

Informe Final Capa de Alto Nivel del Poyecto Redes y Microncontroladores

Autor: Nahuel Defossé, Lautaro Pecile

Fecha: Marzo 2010

Introducción

El objetivo de la capa de alto nivel del proyecto "Microcontroladores e Internet" es la adquisición y persistencia de datos, además de proveer un mecanismo de representación de la información adquirida permitiendo el envío de ordenes (comandos) como lazo de realimentación humana sobre el sistema. Este tipo de software se conoce como sistema SCADA (*Supervisory Control and Data Acquisition*). Un sistema SCADA generalmente se acompaña de algún tipo de interfase que permite la interacción con el usuario, denominado HMI, (*Human Machine Interface*).

Se estableció también como objetivo la utilización de *Software Libre*, para el desarrollo íntegro del software, con el objeto de garantizar la independencia del fabricante y bajo coste de implantación de la tecnología.

El proyecto inicialmente se orientó en el desarrollo de una terminal de consulta y envío de comandos similar a Arquicom utilizando el lenguaje Python y el toolkit gráfico Qt. Se comprobó durante esta etapa que el lenguaje Python a pesar de ser de muy alto nivel, está preparado para trabajar en comunicaciones donde se deben realizar operaciones a nivel de bit con bastante facilidad. Una vez superadas las pruebas iniciales, se plantaron los requerimientos generales como visualización de eventos, ubicación de las terminales (RTU) en mapa o esquema, y alarmas.

Orientación a Eventos

Los lenguajes de muy alto nivel orientados a objetos son ejecutados sobre máquinas virtuales. Generalmente estas proveen el mecanismo de hilos (*Threads*) para lograr concurrencia, pero están limitados en varios aspectos [StephDoyle2009].

Los hilos son la herramienta más común a la hora de realizar concurrencia y consisten en la capacidad de poder ejecutar concurrentemente varios fragmentos de código por parte de un programa. Un ejemplo puede ser la atención de varios clientes, o en el caso de el presente proyecto, la consulta a varios concentradores.

Un hilo consiste en un proceso o método (en un lenguaje OO), el cual es ejecutado por un programador o scheduler apropiativo provisto por el sistema operativo. No se puede saber a ciencia cierta en que momento se va a ejecutar cada hilo y se suelen usar primitivas de sincronización como semáforos o monitores para garantizar la integridad de los datos (debido a que todos los hilos comparten su estado/datos).

En el caso de una aplicación orientada a eventos, la concurrencia se logra a través de la subdivisión voluntaria en el momento de la programación en pequeños fragmentos de código (funciones o métodos) que son ejecutados por un scheduler, generalmente apodado reactor en honor a un patrón de diseño de concurrencia [WikiReactor]. En este enfoque, no se requieren primitivas de sincronización debido a que se trata de multitarea cooperativa.

En general las arquitecturas orientadas a eventos en entornos con alta carga de entrada salida son más beneficiosas que los hilos. Se puede tomar como ejemplo de este hecho, el proyecto Apache [MINA].

El framework de red orientado a eventos [Twisted] fue elegido para la implementación de la capa de comunicaciones con las unidades remotas, debido a que provee un enfoque simple y orientado a objetos de la multitarea colaborativa.

Software de Adquisición de Datos

Una vez seleccionado el framework de comunicaciones asincrónicas, se decidió el método de acceso a la base de datos. En vez de seleccionar una base de datos particular, se optó por utilizar un mapeador objeto-relacional, que consiste en una capa de software que presenta la base de datos como objetos.

Si bien el lenguaje de acceso a las bases de datos, conocido como SQL está normado por ISO/ANSI, cada fabricante incorpora sutiles diferencias que hacen que se requieran pequeños arreglos para poder manejar una base de datos u otra. Los mapeadores objeto relacionales realizan una abstracción por sobre estas diferencias.

Se seleccionó como mecanismo de acceso a la base de datos el mapeador objeto relacional [SQLAlchemy] debido a que permite un nivel de detalle muy alto a la hora de realizar los mapeos, posee varios subproyectos de interés como Elixir y tiene una extensa comunidad.

Las tablas definidas para la adquisición de datos fueron las siguientes:

- CO - Concentrador
Posee un número y la dirección IP
- UC - Unidad de Control
Posee un número de UC, una relación hacia el CO al que pertenece
- SV - Variable de Sistema
Representa una variable de sistema, posee como atributo el número (posición), y está relacionada con la UC a la que pertenece.
- DI - Variable de Sistema
Representa una variable digital de un bit, posee como atributo el número (posición), y está relacionada con la UC a la que pertenece.
- AI - Variable de Sistema
Representa una variable analógica, posee como atributo el número (posición), y está relacionada con la UC a la que pertenece.
- EV - Evento
Representa un evento en el sistema, posee la impronta de tiempo impuesta por la unidad de control, la relación con la unidad de control, el tipo de evento y el valor (representan pasajes de 0 a 1 o de 1 a 0).

Ciertas variables de configuración fueron almacenadas en archivos con formato legible para el humano, provista por el paquete [config]:

```
picnet :
{
  tcp_port : 9761
  id_rs485 : 1
}
scada :
{
  verbosity : 1
  type : 'sqlite3'
  user : ''
  password : ''
  host : ''
  dbname : 'dsem.db'
  options : ''
}
```

```
save_di : False
save_ai : False
save_ev : True
save_sv : False
}
```

A este software se lo bautizó como Alsvid en honor a un corcel de la mitología escandinava, su nombre significa "*muy velóz*".

Integración de SCADA/HMI: Sistema Avrak

Se desarrolló como actividad de extensión una implementación sobre el motor de adquisición de datos una interfase hombre máquina (HMI) para un sistema de semaforización.

Por motivos de practicidad se decidió embeber el motor SCADA en la aplicación HMI, obteniéndose un único ejecutable que solo depende de el motor de base de datos. En este caso se seleccionó MySQL.

Este sistema presenta sobre un mapa de la ciudad el conjunto de puntos que representan el estado de cada esquina. Para lograr este objetivo se agregaron nuevas tablas a la base de datos:

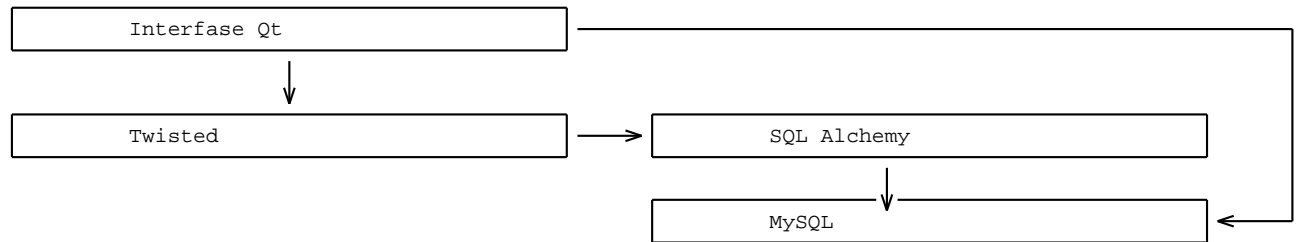
- Calle
Almacena el nombre de la calle y un identificador entero
- Esquina
Almacena las relaciones con esquina calle y la posición en el mapa
- Esquina_Calles
Almacena el nombre de las calles, el sentido y el ángulo
- Semáforo
Almacena la relación con la UC, el tipo de movimiento y el subtipo de movimiento permitiendo definir un semáforo vehicular, semáforo de giro y semáforo peatonal.

La interfase visual consistió en 4 pestañas:

- Mapa
Permite la visualización de las diferentes esquinas de la ciudad
- Eventos
Listado de eventos, permitiendo filtrado en función de su fecha y su tipo, además de relizar impresiones.
- Configuración
Permite la definición de Unidades de Control y Concentradores.

En este proyecto se realizó un desarrollo de una interfase de configuración gráfica mediante Qt. Se puede ver un ejemplo de la funcionalidad en el siguiente video [[YouTubeAvrak](#)].

El siguiente digrama expone la arquitectura del sistema:



Este proyecto se bautizó Avrak, otro corcel de la mitología escandinava.

Definición de SCADA basado en Web y HTML5

Avrak constituyó una prueba de que una arquitectura de multitarea colaborativa puede ser usada en conjunto con otros componentes de software para lograr un sistema de monitoreo y control, sin embargo cuenta con la desventaja de que el sistema depende en buena parte de código específico para la plataforma como la biblioteca gráfica Qt y la base de datos.

Se planteó como mejora la utilización del protocolo HTTP como mecanismo de acceso y control a los datos y la separación de el manejo de el protocolo de comunicaciones (picnet) de la aplicación específica de control (semáforos en el caso de Avrak). El nuevo objetivo será acceder a los datos tanto con un browser (navegador) como con aplicaciones que puedan generar y consumir HTTP (aplicaciones de escritorio, celulares, demonios de tele-control, etc.).

Para lograr exponer los datos mediante HTTP se planteó la definición de una API.

WikiReactor	Patrón de diseño reactor http://en.wikipedia.org/wiki/Reactor_pattern
StephDoyle2009	http://softwareramblings.com/2008/07/multi-thread-scaling-issues-with-python-and-ruby.html
MINA	http://mina.apache.org/performance-test-reports.html
Twisted	http://twistedmatrix.com/trac/
config	http://pypi.python.org/pypi/config/0.3.7
SQLAlchemy	http://www.sqlalchemy.org/
YouTubeAvrak	http://www.youtube.com/watch?v=GS1JYdZlrK4