MODULE Module1

VAR string data;

VAR socketdev server\_socket;

VAR socketdev client\_socket;

VAR string recibir:="";

PERS string server\_ip := "172.16.78.65";

PERS num send\_port := 8007;

VAR socketstatus sockStat;

VAR bool conectado := FALSE;

VAR jointtarget jt;

VAR num ok;

CONST num SERVER\_BAD\_MSG := 0;

CONST num SERVER\_OK := 1;

PERS string recv\_data:="POSX266.69Y-231.67Z365.78QS0.00Qi0.00Qj1.00Qk0.00 ";

VAR num tool := 0;

VAR num tool\_old := 0;

VAR num pos := 0;

VAR num long := 1;

VAR intnum siglint;

VAR intnum conf\_interrup;

PERS bool confirmacion;

PERS bool confirmacion2:=FALSE;

PROC main()

!Conexion del socket e inicializacion de variables

conectado := FALSE;

recv\_data := "";

ServerCreateAndConnect server\_ip, send\_port;

conectado := TRUE;

confirmacion2 := FALSE;

CONNECT siglint WITH reset\_routine;

!Interrupcion asociada a la activacion de la señal de entrada DoServerReset (input 9),

!encargada de reiniciar los sockets

ISignalDI DoServerReset,1,siglint;

CONNECT conf\_interrup WITH conf\_routine;

!Interrupcion asociada a la puesta a True de la variable persistente confirmacion,

!encargada de iniciar el proceso de envio de confirmacion al programa

IPers confirmacion,conf\_interrup;

WHILE TRUE DO

sockStat := SocketGetStatus(client\_socket);

IF sockStat = SOCKET\_CONNECTED THEN

!Si la conexion con el cliente es correcta, se procede a escuchar lo que este envie

SocketReceive client\_socket,\Str:=recibir \Time:=30;

long := StrLen(recibir);

pos := StrFind(recibir,1," ");

!Si los mensajes se acumulan, se separan por los espacios colocados entre ellos y se

!toma el último

WHILE pos <> long DO

pos := StrFind(recibir,1," ");

long := long - pos;

recibir := StrPart(recibir,pos+1,long);

ENDWHILE

IF recibir <> "DUMMY " THEN

!Se guardan los datos recibidos en la variable recv\_data, siempre que no sean

!la palabra "DUMMY ", empleada para establecer la conexion cliente-servidor

recv\_data := recibir;

ENDIF

ENDIF

ENDWHILE

!Gestor de errores

ERROR

IF ERRNO=ERR\_SOCK\_CLOSED THEN

TPWrite "SERVIDOR: El cliente ha cerrado la conexion.";

conectado := FALSE;

WaitTime(0.01);

!Cierre del servidor

SocketClose client\_socket;

SocketClose server\_socket;

!Reinicio del servidor

ServerCreateAndConnect server\_ip, send\_port;

conectado := TRUE;

ok := SERVER\_OK;

RETRY;

ELSEIF ERRNO=ERR\_SOCK\_TIMEOUT THEN

TPWrite "SERVIDOR: Se ha sobrepasado el tiempo máximo de espera";

conectado := FALSE;

WaitTime(0.01);

!Cierre del servidor

SocketClose client\_socket;

SocketClose server\_socket;

!Reinicio del servidor

ServerCreateAndConnect server\_ip, send\_port;

conectado := TRUE;

!reconnected := TRUE;

ok := SERVER\_OK;

RETRY;

ENDIF

recv\_data := " ";

ENDPROC

PROC ServerCreateAndConnect(string ip, num port)

!Funcion encargada de crear el servidor y establecer la conexion con el cliente

VAR string clientIP;

SocketCreate server\_socket;

SocketBind server\_socket, ip, port;

SocketListen server\_socket;

TPWrite "SERVIDOR: Esperando conexiones entrantes...";

WaitTime(0.01);

WHILE SocketGetStatus(client\_socket) <> SOCKET\_CONNECTED DO

SocketAccept server\_socket,client\_socket \ClientAddress:=clientIP \Time:=WAIT\_MAX;

IF SocketGetStatus(client\_socket) <> SOCKET\_CONNECTED THEN

TPWrite "SERVIDOR: Problema con conexión entrante.";

WaitTime(0.01);

TPWrite "Intentando reconectar.";

WaitTime(0.01);

ENDIF

!Espera 0.5 segundos para intentar reconectar

WaitTime 0.5;

ENDWHILE

!TPWrite "SERVIDOR: Conectado a la IP " + clientIP;

ENDPROC

TRAP reset\_routine

!Rutina encargada de reiniciar el servidor mediante la activacion y desactivacion

!de la señal "DoServerReset", asignada a la entrada 9

TPWrite "SERVIDOR: Reinicio de Servidor solicitado";

WaitTime(0.01);

WaitDI DoServerReset, 0;

conectado := FALSE;

!Cierre del servidor

SocketClose client\_socket;

SocketClose server\_socket;

!Reinicio del servidor

ServerCreateAndConnect server\_ip, send\_port;

conectado := TRUE;

!reconnected := TRUE;

ok := SERVER\_OK;

TPWrite "SERVIDOR: Reinicio completado";

WaitTime(0.01);

ERROR

RAISE;

ENDTRAP

TRAP conf\_routine

!Rutina encargada de enviar la confirmacion al programa

IF confirmacion = TRUE THEN

SocketSend client\_socket \str := "AccionRealizada";

WaitTime(0.05);

confirmacion2 := TRUE;

WHILE confirmacion = TRUE DO

WaitTime(0.01);

ENDWHILE

ENDIF

confirmacion2 := FALSE;

ENDTRAP

ENDMODULE