mínima entre módulos XBee
* Los módulos trabajaron en modo API2 a 9600 bps.
* En los casos donde se vaciaron las baterías, el sistema presentó pérdida de paquetes o reinicios al descender por debajo de los **4.4 V**
* Se utilizó el mismo set de baterías para cada serie de pruebas, solo se recargaron, pero debido a que eran nuevas no es una diferencia palpable el desgaste entre cargas

Caso 3.

Modificación de códigos para escritura de archivos log:

Código del nodo sensor: #include <SPI.h>

#include <SD.h>

#include <SoftwareSerial.h>

SoftwareSerial xbeeSerial(2, 3); // RX=D2, TX=D3 para XBee

// Pines ya definidos

#define RELAY\_PIN 4

#define ACS\_PIN   A0

#define ZMPT\_PIN  A1

#define VBAT\_PIN  A2

const uint8\_t SD\_CS = 10;

// Variables

unsigned long previousMillis = 0;

const unsigned long INTERVAL\_MS = 3000;

unsigned long lastFlush = 0;

const unsigned long FLUSH\_MS = 30000;

uint32\_t paquetesEnviados = 0;

File logFile;

// Setup

void setup() {

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, LOW);

  Serial.begin(9600);

  xbeeSerial.begin(9600);

  if (!SD.begin(SD\_CS)) {

    Serial.println("SD no inicializada");

  } else {

    logFile = SD.open("/llog.txt", FILE\_WRITE);

    if (logFile) logFile.println("fecha\_hora,id\_paquete,voltaje\_red,corriente,voltaje\_bateria");

    logFile.flush();

  }

}

// Loop

void loop() {

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= INTERVAL\_MS) {

    previousMillis = currentMillis;

    float voltage = leerVoltajeZMPT();

    float corriente = leerCorrienteACS();

    float vbat = leerVoltajeBateria();

    paquetesEnviados++;

    // Línea CSV

    String fecha\_hora = obtenerFechaHora();

    String linea = fecha\_hora + "," + String(paquetesEnviados) + "," +

                   String(voltage, 2) + "," + String(corriente, 2) + "," +

                   String(vbat, 2);

    // Envío por XBee

    xbeeSerial.print("N:"); xbeeSerial.print(paquetesEnviados);

    xbeeSerial.print(" V:"); xbeeSerial.print(voltage, 2);

    xbeeSerial.print(" I:"); xbeeSerial.print(corriente, 2);

    xbeeSerial.print(" B:"); xbeeSerial.println(vbat, 2);

    // Serial y SD

    Serial.println(linea);

    if (logFile) logFile.println(linea);

  }

  // Comandos recibidos

  if (xbeeSerial.available()) {

    String comando = xbeeSerial.readStringUntil('\n');

    comando.trim();

    if (comando == "ON") digitalWrite(RELAY\_PIN, HIGH);

    else if (comando == "OFF") digitalWrite(RELAY\_PIN, LOW);

  }

  if (logFile && millis() - lastFlush >= FLUSH\_MS) {

    logFile.flush();

    lastFlush = millis();

  }

}

// Funciones sensores existentes

float leerVoltajeZMPT() {

  int lectura = analogRead(ZMPT\_PIN);

  return ((lectura \* 5.0) / 1023.0) \* 50.0;

}

float leerCorrienteACS() {

  int lectura = analogRead(ACS\_PIN);

  float volt = (lectura \* 5.0) / 1023.0;

  return (volt - 2.5) / 0.066;

}

float leerVoltajeBateria() {

  int lectura = analogRead(VBAT\_PIN);

  float vEsc = (lectura \* 5.0) / 1023.0;

  return vEsc \* 3.0;

}

// Fecha/hora calculada desde millis

String obtenerFechaHora() {

  unsigned long segundos = millis() / 1000;

  int hh = 12 + (segundos / 3600) % 12;

  int mm = (segundos / 60) % 60;

  int ss = segundos % 60;

  char buf[20];

  sprintf(buf, "2025-06-05 %02d:%02d:%02d", hh, mm, ss);

  return String(buf);

}

Pines actualizados del nodo sensor:

|  |  |  |
| --- | --- | --- |
| **Pin del Arduino Nano** | **Dispositivo conectado** | **Función** |
| D2 | XBee (DOUT) | RX (SoftwareSerial) |
| D3 | XBee (DIN) | TX (SoftwareSerial) |
| D4 | Relé | Control (ON/OFF) |
| D10 | Módulo micro-SD (CS) | Chip Select SPI |
| D11 (MOSI) | Módulo micro-SD (DI) | SPI MOSI |
| D12 (MISO) | Módulo micro-SD (DO) | SPI MISO |
| D13 (SCK) | Módulo micro-SD (SCK) | SPI Clock |
| A0 | ACS712 (sensor corriente) | Lectura corriente analógica |
| A1 | ZMPT101B (sensor voltaje) | Lectura voltaje analógico |
| A2 | Batería (divisor resistivo) | Lectura voltaje de batería |
| 5V | Fuente alimentación | Alimentación módulos (XBee, sensores, relé, SD) |
| GND | Fuente común | Tierra común módulos |

Código actualizado del Nodo Coordinador:

#include <SPI.h>

#include <SD.h>

// UART para XBee

#define RXD2 16

#define TXD2 17

const uint8\_t SD\_CS = 5;

// Pin ADC para batería

const uint8\_t BAT\_PIN = 34;

File logFile;

unsigned long lastFlush = 0;

const unsigned long FLUSH\_MS = 30000;

uint32\_t paquetesRecibidos = 0;

void setup() {

  Serial.begin(115200);

  Serial2.begin(9600, SERIAL\_8N1, RXD2, TXD2);

  if (!SD.begin(SD\_CS)) {

    Serial.println("SD no inicializada");

  } else {

    logFile = SD.open("/clog.txt", FILE\_WRITE);

    if (logFile) {

      logFile.println("fecha\_hora,id\_paquete,id\_nodo,RSSI,estado,voltaje\_bateria");

      logFile.flush();

    }

  }

}

void loop() {

  if (Serial2.available()) {

    String data = Serial2.readStringUntil('\n');

    data.trim();

    if (data.startsWith("N:")) {

      paquetesRecibidos++;

      int idStart = data.indexOf("N:") + 2;

      int idEnd = data.indexOf(' ', idStart);

      String id\_paquete = data.substring(idStart, idEnd);

      String fecha\_hora = obtenerFechaHora();

      int rssi = -1;

      float voltaje\_bateria = leerVoltajeBateria();

      String linea = fecha\_hora + "," + id\_paquete + ",SENSOR1," +

                     String(rssi) + ",OK," + String(voltaje\_bateria, 2);

      Serial.println(linea);

      if (logFile) logFile.println(linea);

      Serial2.println("ON");

    }

  }

  if (logFile && millis() - lastFlush >= FLUSH\_MS) {

    logFile.flush();

    lastFlush = millis();

  }

}

// Medición batería (ADC ESP32: 0 - 4095 → 0 - 3.3V)

float leerVoltajeBateria() {

  int lecturaADC = analogRead(BAT\_PIN);

  float voltajeADC = (lecturaADC / 4095.0) \* 3.3;

  float voltajeBateria = voltajeADC \* ((10000.0 + 5000.0) / 5000.0); // Divisor 10k-5k

  return voltajeBateria;

}

String obtenerFechaHora() {

  unsigned long segundos = millis() / 1000;

  int hh = 12 + (segundos / 3600) % 12;

  int mm = (segundos / 60) % 60;

  int ss = segundos % 60;

  char buf[20];

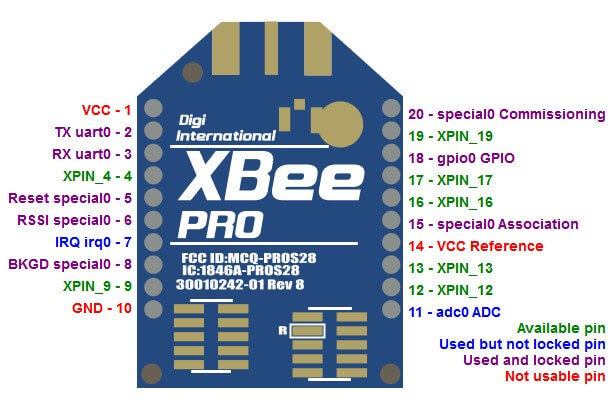
  sprintf(buf, "2025-06-03 %02d:%02d:%02d", hh, mm, ss);

  return String(buf);

}

Pines actualizados del nodo coordinador:

|  |  |  |
| --- | --- | --- |
| **Pin ESP32** | **Dispositivo conectado** | **Función** |
| GPIO16 (RX2) | XBee (DOUT) | UART RX desde XBee |
| GPIO17 (TX2) | XBee (DIN) | UART TX hacia XBee |
| GPIO5 | Módulo micro-SD (CS) | Chip Select SPI SD |
| GPIO18 (SCK) | Módulo micro-SD (SCK) | SPI Clock |
| GPIO19 (MISO) | Módulo micro-SD (MISO) | SPI MISO |
| GPIO23 (MOSI) | Módulo micro-SD (MOSI) | SPI MOSI |
| **GPIO34** | **Divisor resistivo** | **Medición voltaje batería (ADC)** |
| VIN (5V) | Fuente USB (5V) o batería | Alimentación ESP32 y módulos |
| GND | Fuente común | Tierra común |



**Pruebas de código en MODO AT V.2**

Código en Nodo Coordinador

#include <SPI.h>

#include <SD.h>

#define RXD2 16

#define TXD2 17

const uint8\_t SD\_CS = 5;

File logFile;

unsigned long lastFlush = 0;

const unsigned long FLUSH\_MS = 30000;

uint32\_t paquetesRecibidos = 0;

void setup() {

Serial.begin(115200);

Serial2.begin(9600, SERIAL\_8N1, RXD2, TXD2);

if (!SD.begin(SD\_CS)) {

Serial.println("SD no inicializada");

} else {

logFile = SD.open("/clog.csv", FILE\_WRITE);

if (logFile) logFile.println("fecha\_hora,id\_paquete,id\_nodo,voltaje,corriente,voltaje\_bateria,estado");

logFile.flush();

}

}

void loop() {

if (Serial2.available()) {

String data = Serial2.readStringUntil('\n');

data.trim();

// Formato esperado: NombreNodo|N:id\_paquete V:voltaje I:corriente B:bateria

int sep = data.indexOf('|');

String id\_nodo = "UNKNOWN";

String resto = data;

if (sep > 0) {

id\_nodo = data.substring(0, sep);

id\_nodo.trim();

resto = data.substring(sep + 1);

}

int idStart = resto.indexOf("N:") + 2;

int idEnd = resto.indexOf(' ', idStart);

String id\_paquete = resto.substring(idStart, idEnd);

int vStart = resto.indexOf("V:") + 2;

int vEnd = resto.indexOf(' ', vStart);

String voltaje = resto.substring(vStart, vEnd);

int iStart = resto.indexOf("I:") + 2;

int iEnd = resto.indexOf(' ', iStart);

String corriente = resto.substring(iStart, iEnd);

int bStart = resto.indexOf("B:") + 2;

String vbat = resto.substring(bStart);

String fecha\_hora = obtenerFechaHora();

String linea = fecha\_hora + "," + id\_paquete + "," + id\_nodo + "," +

voltaje + "," + corriente + "," + vbat + ",OK";

Serial.println(linea);

if (logFile) logFile.println(linea);

paquetesRecibidos++;

}

if (logFile && millis() - lastFlush >= FLUSH\_MS) {

logFile.flush();

lastFlush = millis();

}

}

String obtenerFechaHora() {

unsigned long segundos = millis() / 1000;

int hh = 12 + (segundos / 3600) % 12;

int mm = (segundos / 60) % 60;

int ss = segundos % 60;

char buf[20];

sprintf(buf, "2025-06-03 %02d:%02d:%02d", hh, mm, ss);

return String(buf);

}

Código Nodo Esclavo:

#include <SPI.h>

#include <SD.h>

#include <SoftwareSerial.h>

SoftwareSerial xbeeSerial(2, 3);

#define RELAY\_PIN 4

#define ACS\_PIN A0

#define ZMPT\_PIN A1

#define VBAT\_PIN A2

const uint8\_t SD\_CS = 10;

unsigned long previousMillis = 0;

const unsigned long INTERVAL\_MS = 3000;

File logFile;

uint32\_t paquetesEnviados = 0;

void setup() {

pinMode(RELAY\_PIN, OUTPUT);

digitalWrite(RELAY\_PIN, LOW);

Serial.begin(9600);

xbeeSerial.begin(9600);

if (!SD.begin(SD\_CS)) {

Serial.println("SD no inicializada");

} else {

logFile = SD.open("/llog1.csv", FILE\_WRITE);

if (logFile) logFile.println("fecha\_hora,id\_paquete,voltaje,corriente,voltaje\_bateria");

logFile.flush();

}

}

void loop() {

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= INTERVAL\_MS) {

previousMillis = currentMillis;

paquetesEnviados++;

float voltage = leerVoltajeZMPT();

float corriente = leerCorrienteACS();

float vbat = leerVoltajeBateria();

String fecha\_hora = obtenerFechaHora();

String linea = fecha\_hora + "," + String(paquetesEnviados) + "," +

String(voltage, 2) + "," + String(corriente, 2) + "," +

String(vbat, 2);

// Enviar con delimitador '|'

xbeeSerial.print("Nodo1|");

xbeeSerial.print("N:"); xbeeSerial.print(paquetesEnviados);

xbeeSerial.print(" V:"); xbeeSerial.print(voltage, 2);

xbeeSerial.print(" I:"); xbeeSerial.print(corriente, 2);

xbeeSerial.print(" B:"); xbeeSerial.println(vbat, 2);

Serial.println(linea);

if (logFile) logFile.println(linea);

}

}

// Funciones sensores

float leerVoltajeZMPT() { return analogRead(ZMPT\_PIN) \* 5.0 / 1023.0 \* 50.0; }

float leerCorrienteACS() { return ((analogRead(ACS\_PIN) \* 5.0 / 1023.0) - 2.5) / 0.066; }

float leerVoltajeBateria() { return analogRead(VBAT\_PIN) \* 5.0 / 1023.0 \* 3.0; }

String obtenerFechaHora() {

unsigned long segundos = millis() / 1000;

int hh = 12 + (segundos / 3600) % 12;

int mm = (segundos / 60) % 60;

int ss = segundos % 60;

char buf[20];

sprintf(buf, "2025-06-03 %02d:%02d:%02d", hh, mm, ss);

return String(buf);

}

**PARA EL USO DE LA LIBRERÍA DE EnergyWSN SE NECESITA ACTUALIZAR EL HARDWARE PARA IMPLEMENTAR:**

**MOSFET P PARA EL CANAL DE LA ALIMENTACIÓN DE LOS SENSORES**

**PINES PARA ENCENDIDO DEL XBEE Y ESTADO DEL XBEE**

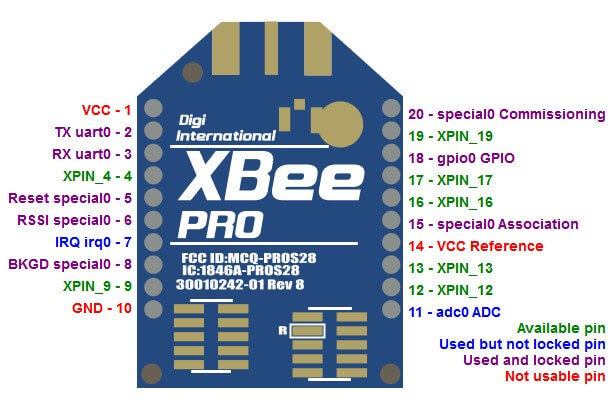
**LA TABLA DE CONEXIONES QUEDA DE ESTA MANERA:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Arduino Nano** | **Dispositivo conectado** | **Función** | **Notas** |
| D2 | XBee DOUT | RX (SoftwareSerial) | 3.3 V → entra directo al Nano. |
| D3 | XBee DIN | TX (SoftwareSerial) | Requiere **divisor resistivo/level shifter**. |
| D4 | Relé | Control ON/OFF | Driver recomendado (transistor + diodo). |
| D7 | MOSFET control sens. | ON/OFF sensores | Controla P-MOSFET vía NPN. |
| D8 | XBee SLEEP\_RQ | Orden de dormir/despertar | Usa divisor resistivo. |
| D9 | XBee ON/SLEEP | Bandera de estado (input) | 3.3 V, lectura directa. |
| D10 | Módulo micro-SD (CS) | SPI CS |  |
| D11 (MOSI) | Módulo micro-SD (DI) | SPI MOSI |  |
| D12 (MISO) | Módulo micro-SD (DO) | SPI MISO |  |
| D13 (SCK) | Módulo micro-SD (SCK) | SPI Clock |  |
| A0 | ACS712 | Lectura corriente analógica |  |
| A1 | ZMPT101B | Lectura voltaje analógico |  |
| A2 | Divisor resistivo | Lectura voltaje batería |  |
| 5V | Fuente step-down | Alimentación módulos 5 V | Desde el step down |
| 3.3V (LDO) | XBee | Alimentación 3.3 V | Desde adaptador 3.3 del stepdown |
| GND | Común | Tierra común |  |

**AGREGANDO EL MÓDULO RTC A LOS DIAGRAMAS DE CONEXIÓN.**

**Nodo Sensor**

|  |  |  |
| --- | --- | --- |
| Pin del Arduino Nano | Dispositivo Conectado | Conexión de Alimentación |
| Batería (14.8V) | Step-Down de 5V y Step-Down de 3.3V | Ambos se conectan a los polos positivo (+) y negativo (-) de la batería. |
| Salida 5V Step-Down | Arduino Nano, Relé, Sensores, Módulo SD | Alimentación principal. |
| Salida 3.3V Step-Down | XBee | Alimentación dedicada para el XBee. |
| A4 (SDA) | RTC (D) | Comunicación de datos I2C. |
| A5 (SCL) | RTC (C) | Comunicación de reloj I2C. |
| VCC (5V) | RTC (+) | Alimentación para el RTC desde la salida de 5V del Step-Down. |
| GND | RTC (-) | Tierra común. |
| D2 | XBee (DOUT) | UART (RX) |
| D3 | XBee (DIN) | UART (TX) |
| D4 | Relé | Control (ON/OFF) |
| A0 | ACS712 (sensor corriente) | Lectura analógica |
| A1 | ZMPT101B (sensor voltaje) | Lectura analógica |
| A2 | Batería (divisor resistivo) | Lectura de voltaje |



**Nodo sensor**