API para gestión de procesos y experimentos en una microrred eléctrica inteligente de laboratorio

GIARDINI, David; EZPELETA, Joaquín; JUNCO, Sergio

Escuela de Ingeniería Electrónica; Facultad de Ciencias Exactas, Ingeniería y Agrimensura; Universidad Nacional de Rosario
LAC, Laboratorio de Automatización y Control
{dgiardini,ezpeleta,sjunco}@fceia.unr.edu.ar

Palabras claves: Microrred inteligente, SCADA, API C++, Gestión de energía

XIX Jornadas de Ciencias, Tecnologías e Innovación
UNR

INTRODUCCIÓN

El LAC desarrolló una **Microrred Eléctrica Inteligente** (**REILAC)**, integrada por convertidores de potencia que permiten el intercambio de energía entre una subred de continua y otra de alterna con generación renovable, almacenamiento y cargas configurables, todo supervisado desde un SCADA. Para habilitar una gestión automática e inteligente, se creó una **API en C++**, que facilita la programación de algoritmos de control mediante una capa de abstracción para el acceso a las variables del sistema. Con arquitectura cliente-servidor, esta API simplifica la codificación, admite múltiples agentes de gestión y permite operar en distintas computadoras de la red local.

ARQUITECTURA

El sistema tiene una arquitectura sistema cliente-servidor.
El **servidor**, denominado gestror_srv.exe, accede a las variables del SCADA mediante estándares **OPC/DDE** y las expone a través de un **servidor UDP/IP**.
Los **clientes**, desarrollados en C++ con la librería **LAC-Grid++**, instancian objetos de la clase **Bloque**, que pueden ser de tipo **entrada, salida, constante o PI**, con posibilidad de ampliarse a otros tipos. Los bloques de **entrada** adquieren automáticamente los valores de las variables mediante un **hilo en background** que realiza las solicitudes al servidor, mientras que los bloques de **salida** solicitan la escritura de variables al servidor de forma periódica.
Gracias a la **sobrecarga de operadores**, es posible realizar operaciones algebraicas, control de flujo y cálculos complejos de manera sencilla, lo que permite trasladar diagramas de bloques o de flujo al programa de forma clara y legible.

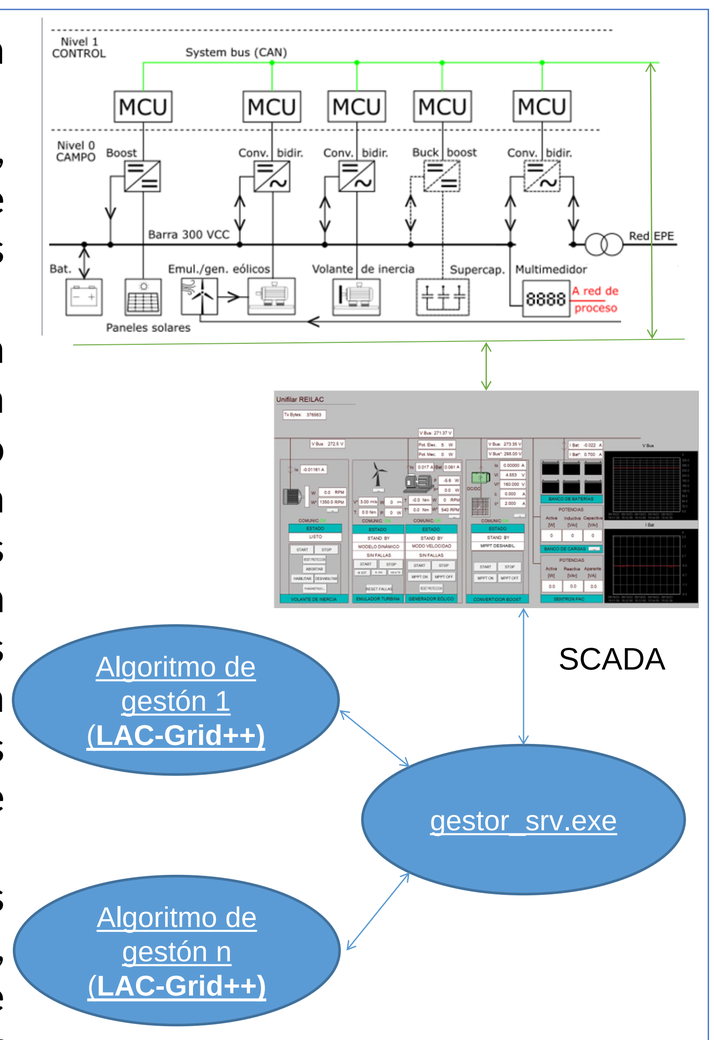

Figura 1. Arquitectura del sistema con gestión de energía vía LAC-Grid++

PROGRAMACIÓN LAC-Grid++

La librería **LAC-Grid++** está basada en **programación orientada a objetos** y hace uso extensivo del **polimorfismo** mediante la **sobrecarga de operadores**. Los elementos de tipo **Bloque** pueden interconectarse entre sí utilizando notación algebraica, lo que otorga al usuario potencia y flexibilidad. De esta forma, no es necesario ser un programador experto en C++, sino contar con conocimientos en **gestión energética** para aprovechar sus capacidades

```
friend Bloque operator+(Bloque a, Bloque b){
    Bloque temp(TIPO_TEMP);
    temp.valor_interno=a.salida() + b.salida();
    return temp;
}
friend float operator+(double x, Bloque& b){
    return (float)x+b.valor_interno;
}
friend float operator+(Bloque& b,double x){
    return b.valor_interno+(float)x;
}
```

Figura 2. Detalle de la librería. Sobrecarga del operador suma

USO DE LA LIBRERÍA

Al desarrollador se le proporciona una **plantilla de programación** que contiene el código común necesario para cualquier implementación, simplificando el inicio del desarrollo. Así, su tarea principal consiste en definir los distintos **bloques constitutivos** del sistema y en implementar la **lógica de control o gestión energética** que desee aplicar. La plantilla ya incluye la estructura de comunicación con el servidor, la actualización de variables de entrada y salida, y los hilos de ejecución necesarios, por lo que el usuario no debe preocuparse por estos detalles de bajo nivel.
Una vez completada la definición de bloques y la lógica, el algoritmo se **compila** utilizando el comando **make**, generando un ejecutable listo para integrarse con la microrred y comenzar a gestionar. Esto permite que usuarios con conocimientos en **gestión energética** puedan implementar y probar sus estrategias sin necesidad de experiencia avanzada en programación en C++.

```
/*************************************************************************
Este código se llama cíclicamente.
VBUS e IBAT son objetos Bloque tipo entrada
SP_RPM es objeto Bloque tipo salida. ERR es objeto Bloque genérico
PI_VBUS y PI_IBAT son objetos Bloque tipo controlador PI
*************************************************************************/
if(VBUS>=285){
    if(ctrl_i_cte){    // El algoritmo está en modo regulación en corriente...
        ctrl_i_cte=0;  // ...ahora regula tensión
        PI_VBUS.SetIngtegrador()=SP_RPM; // Se re-inicializa el integrador
                                         // del PI para evitar discontinuidades
    }
    ERR=(285-VBUS); // Valor de error de tensión
    PI_VBUS=ERR;    // Alimentación del bloque PI de tensión
    SP_RPM=PI_VBUS; // Nuevo valor deseado de RPMs del generador eólico

}else{
    if(ctrl_i_cte==0){  // El algoritmo está en modo regulación en tensión...
        ctrl_i_cte=1;   // ...ahora regula corriente
        PI_IBAT.SetIngtegrador=SP_RPM; // Se re-inicializa el integrador
    }

    ERR=SP_IBAT-IBAT;// Valor de error de tensión
    PI_IBAT=ERR;     // Alimentación del bloque PI de corriente
    SP_RPM=PI_IBAT;  // Nuevo valor deseado de RPMs del generador eólico
}
```

Figura 3. Programa de usuario para carga de baterías a corriente constante y tensión flotante usando el generador eólico

CONCLUSIONES

El enfoque orientado a objetos y el uso de sobrecarga de operadores en LAC-Grid++ permitió implementar algoritmos de gestión energética de forma clara y legible. La arquitectura cliente-servidor, junto con la automatización de adquisición y escritura de variables, facilitó el desarrollo de soluciones escalables y flexibles.
La combinación de plantillas predefinidas y bloques interconectables redujo la necesidad de conocimientos avanzados en programación, permitiendo a los usuarios centrarse en la lógica de gestión de la energía en la microrred.
En conjunto, estas características potencian la eficiencia, la modularidad y la rapidez en la creación, prueba y despliegue de estrategias de gestión energética, promoviendo tanto la experimentación como la implementación práctica.