□ SISTEMA INVERSOR HÍBRIDO - MANUAL TÉCNICO COMPLETO

Parte 4: Protecciones y Machine Learning

---

13. PROTECCIÓN CONTRA EMBALAMIENTO

13.1 ¿Qué es el Embalamiento?

El embalamiento ocurre cuando la turbina eólica **gira demasiado rápido** debido a viento excesivo.

Consecuencias:

- □ Destrucción mecánica (palas, rodamientos)
- □ Sobrevoltaje (>100V puede destruir electrónica)
- □ Ruido excesivo y vibraciones peligrosas
- □ Posible incendio por fricción

13.2 Parámetros de Medición

1. Velocidad del Viento:

```
Medición: Anemómetro (pulsos/seg)

Cálculo velocidad:
v = 2π × r × RPS

Donde:
r = radio anemómetro (0.15m)
RPS = revoluciones por segundo

Umbral crítico: v > 25 m/s (90 km/h)
```

2. Voltaje de la Turbina:

```
Medición: ADC + divisor resistivo

Sistema 48V nominal:
- Normal: 48-54V
- Carga completa: 54-58V
- □ Límite: 60V
- □ PELIGRO: >65V

Relación: Voltaje □ RPM
Si viento aumenta → RPM aumentan → Voltaje sube
```

3. RPM de la Turbina:

```
Cálculo desde velocidad viento:

RPM = (v × TSR × 60) / (2π × r)

Donde:
TSR = Tip Speed Ratio (típico: 6)
v = velocidad viento (m/s)
r = radio pala (m)

Para turbina D=2.5m, TSR=6, v=25m/s:
RPM = (25 × 6 × 60) / (2π × 1.25)
RPM = 9000 / 7.85
RPM = 1,146 RPM 

Umbral seguro: 500 RPM
```

13.3 Ecuaciones de Protección

Potencia del Viento (Aumenta con v³):

```
P = 0.5 × ρ × A × v³

Ejemplo turbina 2.5m:
A = 4.91 m²

A 12 m/s (normal):
P = 0.5 × 1.225 × 4.91 × (12)³
P = 5,193 W  (dentro de límites)

A 25 m/s (embalamiento):
P = 0.5 × 1.225 × 4.91 × (25)³
P = 47,578 W  (26x más potencia!)
```

Fuerza Centrífuga (Aumenta con RPM²):

```
F = m × ω² × r

Donde:
m = masa pala (kg)
ω = velocidad angular (rad/s)
r = radio (m)

Pala 2kg, r=1.25m:

A 300 RPM (normal):
ω = 300 × 2π / 60 = 31.4 rad/s
F = 2 × (31.4)² × 1.25 = 2,463 N (251 kg)

A 600 RPM (embalamiento):
ω = 62.8 rad/s
F = 2 × (62.8)² × 1.25 = 9,852 N (1,005 kg) 
```

```
¡La fuerza se CUADRUPLICA!
```

13.4 Sistema de Frenado

Resistencia de Frenado:

```
Especificaciones:
- Resistencia: 10Ω
- Potencia: 2,000W
- Tipo: Alambre o rejilla
- Montaje: Ventilado, al exterior

Potencia disipada:
P = V² / R

A 48V: P = (48)² / 10 = 230W
A 65V: P = (65)² / 10 = 422W
A 70V: P = (70)² / 10 = 490W
```

Tiempo de Frenado:

```
Energía cinética turbina:
E = 0.5 × I × ω²

Donde:
I = momento inercia (kg·m²)
ω = velocidad angular (rad/s)

Para turbina 2kW, I≈5 kg·m²:
ω = 500 RPM = 52.4 rad/s

E = 0.5 × 5 × (52.4)²
E = 6,865 J

Tiempo frenado con resistencia 10Ω a 48V:
P_freno = 230W
t = E / P = 6,865 / 230 = 30 segundos

Tiempo frenado a 65V (embalamiento):
P_freno = 422W
t = 6,865 / 422 = 16 segundos ✓
```

13.5 Lógica de Protección (Código)

Detección:

```
bool verificarEmbalamiento() {
    // Leer sensores
    float viento = sensores.velocidad_viento;
```

```
    float voltaje = sensores.voltaje_turbina;
    float rpm = calcularRPM(viento);

    // Verificar umbrales (OR lógico)
    bool peligro = false;

    if (viento > 25.0) {          // >25 m/s
        peligro = true;
        Serial.println("⚠ Viento excesivo");
    }

    if (voltaje > 65.0) {          // >65V
        peligro = true;
        Serial.println("⚠ Sobrevoltaje");
    }

    if (rpm > 500) {               // >500 RPM
        peligro = true;
        Serial.println("⚠ RPM excesivo");
    }

    return peligro;
}
```

Activación:

```
void activarProteccion() {
    Serial.println("🔴 ACTIVANDO PROTECCIÓN");

    // 1. Desconectar turbina (relé GPIO17)
    digitalWrite(PIN_RELE_EOLICA, LOW);
    delay(2000);  // Esperar 2 segundos

    // 2. Activar resistencia frenado (relé GPIO23)
    digitalWrite(PIN_RELE_FRENO, HIGH);

    Serial.println("✓ Freno activado");
}
```

Desactivación:

```
void desactivarProteccion() {
    // Verificar que condiciones son seguras
    if (viento < 20 && voltaje < 60 && rpm < 450) {

        // Desactivar freno
        digitalWrite(PIN_RELE_FRENO, LOW);
        delay(5000);  // Esperar 5 segundos

        // Reconectar turbina
        digitalWrite(PIN_RELE_EOLICA, HIGH);
```

```
        Serial.println("✅ Sistema restaurado");
    }
}
```

Interlock (Seguridad):

```
// NUNCA tener relé eólica Y freno activos juntos
void setRelayFreno(bool estado) {
    if (estado && relays.eolica) {
        // Desconectar eólica primero
        digitalWrite(PIN_RELE_EOLICA, LOW);
        delay(100);
    }
    digitalWrite(PIN_RELE_FRENO, estado);
}
```

---

14. PROTECCIÓN DE BATERÍA

14.1 Zona Óptima (25-80%)

Beneficios:

```
Mantener SOC entre 25-80%:
- Maximiza vida útil (5,000+ ciclos)
- Reduce estrés químico
- Evita sobrecarga
- Evita descarga profunda

Fuera de zona óptima:
- 0-25%: Daño por descarga profunda
- 80-100%: Estrés por sobrecarga
```

Ecuación de Degradación:

```
Ciclos_vida = Ciclos_base × Factor_DoD × Factor_temp

Factor_DoD:
DoD 80%: Factor = 1.00 (5,000 ciclos)
DoD 50%: Factor = 1.60 (8,000 ciclos)
DoD 30%: Factor = 2.40 (12,000 ciclos)

Factor_temp (referencia 25°C):
15°C: Factor = 1.30
25°C: Factor = 1.00
35°C: Factor = 0.70
45°C: Factor = 0.40

Ejemplo:
```

```
Batería LiFePO4, DoD 50%, temp 35°C:
Ciclos = 5,000 × 1.60 × 0.70 = 5,600 ciclos
```

14.2 Estrategia de Uso

Prioridad 1: Uso Directo

```
Solar + Eólica → Consumo (SIN batería)

Ventajas:
- Eficiencia 100% (sin conversión)
- Sin degradación batería
- Máximo aprovechamiento

Condición:
IF (Generación >= Consumo) THEN
    Usar_directo = TRUE
    Cargar_batería = FALSE
END IF
```

Prioridad 2: Cargar Batería (Solo 25-80%)

```
IF (Excedente > 0 AND SOC < 80%) THEN
    Cargar_batería = TRUE
END IF

IF (SOC >= 80%) THEN
    Cargar_batería = FALSE
    Desviar_excedente_a_resistencia_dump()
END IF
```

Prioridad 3: Batería como Respaldo

```
IF (Generación < Consumo AND SOC > 25%) THEN
    Usar_batería = TRUE
END IF

IF (SOC <= 25%) THEN
    Activar_red_backup()
    // O reducir cargas no esenciales
END IF
```

14.3 Cálculos de Protección

Corriente Máxima de Carga:

```
I_max_carga = C_batería × C-rate

Para LiFePO4:
C-rate típico: 0.5C (carga en 2 horas)
```

```
Ejemplo batería 200Ah:
I_max = 200 × 0.5 = 100A

Potencia máxima:
P_max = V × I = 48V × 100A = 4,800W
```

Corriente Máxima de Descarga:

```
I_max_descarga = C_batería × C-rate_descarga

Para LiFePO4:
C-rate descarga: 1.0C (descarga en 1 hora)

Ejemplo:
I_max = 200 × 1.0 = 200A
P_max = 48V × 200A = 9,600W
```

Tiempo de Autonomía:

```
t_autonomía = (C_batería × SOC_actual × DoD) / P_consumo

Ejemplo:
Batería: 10 kWh
SOC actual: 60%
DoD permitido: 35% (hasta 25%)
Consumo: 500W

Energía disponible:
E = 10 × 0.60 × 0.35 = 2.1 kWh

Tiempo:
t = 2.1 / 0.5 = 4.2 horas
```

---

15. MACHINE LEARNING

15.1 Datos de Entrenamiento

Fuente: NASA POWER API

```
Período: 10 años (2014-2024)
Frecuencia: Mensual
Total muestras: 10 años × 12 meses = 120 registros

Variables (features):
1. Mes (1-12)
2. Día del año (1-365)
3. Temperatura (°C)
4. Humedad (%)
```

```
5. Presión (kPa)
6. Cielo despejado histórico
7. Viento histórico

Target (salida):
- Solar: Irradiancia (kWh/m²/día)
- Eólico: Velocidad viento (m/s)
```

15.2 Modelos Implementados

Modelo Solar: Random Forest Regressor

```python
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(
    n_estimators=100,      # 100 árboles
    max_depth=15,          # Profundidad máxima
    min_samples_split=5,
    random_state=42
)

# Entrenar
model.fit(X_train, y_train)

# Predecir
y_pred = model.predict(X_test)
```

Modelo Eólico: Gradient Boosting

```python
from sklearn.ensemble import GradientBoostingRegressor

model = GradientBoostingRegressor(
    n_estimators=100,
    max_depth=5,
    learning_rate=0.1,
    random_state=42
)

model.fit(X_train, y_train)
```

15.3 Métricas de Evaluación

R² Score (Coeficiente de Determinación):

```
R² = 1 - (SS_residual / SS_total)

Interpretación:
R² = 1.00 → Predicción perfecta
R² = 0.87 → Explica 87% de la variación
R² = 0.50 → Regular
```

```
R² = 0.00 → Modelo inútil

Ejemplo:
Modelo Solar: R² = 0.872 (87.2% precisión)
```

MAE (Mean Absolute Error):

```
MAE = (1/n) × Σ|y_real - y_pred|

Ejemplo Solar:
MAE = 0.35 kWh/m²/día

Interpretación:
Si predicción es 4.5 kWh/m²/día
→ Real estará entre 4.15 y 4.85
```

RMSE (Root Mean Square Error):

```
RMSE = √[(1/n) × Σ(y_real - y_pred)²]

Penaliza errores grandes más que MAE

Si RMSE > MAE:
→ Hay algunos errores grandes
Si RMSE ≈ MAE:
→ Errores distribuidos uniformemente
```

15.4 Feature Importance

Ejemplo Real:

```
Modelo Solar:
cielo_despejado:     45.3% 
mes:                 28.1% 
temperatura:         12.4% 
dia_año:              8.2% 
humedad:              3.5%
presion:              1.8%
viento_historico:     0.7%

Interpretación:
- El "cielo despejado" es el factor MÁS importante
- El mes también importa mucho (estacionalidad)
- Los demás factores son menos relevantes
```

15.5 Validación Cruzada

5-Fold Cross Validation:

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(
    model, X, y,
    cv=5,           # 5 folds
    scoring='r2'
)

print(f"R² medio: {scores.mean():.3f}")
print(f"Desv. std: {scores.std():.3f}")

Ejemplo resultado:
R² scores: [0.891, 0.867, 0.854, 0.876, 0.868]
R² medio: 0.871
Desv. std: 0.012

Interpretación:
 Modelo consistente (baja desviación)
 No hay overfitting
 Generaliza bien
```

---

16. ESTRATEGIAS INTELIGENTES

16.1 Decisión Automática de Fuentes

Algoritmo:

```
def decidir_fuente():
    # Leer datos
    solar = sensores.potencia_solar
    eolica = sensores.potencia_eolica
    consumo = sensores.consumo
    soc = bateria.soc

    # Regla 1: Uso directo si hay suficiente
    total_renovable = solar + eolica
    if total_renovable >= consumo:
        return "uso_directo"

    # Regla 2: Batería si SOC > 25%
    deficit = consumo - total_renovable
    if soc > 25:
        return "bateria"

    # Regla 3: Red backup
    return "red_backup"
```

Priorización:

```
Prioridad 1: Solar (más estable)
Prioridad 2: Eólica (variable)
Prioridad 3: Batería (zona 80-25%)
Prioridad 4: Red backup

Excepciones:
- Si viento fuerte → Priorizar eólica
- Si batería baja → Cargar primero
- Si noche → Solo eólica + batería
```

16.2 Predicción y Optimización

Predicción 24 Horas:

```python
def optimizar_24h():
    # Obtener pronóstico
    forecast = openweather.get_forecast()

    # Predecir generación hora por hora
    for hour in range(24):
        solar_pred = ml.predict_solar(hour)
        wind_pred = ml.predict_wind(hour)
        consumo_pred = patterns.predict_consumption(hour)

        # Calcular balance
        balance = solar_pred + wind_pred - consumo_pred

        if balance < 0:
            # Déficit esperado
            acciones.append("cargar_bateria_previo")
        else:
            # Excedente esperado
            acciones.append("usar_directo")

    return acciones
```

Aprendizaje de Patrones:

```python
def aprender_patrones():
    # Recopilar datos históricos
    data = db.get_last_30_days()

    # Detectar patrones por hora
    for hour in range(24):
        consumo_hora = data.filter(hour=hour)
        promedio = consumo_hora.mean()
        desviacion = consumo_hora.std()

        patterns[hour] = {
            "promedio": promedio,
            "min": promedio - desviacion,
```

```
            "max": promedio + desviacion
        }

    # Identificar horas pico
    horas_pico = sorted(patterns,
                        key=lambda h: patterns[h]["promedio"],
                        reverse=True)[:5]

    return patterns, horas_pico
```

16.3 Ecuaciones de Optimización

Función Objetivo:

```
Minimizar: Costo_total + Degradación_batería

Costo_total = Σ(P_red × precio_kWh)
Degradación = Σ(Ciclos × costo_reemplazo)

Restricciones:
- SOC_min ≤ SOC ≤ SOC_max
- I_carga ≤ I_max
- P_total ≥ P_consumo
```

Balance de Potencias:

```
P_solar + P_eolica + P_bateria + P_red = P_consumo + Pérdidas

Donde:
Pérdidas = P_inversor + P_cables + P_controlador

P_inversor = P_consumo × (1 - η_inversor)
η_inversor típico = 0.95 (5% pérdidas)
```

FIN PARTE 4

El manual continúa con la Parte 5: Instalación y Uso

✅ RESUMEN PARTES 1-4:

Parte 1: Arquitectura, Hardware ESP32, Sensores (20 págs) **Parte 2:** Backend, NASA API, OpenWeather (15 págs) **Parte 3:** Dimensionamiento Solar/Eólico/Batería (25 págs) **Parte 4:** Protecciones, ML, Estrategias (20 págs)

TOTAL: ~80 páginas técnicas completas ✅

Para generar PDF:

1. Usar https://dillinger.io/
2. O ejecutar: `CONVERTIR_A_PDF.bat`
3. O VSCode + extensión "Markdown PDF"