Uso de los sockets en Ada

El programa que controla el funcionamiento de la maqueta de trenes debe leer la posición de los trenes para determinar si hay riesgo de colisión y en que momento debe accionar los cambios de aguja.

Dicha información es elaborada en el computador que maneja las cámaras y se almacena en ciertos campos de una estructura de tipo tren_t. De estos campos se toma el nombre del tren, que puede ser tg para el tren grande o tp para el pequeño. El nombre ocupa los 2 primeros caracteres del string, luego se pone un separador que es el carácter :.

A continuación se especifica la región en la que se encuentra dicho tren, dichas regiones son:

r3b, ar1, ar2, ar3, ar4, ar5, ar6, ar7, ar8, ar9, r10.

Estas regiones coinciden con las descritas en el gráfico de la maqueta con una salvedad que es que la región r3 del esquema se divide en 2 regiones que son r3b, desde el cambio de vías de las terminales hasta el cambio de vía del óvalo interior, la región r3 propiamente dicha alcanza desde este punto hasta la región r4. Como se aprecia el nombre de cada región ocupa exactamente 3 caracteres. Luego viene nuevamente el separador :

Por último se ofrece, expresada en milímetros, la distancia del centro de gravedad del tren, medida en línea recta, al siguiente separador entre regiones hacia el cual se dirige. Esta distancia se expresa en forma de número entero de 4 dígitos, rellenando con espacios en blanco las posiciones de la izquierda que no tienen valor.

Un mensaje como

tg:ar4: 56 En total 11 caracteres

Significa que el tren grande está en la región 4 y dista de la próxima región 56 mm.

tp:r10:1200 En total 11 caracteres

Significa que el tren pequeño está en la región 10 y dista de la próxima región 1200 mm.

Se pueden aprovechar los atributos image y value de los tipos escalares para pasar de un tipo a string y de string al tipo empleando la facilidad del Ada de trabajar con fragmentos de arrays unidimensionales. Para ilustrar este aspecto os muestro la forma en que se construye el mensaje en el computador que procesa las imágenes.

```
procedure escribe_socket(tren : tren_t ) is
   mensaje: string(1..11);
   aux     : integer;
   begin
      mensaje(1..2) :=tren.nombre;
      mensaje(3..3) :=":";
      mensaje(4..6) :=Region_t'Image(tren.reg);
      mensaje(7..7) :=":";
      aux:=integer(tren.distancia_aislador);
      if (aux > 999)       then
         mensaje(8..11):=integer'image(aux)(2..5);
```

```
      elsif ( (aux > 99 ) and (aux  < 999 )) then
         mensaje(9..11):=integer'image(aux)(2..4);
         mensaje(8):=' ';          -- 1 blanco
      elsif ( (aux > 9 ) and (aux  < 99 )) then
         mensaje(10..11):=integer'image(aux)(2..3);
         mensaje(8..9):="  ";      -- 2 blancos
      else
         mensaje(11..11):=integer'image(aux)(2..2);
         mensaje(8..10):="   ";          -- 3 blancos
      end if;
      Put_line(Outgoing_Socket,mensaje);
   end Escribe_Socket;
```

Aquí se emplea 'image para construir el string, para extraer la información del string se empleará 'value

Por otra parte es preciso inicializar lo sockets antes de usarlos, para ello se incluirá

```
with Sockets.Stream_IO; use Sockets, Sockets.Stream_IO;
```

Se declarará una variable de tipo socket

```
Outgoing_Socket : Socket_FD;
```

Y se inicializarán los sockets

```
procedure Init_Socket is
begin
    Socket (Outgoing_Socket, PF_INET, SOCK_STREAM);
    Connect (Outgoing_Socket, "remotehost", 50000);
end Init_Socket;
```

La comunicación entre los computadores se establece empleando un cable cruzado. Se prepara a los computadores para que las direcciones ip de los mismos sean:

10.1.1.1 para el control de la maqueta

10.1.1.2 para el que procesa las imágenes.

Desde el punto de vista del receptor es preciso:

```
with Sockets.Stream_IO; use Sockets, Sockets.Stream_IO;    --Traer el paquete
Sock: Socket_FD;        -- Declarar unas variables de tipo Socket_FD;
Accepting_Socket: Socket_FD;
Incoming_Socket: Socket_FD;
Mensaje: String(1..11);      -- String para almacenar el mensaje del socket
```

-- Operaciones para preparar el socket

```
Socket(Accepting_Socket, PF_INET;SOCK_STREAM);
Setsockpot(Accepting_Socket, SOL_SOCKET, SO_REUSEADDR,1);
Bind (Accepting_Socket);
Listen (Accepting_Socket);


Mensaje:=Get_Line(Sock);      -- leer el mensaje
Shutdown(Sock,Both);          -- cerrar el socket al cortar la conexión
```

Un ejemplo de utilización se puede ver en el archivo detector.adb. En este el procedimiento principal inicializa el socket y crea dinámicamente una tarea que tiene un punto de cita en el que recibe un parámetro de tipo Socket_FD. Extrae de él el mensaje y lo analiza. No hace falta que se siga el mismo esquema que en el ejemplo.