SISTEMA INVERSOR HÍBRIDO - MANUAL TÉCNICO COMPLETO

Parte 1: Arquitectura y Componentes

ÍNDICE GENERAL

PARTE 1: ARQUITECTURA Y COMPONENTES

1. Introducción y Objetivos
2. Arquitectura del Sistema
3. Hardware ESP32
4. Sensores y Medición

PARTE 2: BACKEND Y APIs (ver MANUAL_COMPLETO_PARTE_2.md) 5. Backend FastAPI 6. NASA POWER API 7. OpenWeather API 8. Endpoints y Comunicación

PARTE 3: CÁLCULOS Y ECUACIONES (ver MANUAL_COMPLETO_PARTE_3.md) 9. Dimensionamiento Solar 10. Dimensionamiento Eólico 11. Dimensionamiento Batería 12. Cargas Inductivas

PARTE 4: PROTECCIONES Y ML (ver MANUAL_COMPLETO_PARTE_4.md) 13. Protección Embalamiento 14. Protección Batería 15. Machine Learning 16. Estrategias Inteligentes

PARTE 5: INSTALACIÓN Y USO (ver MANUAL_COMPLETO_PARTE_5.md) 17. Configuración Completa 18. Instalación Paso a Paso 19. Troubleshooting 20. Referencias

---

1. INTRODUCCIÓN Y OBJETIVOS

1.1 Descripción del Sistema

El **Sistema Inversor Híbrido Inteligente** es una solución completa para generación, almacenamiento y gestión de energía renovable que combina:

- **Energía Solar** (paneles fotovoltaicos)
- **Energía Eólica** (turbina de viento)
- **Almacenamiento** (baterías LiFePO4)
- **Inteligencia Artificial** (predicción y optimización)

1.2 Objetivos del Sistema

1. **Autonomía Energética**: Independencia de la red eléctrica
2. **Optimización Inteligente**: IA decide qué fuente usar en cada momento
3. **Protección Avanzada**: Salvaguarda equipos y baterías
4. **Monitoreo Remoto**: Dashboard web accesible desde cualquier lugar
5. **Escalabilidad**: Soporta múltiples dispositivos ESP32

1.3 Características Principales

Técnicas:

- Microcontrolador: ESP32-WROOM-32 (Dual Core 240MHz)
- Comunicación: WiFi 802.11 b/g/n
- Sensores: 7 canales ADC de 12 bits
- Control: 5 relés (solar, eólica, red, carga, freno)
- Protecciones: Embalamiento, sobrecarga, descarga profunda

Software:

- Backend: Python FastAPI
- Frontend: React 18
- Firmware: Arduino C++ con FreeRTOS
- Base de datos: SQLite
- APIs: OpenWeather, NASA POWER

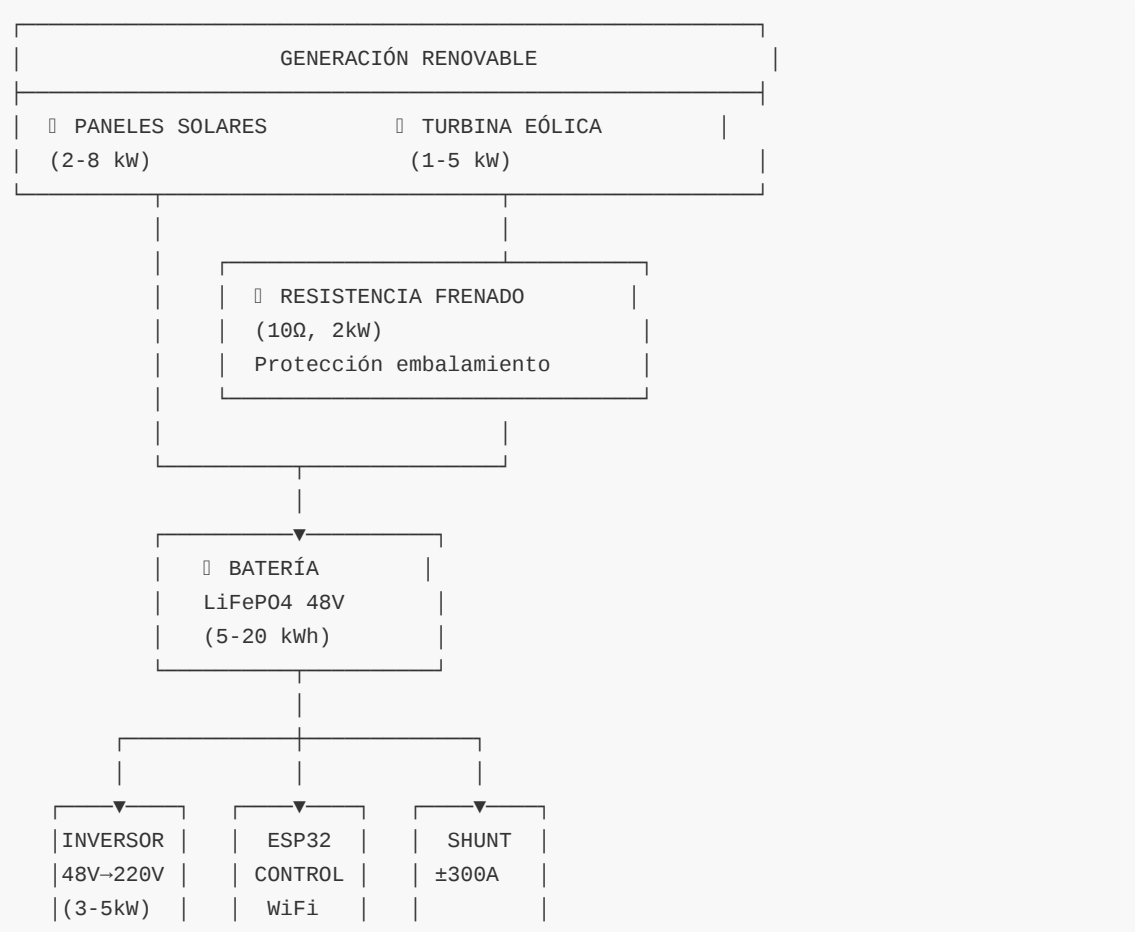
Inteligencia Artificial:

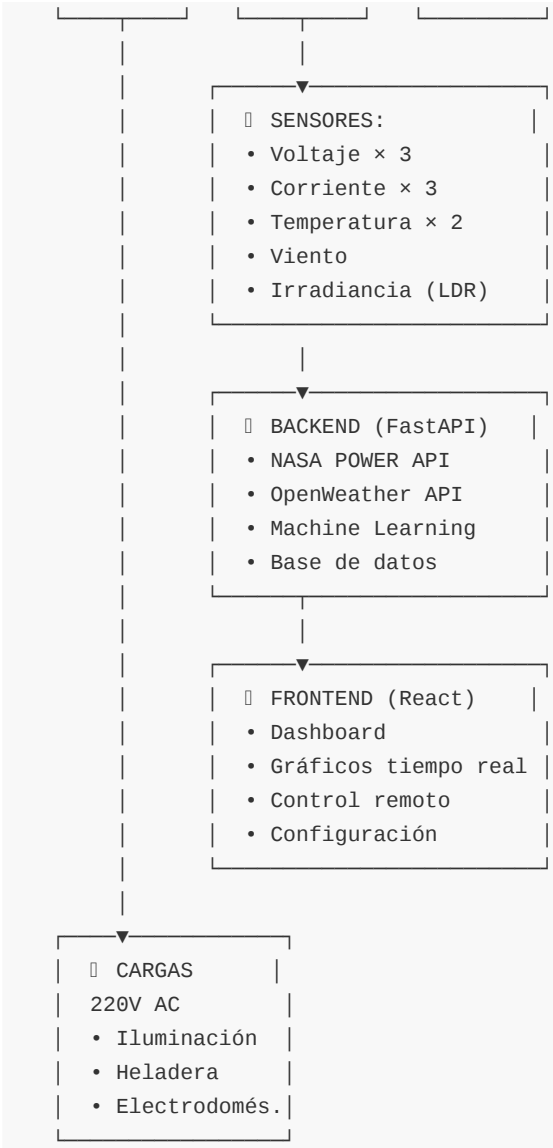
- Random Forest para predicción solar
- Gradient Boosting para predicción eólica
- Aprendizaje de patrones de consumo
- Optimización automática de fuentes

---

2. ARQUITECTURA DEL SISTEMA

2.1 Diagrama General





2.2 Flujo de Datos

Telemetría (ESP32 → Backend):

```
Cada 5 segundos:
ESP32 → HTTP POST → Backend
{
  "voltaje_bat": 52.4,
  "corriente_solar": 12.5,
  "corriente_eolica": 8.3,
  "temperatura": 25.6,
  "velocidad_viento": 6.2
}
```

Comandos (Backend → ESP32):

```
Cada 10 segundos:
ESP32 → HTTP GET → Backend
{
  "rele_solar": true,
  "rele_eolica": false,
  "modo_auto": true
}
```

Heartbeat (ESP32 → Backend):

```
Cada 30 segundos:
ESP32 → HTTP POST → Backend
{
  "uptime": 3600,
  "free_heap": 245000,
  "rssi": -45
}
```

---

3. HARDWARE ESP32

3.1 Especificaciones

Característica	Especificación
MCU	ESP32-WROOM-32
CPU	Dual Core Xtensa LX6 @ 240MHz
RAM	520 KB SRAM
Flash	4 MB
WiFi	802.11 b/g/n (2.4 GHz)
ADC	12-bit, 18 canales
GPIO	34 pines programables
Comunicación	UART, SPI, I2C, I2S
Voltaje	3.3V (entrada 5V vía USB)

3.2 Pinout Detallado

Entradas Analógicas (ADC):

```
GPIO34 (ADC1_CH6) → Voltaje Batería 1 (0-60V)
GPIO35 (ADC1_CH7) → Voltaje Batería 2 (0-60V)
GPIO32 (ADC1_CH4) → Voltaje Batería 3 (0-60V)
GPIO33 (ADC1_CH5) → Corriente Solar (Shunt 300A)
GPIO36 (ADC1_CH0) → Corriente Eólica (Shunt 300A)
GPIO39 (ADC1_CH3) → Corriente Consumo (Shunt 300A)
```

```
GPIO25 (ADC2_CH8) → Irradiancia Solar (LDR)
GPIO26 (GPIO)     → Velocidad Viento (Anemómetro)
```

Salidas Digitales (Relés):

```
GPIO16 → Relé 1: Solar ON/OFF (30A)
GPIO17 → Relé 2: Eólica ON/OFF (30A)
GPIO18 → Relé 3: Red Backup ON/OFF (30A)
GPIO19 → Relé 4: Carga ON/OFF (30A)
GPIO23 → Relé 5: Resistencia Frenado (30A)
```

Sensores Digitales:

```
GPIO4 (1-Wire) → DS18B20 Temperatura Paneles
GPIO5 (1-Wire) → DS18B20 Temperatura Batería
```

Opcional (Display):

```
GPIO21 (I2C_SDA) → Display OLED
GPIO22 (I2C_SCL) → Display OLED
```

3.3 Esquema de Conexión

Medición de Voltaje (Divisor Resistivo):

```
Batería (+) ──[100kΩ]──┬──[10kΩ]── GND
                       │
                    GPIO34
                   (ADC 0-3.3V)

Factor: 0-60V → 0-3.3V
Cálculo: V_real = ADC_value × (110kΩ/10kΩ) × (3.3V/4095)
```

Medición de Corriente (Shunt 300A):

```
Carga (+) ──[Shunt 75mV@300A]──┬── Batería (+)
                               │
                          [OpAmp x44]
                               │
                            GPIO33
                           (ADC 0-3.3V)

Shunt: 75mV @ 300A
OpAmp: Ganancia 44x → 3.3V @ 300A
Factor: 0.0732 A/bit
```

---

4. SENSORES Y MEDICIÓN

4.1 Sensores de Voltaje

Especificaciones:

- **Tipo**: Divisor resistivo 100kΩ / 10kΩ
- **Rango**: 0-60V DC
- **Precisión**: ±2% (resistencias 1%)
- **Resolución**: ~14.6 mV por bit

Calibración:

```
// Factor de calibración
#define VOLTAJE_FACTOR 0.01465  // 60V / 4095 bits

// Lectura
int adc = analogRead(PIN_VOLTAJE_BAT1);
float voltaje = adc * VOLTAJE_FACTOR;
```

Ecuación:

```
V_medido = ADC_bits × (R1 + R2) / R2 × V_ref / ADC_max
V_medido = ADC × (110k / 10k) × (3.3V / 4095)
V_medido = ADC × 0.01465
```

4.2 Sensores de Corriente

Especificaciones Shunt:

- **Modelo**: 75mV @ 300A
- **Resistencia**: 0.00025Ω (250 μΩ)
- **Potencia**: 22.5W @ 300A
- **Material**: Manganina (bajo coeficiente térmico)

Acondicionamiento de Señal:

```
Shunt: 0-75mV @ 0-300A
OpAmp: Ganancia 44x
Salida: 0-3.3V @ 0-300A

Factor: 300A / 4095 bits = 0.0732 A/bit
```

Calibración:

```
#define CORRIENTE_FACTOR 0.0732  // 300A / 4095

int adc = analogRead(PIN_CORRIENTE_SOLAR);
float corriente = adc * CORRIENTE_FACTOR;
```

4.3 Sensor de Irradiancia (LDR)

Especificaciones:

- **Modelo**: GL5528
- **Resistencia oscuridad**: 1 MΩ
- **Resistencia 10 lux**: 10-20 kΩ
- **Resistencia 100 lux**: 2-4 kΩ

Circuito:

```
3.3V ──[10kΩ]──┬──[LDR]── GND
               │
            GPIO25
```

Calibración:

```
// 0-4095 bits → 0-1200 W/m²
int adc = analogRead(PIN_IRRADIANCIA);
float irradiancia = (adc / 4095.0) * 1200;
```

4.4 Anemómetro (Velocidad Viento)

Especificaciones:

- **Tipo**: Reed switch + imán
- **Pulsos**: 1 pulso por revolución
- **Rango**: 0-50 m/s
- **Precisión**: ±0.5 m/s

Cálculo:

```
// Contar pulsos en 1 segundo
int pulsos = contadorPulsos;

// Calcular RPS (revoluciones por segundo)
float rps = pulsos / 1.0;

// Radio del anemómetro (metros)
float radio = 0.15;  // 15 cm

// Velocidad = 2π × radio × RPS
float velocidad_ms = 2 * PI * radio * rps;
```

Ecuación:

```
v = 2πr × RPS
v = 2 × 3.1416 × 0.15 × RPS
v = 0.942 × RPS (m/s)
```

FIN PARTE 1

Continúa en MANUAL_COMPLETO_PARTE_2.md