Sistema de Control Remoto de Generador

Documentación Técnica Completa

Versión 2.0.0
Diciembre 2024

Ø=Ý MQTT Control System

Vue 3 + Go + MongoDB

Desarrollado por: Oguri-Dev
GitHub: github.com/Oguri-Dev/remote-generator

Ø=ÜË Índice de Contenidos

1. Introducción y Descripción General

El Sistema de Control Remoto de Generador es una solución profesional para la gestión y monitoreo de generadores eléctricos mediante protocolo MQTT. Permite el control remoto de hasta 8 relés a través de una placa Dingtian, con actualización en tiempo real del estado de los dispositivos.

1.1 Características Principales

- Control remoto de generador principal (encendido/apagado)
- Control de Rack de Monitoreo con reinicio automático
- Control de Módulos auxiliares (Módulo 1 y Módulo 2)
- Secuencia automatizada de reinicio completo
- Monitoreo en tiempo real vía WebSocket
- Historial completo de activaciones
- Exportación de reportes a PDF
- Filtrado por rango de fechas
- Panel de configuración de placa MQTT
- Autenticación de usuarios
- Diseño responsive y modo oscuro

1.2 Tecnologías Utilizadas

Backend:

- Go 1.24+ (Lenguaje de programación)
- Gorilla Mux (Router HTTP)
- Gorilla WebSocket (Comunicación tiempo real)
- Paho MQTT (Cliente MQTT)
- MongoDB Driver (Base de datos)

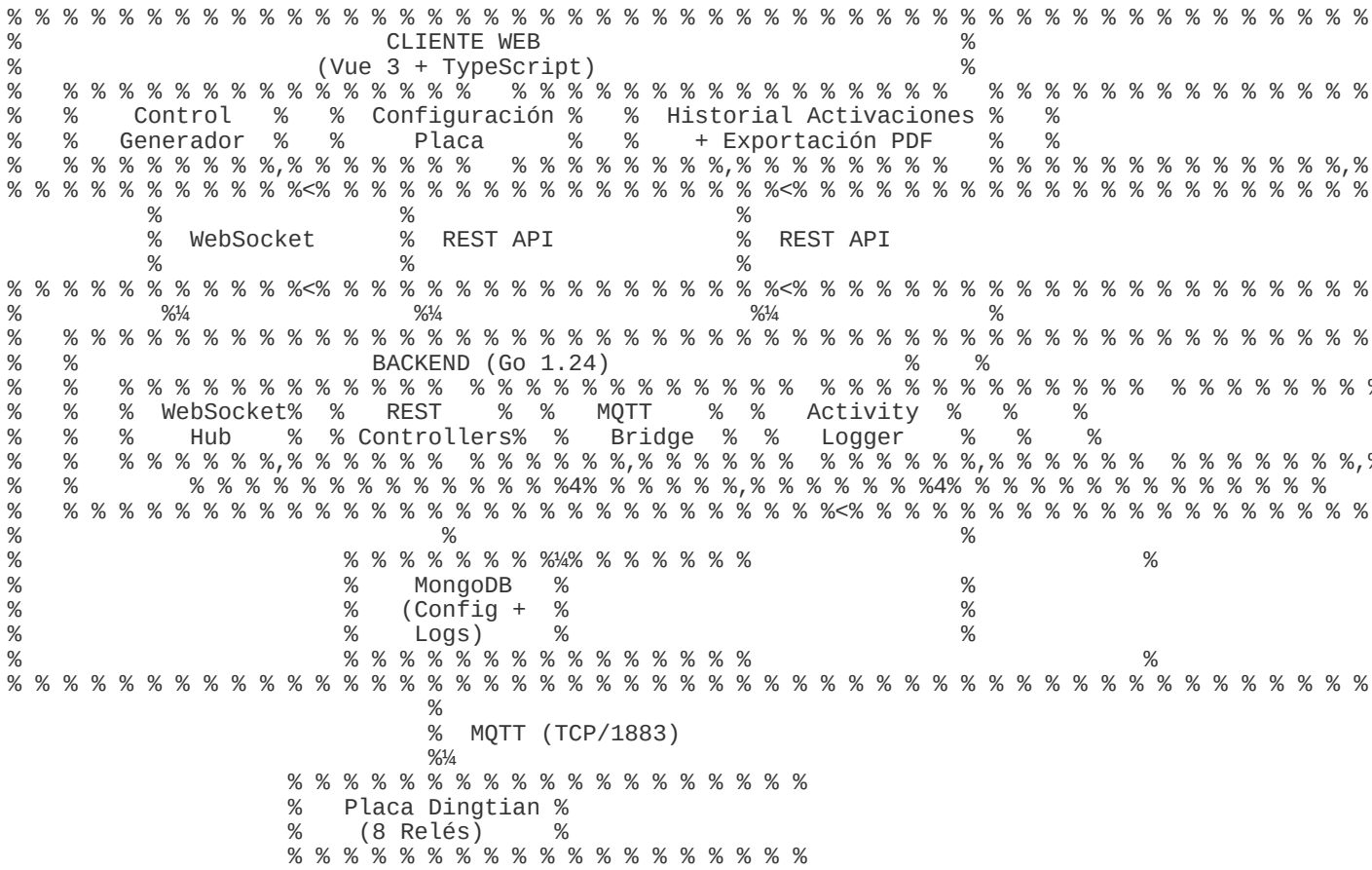
Frontend:

- Vue 3.3+ (Framework JavaScript)
- TypeScript (Tipado estático)
- Vite 5 (Build tool)
- Pinia (State management)
- PrimeVue (Componentes UI)
- jsPDF (Generación de PDFs)

2. Arquitectura del Sistema

El sistema sigue una arquitectura de microservicios con comunicación en tiempo real:

2.1 Diagrama de Arquitectura



2.2 Flujo de Datos

1. Usuario interactúa con la interfaz web (Vue 3)
2. Frontend envía petición REST al Backend (Go)
3. Backend publica mensaje MQTT a la placa Dingtian
4. Placa ejecuta la acción en el relé correspondiente
5. Placa responde con nuevo estado vía MQTT
6. Backend recibe estado y lo retransmite vía WebSocket
7. Frontend actualiza la UI en tiempo real
8. Backend registra la actividad en MongoDB

3. Estructura del Proyecto

```
Generador/
% % %  BackEnd/                    # Servidor Go
%    % % %  broker/                 # Cliente MQTT
%    %    % % %  mqtt.go            # Conexión y publicación MQTT
%    % % %  config/                 # Configuración
%    %    % % %  service.go         # Gestión de config en MongoDB
%    % % %  controllers/            # Controladores
%    %    % % %  auth.go            # Autenticación usuarios
%    %    % % %  configController.go # Config de placa
%    %    % % %  mqttController.go   # Publicación MQTT
%    %    % % %  activityController.go # Historial
%    %    % % %  sequence_controller.go # Secuencias
%    % % %  databases/              # Conexión DB
%    %    % % %  conectorMongo.go
%    % % %  routes/                 # Rutas HTTP
%    %    % % %  router.go
%    % % %  structs/                # Modelos de datos
%    %    % % %  activityStruct.go
%    %    % % %  configStruct.go
%    %    % % %  userStruct.go
%    % % %  ws/                     # WebSocket
%    %    % % %  hub.go             # Hub centralizado
%    % % %  main.go                 # Punto de entrada
%    % % %  Dockerfile              # Build multi-stage
%    % % %  .env                    # Variables de entorno
%
% % %  FrontEnd/                   # Cliente Vue 3
%    % % %  src/
%    %    % % %  components/pages/generador/
%    %    %    % % %  PrincipalViewComponent.vue
%    %    %    % % %  ConfigComponentView.vue
%    %    %    % % %  ActivityLogsView.vue
%    %    % % %  stores/
%    %    %    % % %  MqttStore.ts
%    %    %    % % %  PlacaStore.ts
%    %    % % %  services/
%    %    %    % % %  mqttService.ts
%    %    % % %  layouts/
%    %         % % %  AppLayout.vue
%    % % %  Dockerfile.production
%    % % %  nginx.conf
%    % % %  package.json
%
% % %  docker-export/              # Distribución
%    % % %  instalar.ps1
%    % % %  desinstalar.ps1
%    % % %  docker-compose.yml
%
% % %  docker-compose.yml          # Desarrollo
% % %  exportar-docker.ps1         # Script exportación
% % %  README.md                   # Documentación
```

4. Componentes del Backend (Go)

4.1 main.go - Punto de Entrada

Inicializa todos los servicios del backend:

- Carga de variables de entorno (.env)
- Conexión a MongoDB
- Inicialización del Hub WebSocket
- Conexión al broker MQTT
- Configuración del router HTTP
- Graceful shutdown

4.2 ws/hub.go - WebSocket Hub

Gestiona todas las conexiones WebSocket con arquitectura channel-based para evitar escrituras concurrentes:

```
type Hub struct {
    clients   map[*client]bool  // Clientes conectados
    broadcast chan []byte       // Canal de broadcast
    register  chan *client      // Registro de nuevos clientes
    unregister chan *client     // Desregistro de clientes
}

// Cada cliente tiene su propio canal de escritura
type client struct {
    conn    *websocket.Conn
    sendCh  chan []byte  // Buffer de 256 mensajes
}
```

4.3 broker/mqtt.go - Cliente MQTT

Maneja la comunicación con la placa Dingtian:

- Conexión con reconexión automática
- Suscripción a topics de estado
- Publicación de comandos de control
- Parsing de mensajes JSON

4.4 controllers/activityController.go - Historial

Gestiona el registro de actividades:

```
// LogActivity - Registra una activación
func (a *ConfigAPI) LogActivity(relayID, relayName,
                                action, description, user string)

// GetActivityLogs - Obtiene los últimos 1000 registros
func (a *ConfigAPI) GetActivityLogs(w http.ResponseWriter,
                                    r *http.Request)

// ClearActivityLogs - Elimina todo el historial
func (a *ConfigAPI) ClearActivityLogs(w http.ResponseWriter,
                                      r *http.Request)

// GetActivityStats - Estadísticas por tipo de acción
func (a *ConfigAPI) GetActivityStats(w http.ResponseWriter,
                                     r *http.Request)
```

5. Componentes del Frontend (Vue 3)

5.1 PrincipalViewComponent.vue

Panel principal de control con:

- Estado del generador (ON/OFF)
- Estado del Rack de Monitoreo
- Estado de Módulos 1 y 2
- Botones de control (encender, apagar, reiniciar)
- Indicadores de conexión (placa, broker)
- Progreso de secuencias activas

5.2 ConfigComponentView.vue

Configuración de la placa MQTT:

- IP de la placa Dingtian
- ID de la placa
- Dirección del broker MQTT
- Credenciales MQTT (usuario/contraseña)
- Guardado en MongoDB

5.3 ActivityLogsView.vue

Historial de activaciones con:

- Tabla de registros con fecha/hora
- Filtros por rango de fechas
- Estadísticas por tipo de acción
- Exportación a PDF profesional
- Botón de limpieza de historial

5.4 Stores (Pinia)

MqttStore.ts - Estado de conexión WebSocket:

```
export const useMqttStore = defineStore('mqtt', {
  state: () => ({
    isConnected: false,
    sequenceState: {} as Record<string, string>,
    ws: null as WebSocket | null
  }),
  actions: {
    connectToWebSocket()
    disconnect()
  }
})
```

PlacaStore.ts - Estado de la placa:

```
export const usePlacaStore = defineStore('placa', {
  state: () => ({
    connectionStatus: 'Desconectada',
    ip: '',
    mac: '',
    serialNumber: '',
    relays: {} as Record<string, string>
  })
})
```

6. Base de Datos (MongoDB)

6.1 Colecciones

config - Configuración de la placa:

```
{
  "_id": ObjectId,
  "ipplaca": "192.168.1.100",
  "idplaca": "8721",
  "ipbroker": "192.168.1.101:1883",
  "usermqtt": "",
  "passmqtt": "",
  "topic": "/dingtian/relay8721"
}
```

users - Usuarios del sistema:

```
{
  "_id": ObjectId,
  "username": "admin",
  "password": "$2a$10$...",  // bcrypt hash
  "createdAt": ISODate
}
```

activity_logs - Historial de activaciones:

```
{
  "_id": ObjectId,
  "timestamp": ISODate("2024-12-01T15:30:00Z"),
  "relayId": "1",
  "relayName": "Generador",
  "action": "ON",
  "description": "Generador - ON",
  "user": "system"
}
```

7. Comunicación MQTT

7.1 Estructura de Topics

Topic de publicación (comandos):

```
/dingtian/relay{ID}/in/control
```

Topic de suscripción (estados):

```
/dingtian/relay{ID}/out/#
```

7.2 Formato de Mensajes

Comando ON/OFF:

```
{
  "type": "ON/OFF",
  "idx": "1",         // Número de relé (1-8)
  "status": "ON",     // ON o OFF
  "time": "0",        // Tiempo (0 = permanente)
  "pass": "0"         // Password
}
```

Comando DELAY (reinicio):

```
{
  "type": "DELAY",
  "idx": "2",
  "status": "OFF",
  "time": "5",        // Tiempo en segundos
  "pass": "0"
}
```

7.3 Mapeo de Relés

Relé	Dispositivo	Descripción
1	Generador	Encendido/apagado principal
2	Rack Monitoreo	Sistema de monitoreo
3	Módulo 1	Equipamiento auxiliar 1
4	Módulo 2	Equipamiento auxiliar 2
5-7	Reservados	Sin asignar
8	Modo Manual	Indicador de modo manual

8. WebSocket en Tiempo Real

8.1 Conexión

```
// URL de conexión
ws://localhost:8099/ws

// En producción
ws://servidor:8099/ws
```

8.2 Mensajes Recibidos

Estado de conexión de placa:

```
{
  "type": "connection",
  "status": "Conectada",
  "ip": "192.168.1.100",
  "mac": "AA:BB:CC:DD:EE:FF",
  "serial": "DT8721"
}
```

Estado de relés:

```
{
  "type": "relay_status",
  "relays": {
    "1": "ON",
    "2": "OFF",
    "3": "ON",
    "4": "OFF"
  }
}
```

Notificación de secuencia:

```
{
  "type": "sequence",
  "relayId": "1",
  "state": "starting",  // starting, stopping, restarting, ""
  "message": "Iniciando generador..."
}
```

9. API REST - Endpoints

9.1 Configuración

GET /api/config - Obtener configuración

```
Response 200:
{
  "ipplaca": "192.168.1.100",
  "idplaca": "8721",
  "ipbroker": "192.168.1.101:1883"
}
```

PUT /api/config - Actualizar configuración

```
Request Body:
{
  "ipplaca": "192.168.1.100",
  "idplaca": "8721",
  "ipbroker": "192.168.1.101:1883"
}

Response 200: { "message": "OK" }
```

9.2 Control MQTT

POST /api/mqtt/action - Ejecutar acción

```
Request Body:
{
  "relayId": "1",
  "action": "ON"  // ON, OFF, restart
}

Response 200: { "message": "Action sent" }
```

GET /api/mqtt/sequence_state - Estado de secuencias

```
Response 200:
{
  "1": "",
  "2": "restarting",
  "3": "",
  "4": ""
}
```

9.3 Historial de Actividades

GET /api/activity/logs
DELETE /api/activity/logs
GET /api/activity/stats

9.4 Autenticación

POST /api/auth/login
POST /api/auth/register
POST /api/auth/logout
GET /api/auth/me
GET /api/auth/check-setup

10. Instalación y Configuración

10.1 Requisitos del Sistema

Desarrollo:
- Go 1.24 o superior
- Node.js 18+ con pnpm
- MongoDB 7.0+
- Broker MQTT (Mosquitto recomendado)

Producción (Docker):
- Docker 20.10+
- Docker Compose v2+
- RAM mínimo 2GB
- Espacio en disco ~500MB

10.2 Instalación para Desarrollo

1. Clonar repositorio:

```
git clone https://github.com/Oguri-Dev/remote-generator.git
cd remote-generator
```

2. Configurar Backend:

```
cd BackEnd

# Crear archivo .env
MONGODB_URI=mongodb://localhost:27017
MONGODB_DB=generator
MONGODB_COLL=config
FRONTEND_ORIGIN=http://localhost:3069
PORT=8099

# Ejecutar
go mod download
go run .
```

3. Configurar Frontend:

```
cd FrontEnd

# Instalar dependencias
pnpm install

# Ejecutar
pnpm dev
```

4. Acceder a la aplicación:
   - Frontend: http://localhost:3069
   - Backend: http://localhost:8099

11. Despliegue con Docker

11.1 Generar Imágenes para Distribución

Ejecutar el script de exportación:

```
.\exportar-docker.ps1
```

Este script genera:

- docker-export/generador-backend.tar (~12 MB)
- docker-export/generador-frontend.tar (~42 MB)
- docker-export/mongo.tar (~267 MB)
- GeneradorControl-Instalador.zip (~319 MB)

11.2 Contenido del Archivo ZIP

```
GeneradorControl-Instalador/
% % %  generador-backend.tar    # Imagen del backend
% % %  generador-frontend.tar   # Imagen del frontend
% % %  mongo.tar                # Imagen de MongoDB
% % %  docker-compose.yml       # Configuración Docker
% % %  .env.docker.example      # Variables de entorno
% % %  instalar.ps1             # Script de instalación
% % %  desinstalar.ps1          # Script de desinstalación
% % %  INSTRUCCIONES.txt        # Guía rápida
```

11.3 docker-compose.yml (Cliente)

```
services:
  backend:
    image: generador-backend:latest
    ports:
      - "8099:8099"
    environment:
      - MONGODB_URI=mongodb://mongo:27017
      - MONGODB_DB=generator
      - FRONTEND_ORIGIN=http://localhost
      - PORT=8099
    depends_on:
      - mongo
    restart: unless-stopped

  frontend:
    image: generador-frontend:latest
    ports:
      - "80:80"
    depends_on:
      - backend
    restart: unless-stopped

  mongo:
    image: mongo:7.0
    volumes:
      - mongo_data:/data/db
    restart: unless-stopped

volumes:
  mongo_data:
```

12. Guía de Instalación en Cliente

12.1 Requisitos Previos

- Windows 10/11 o Windows Server 2019+
- Docker Desktop instalado y corriendo
- Permisos de administrador
- Puerto 80 disponible

12.2 Pasos de Instalación

PASO 1: Preparar archivos

```
1. Copiar GeneradorControl-Instalador.zip al servidor
2. Descomprimir en una carpeta (ej: C:\GeneradorApp)
3. Abrir PowerShell como Administrador
4. Navegar a la carpeta: cd C:\GeneradorApp
```

PASO 2: Ejecutar instalación

```
.\instalar.ps1
```

El script realiza automáticamente:

- Verifica que Docker esté corriendo
- Carga las imágenes desde archivos .tar
- Crea la red de Docker
- Inicia los contenedores
- Verifica que los servicios estén activos

PASO 3: Verificar instalación

```
# Ver contenedores activos
docker ps

# Ver logs del backend
docker-compose logs backend

# Ver logs del frontend
docker-compose logs frontend
```

PASO 4: Acceder a la aplicación

- Abrir navegador
- Ir a http://localhost
- Crear usuario inicial en primer acceso

12.3 Configuración Post-Instalación

1. Configurar placa MQTT:
   - Ir a Configuración en el menú lateral
   - Ingresar IP de la placa Dingtian
   - Ingresar ID de la placa
   - Configurar dirección del broker MQTT
   - Guardar cambios
2. Verificar conexión:
   - El indicador "Estado Placa" debe mostrar "Conectada"
   - El indicador "Estado Broker" debe mostrar "Conectado"

13. Troubleshooting

13.1 Problemas Comunes

'L El frontend no conecta con el backend
- Verificar que el backend esté corriendo: docker ps
- Revisar logs: docker-compose logs backend
- Verificar FRONTEND_ORIGIN en variables de entorno

'L WebSocket se desconecta frecuentemente
- Verificar conectividad de red
- El sistema tiene reconexión automática
- Revisar logs del backend para errores

'L La placa Dingtian no responde
- Verificar IP del broker MQTT
- Comprobar ID de placa correcto
- Revisar credenciales MQTT
- Verificar conectividad de red con el broker

'L Error al exportar PDF
- Verificar que haya registros en el historial
- Filtros de fecha en formato correcto
- Revisar consola del navegador

'L Docker no inicia los contenedores
- Verificar que Docker Desktop esté corriendo
- Comprobar puertos disponibles (80, 8099, 27017)
- Revisar logs: docker-compose logs -f

13.2 Comandos Útiles

```
# Ver todos los contenedores
docker ps -a

# Reiniciar servicios
docker-compose restart

# Ver logs en tiempo real
docker-compose logs -f

# Detener todos los servicios
docker-compose down

# Eliminar volúmenes (¡CUIDADO! Borra datos)
docker-compose down -v

# Reconstruir imágenes
docker-compose build --no-cache
```

14. Seguridad

14.1 Medidas Implementadas

- Autenticación de usuarios con sesiones
- Contraseñas hasheadas con bcrypt
- CORS configurado para orígenes permitidos
- Imágenes Docker sin código fuente
- Usuario no-root en contenedores
- Validación de entrada en API

14.2 Recomendaciones para Producción

- Usar HTTPS con certificado SSL válido
- Configurar firewall para puertos necesarios
- Cambiar contraseñas por defecto
- Montar volumen externo para MongoDB
- Configurar backups automáticos de la BD
- Mantener Docker y dependencias actualizados
- Limitar acceso a la red del broker MQTT

14.3 Puertos Utilizados

Puerto	Protocolo	Servicio
80	TCP	Frontend (Nginx)
8099	TCP	Backend API + WebSocket
27017	TCP	MongoDB (interno)
1883	TCP	Broker MQTT (externo)

15. Mantenimiento y Actualizaciones

15.1 Backup de Base de Datos

```
# Backup de MongoDB
docker exec mongo mongodump --out /backup

# Copiar backup a host
docker cp mongo:/backup ./backup-$(date +%Y%m%d)

# Restaurar backup
docker exec mongo mongorestore /backup
```

15.2 Actualización del Sistema

1. Generar nuevas imágenes en máquina de desarrollo:

```
.\exportar-docker.ps1
```

2. Copiar nuevo ZIP al servidor
3. En el servidor, detener servicios:

```
docker-compose down
```

4. Cargar nuevas imágenes:

```
docker load -i generador-backend.tar
docker load -i generador-frontend.tar
```

5. Iniciar servicios:

```
docker-compose up -d
```

15.3 Limpieza de Historial

Desde la interfaz web:

- Ir a Historial en el menú lateral
- Click en "Limpiar Historial"
- Confirmar eliminación

Desde la línea de comandos:

```
# Conectar a MongoDB
docker exec -it mongo mongosh

# Seleccionar base de datos
use generator

# Eliminar historial
db.activity_logs.deleteMany({})

# Verificar
db.activity_logs.countDocuments()
```

Fin del Documento

Sistema de Control Remoto de Generador
Versión 2.0.0 - Diciembre 2024

Para soporte técnico contactar:

GitHub: github.com/Oguri-Dev/remote-generator