Centinela

Monitorización sistemas TCP/IP con Arduino

Objeto del proyecto y descripción

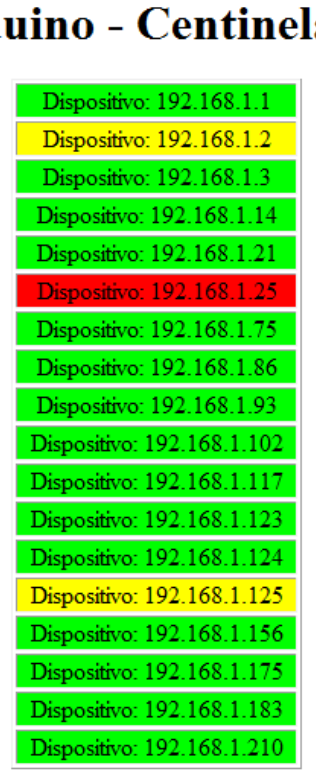
En instalaciones de sistemas informáticos, equipos de control, seguridad, equipos industriales, etc. que están asociados a redes ethernet que usan TCP/IP, puede ser delicado el que algunos equipos dejen de estar operativos.

El proyecto Centinela estará dirigido a monitorizar las IPs de los dispositivos 'sensibles' en una red mediante el protocolo ICMP y avisar a un usuario mediante de las incidencias que se produzcan en el funcionamiento. Aparte de esto, se registrarán los sucesos en una memoria SD y se podrá consultar el estado de los dispositivos por Web.

Los datos de la red a explorar se guardarán en un fichero de texto en la memoria SD, de forma que sea fácilmente configurable desde un PC antes de iniciar el proceso y así el programa servirá para cualquier instalación.

El uso inmediato para este proyecto será la monitorización de una red de 24 puntos de acceso y de dos servidores de un centro TIC de la Junta de Andalucia. Existen aplicaciones que podrían hacer esta tarea en un servidor, pero el acceso a los servidores está restringido por el CGA. Se da el caso de averías en los puntos de acceso de la red, con pérdidas parciales de cobertura y sobrecarga en la red Wifi. Este sistema comunicará la incidencia de pérdida de IP y se podrá actuar de forma inmediata sin que la cobertura WiFi se vea mermada.

Para avisar de una incidencia, se cuenta con una salida conectada a un LED (en la realidad se conectará a un dispositivo intermitente) que nos avisa de que se ha producido algún error. El estado de los dispositivos lo podemos conocer a través de la página Web suministrada por el Arduino. El cuadrado en verde indica que el dispositivo contesta adecuadamente, el color amarillo indica algún fallo ocurrido (puede deberse a una pérdida ocasional de paquetes), el cuadrado pasará a rojo si se producen 5 intentos consecutivos erróneos.



El tiempo de intervalo de exploración está configurado a 15 segundos para cada dispositivo, dejando el resto del tiempo para atender las peticiones Web. En nuestro planteamiento no tiene sentido 'bombardear' a los dispositivos continuamente, si no vigilarlos con cierta constancia, un retraso de dos o tres minutos en la exploración no supone ningún problema.

En caso de pérdida de corriente, recuperación de equipos, etc. queda registrada en la memoria SD los eventos ocurridos, tanto en error como en recuperación. La intención inicial era grabar en el registro SD la fecha/hora del evento, pero como se comenta en las observaciones, no ha sido posible integrar un cliente NTP o un chip RTC con el Arduino por la memoria disponible.

Fichero de configuración y registro

El fichero de configuración se guarda en la SD con el nombre CONFIG.ARD, es un fichero de texto donde grabaremos los parámetros para la aplicación de forma sencilla y con la posibilidad de incluir comentarios. Es muy importante respetar el formato de las IPs para que Arduino la interprete de forma correcta. Se muestra a continuación el contenido de un fichero,
Su sencillez hace innecesaria la descripción de los parámetros:

```
;-----------------------------
;--  IPs Centinela Arduino  --
;-----------------------------
I:192.168.008.007#
N:255.255.248.000#
G:192.168.008.001#
;MAC Arduino
M:144.162.218.000.121.190#
;Dispositivos
D00:192.168.008.001#
D01:192.168.008.002#
D02:192.168.011.203#
D03:192.168.012.210#
D04:192.168.010.214#
D05:192.168.011.215#
D06:192.168.014.246#
D07:192.168.011.209#
D08:192.168.015.234#
D09:192.168.011.229#
D10:192.168.008.237#
D11:192.168.013.250#
D12:192.168.010.205#
D13:192.168.008.241#
D14:192.168.011.201#
;FIN DISPOSITIVOS
; I -> IP Arduino
; N -> Netmask
; G -> Gateway
; Dispositivos Dxx , donde Dxx será el identificador del dispositivo.
```

El fichero de registro, guarda los eventos de error y de recuperación que se produzcan durante el funcionamiento del dispositivo.

```
Error IP: 192.168.1.2
Error IP: 192.168.1.203
Error IP: 192.168.1.210
Error IP: 192.168.1.214
Error IP: 192.168.1.215
```

```
Error IP: 192.168.1.246
Error IP: 192.168.1.209
IP reactivada: 192.168.1.210
Error IP: 192.168.1.246
Error IP: 192.168.1.209
```

Código fuente

```
//--------------------------------------------------------------------------------
// PROYECTO CENTINELA - MONITOR TCP/IP DISPOSITIVOS - CURSO ARDUINO - GABRIEL ASENSIO
//--------------------------------------------------------------------------------
#define INTERVAL 15000    // Intervalo en segundos entre PING y PING
#define Sirena 2          // Pin de salida para avisador
// Cargamos librerias necesarias Ethernet y tarjeta SD
#include <SD.h>
#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>

// ¡¡¡ PRECAUCIÓN ETHERNET Y SD !!!
// On the Ethernet Shield, CS is pin 4. Note that even if it's not
// used as the CS pin, the hardware CS pin (10 on most Arduino boards,
// 53 on the Mega) must be left as an output or the SD library
// functions will not work.
const int chipSelectSD = 4;
const int chipSelectEt = 10;
// Establecer banderas de Problema/s
boolean FlagError[30];
boolean Avisador;
byte Puntero=0;
byte x=0;
byte y=0;
File myFile;
char Cadena[100];
byte ipahora[4];
// Configuración Ethernet, con valores por defecto
byte ip[4]={192,168,1,55};
byte netmask[4]={255,255,255,0};
byte gateway[4]={192,168,1,1};
// byte nntp[4]={192,43,244,18};
byte mac[6]={144,162,218,0,121,190};
// variables para dispositivos
unsigned long equipo[30];
byte estado[30];
unsigned long hora=0;
// Textos para guardar en memoria Flash de programa
char PROGMEM KBCRA[]="<div align='center'><H1>Arduino - Centinela IP</H1></div>";
char PROGMEM INICIO[]="<table width='30%' border='1' align='center'>";
char PROGMEM divverde[] = "<tr><td bgcolor='#00FF00'><div align='center'>";
char PROGMEM divroja[] = "<tr><td bgcolor='#FF0000'><div align='center'>";
char PROGMEM divama[] = "<tr><td bgcolor='#FFFF00'><div align='center'>";

// Para PING
SOCKET pingSocket = 3;     // Originalmente es '0', pero no se mezcla con NTP/Web bien si es '0'

EthernetServer server(80);

//---------------------------------------------------------------------
byte Extraer(char* Trozo, byte Pos){
   return( (((int)Trozo[Pos]-48)*100) + (((int)Trozo[Pos+1]-48)*10) + ((int)Trozo[Pos+2]-48));
   }
//---------------------------------------------------------------------


//=========================================================
//     Enviar contenido de PROGMEM a client de WebServer
//=========================================================
void contentPrinter(EthernetClient client, char *realword) {
```

```
#define STRING_BUFFER_SIZE 30
  int total = 0;
  int start = 0;
  char buffer[STRING_BUFFER_SIZE];
  int realLen = strlen_P(realword);
  memset(buffer,0,STRING_BUFFER_SIZE);
  while (total <= realLen){
    // Enviar contenido a cliente
    strncpy_P(buffer, realword+start, STRING_BUFFER_SIZE-1);
    client.print(buffer);

    // more content to print?
    total = start + STRING_BUFFER_SIZE-1;
    start = start + STRING_BUFFER_SIZE-1;
  }
}
//============================================================



//--------------------------------------------------------------------
void setup(){
  Serial.begin(9600);
  pinMode(Sirena, OUTPUT);      // Avisador es salida
  // OJO al ChipSelect, solo se puede usar SD o Ethernet
  pinMode(chipSelectEt, OUTPUT);  // Modo, inhabilitado por defecto
  // Ver si hay tarjeta SD.
  if(!SD.begin(chipSelectSD)){   // Abrrir SD
     //-----------------------------------------------
     Serial.println("Error SD");
     return;
     }
     //-----------------------------------------------
  Serial.println("SD Ok");
  // Leer la configuración para programa (IP, MAC, NNTP, mail, etc)
  if(SD.exists("CONFIG.ARD")){
     myFile = SD.open("CONFIG.ARD");
     while(myFile.available() > 0 ){
        Cadena[Puntero]=myFile.read();
        if(Cadena[Puntero]==10){
          // Analizamos la cadena correspondiente según comienzo
          switch(Cadena[0]){
             //-- IP del Arduino --
             case 'I':
                ip[0]=Extraer(Cadena,2);   // Primer byte IP
                ip[1]=Extraer(Cadena,6);   // Segundo byte IP
                ip[2]=Extraer(Cadena,10);   // Tercer byte IP
                ip[3]=Extraer(Cadena,14);  // Cuarto byte IP
                break;
             //--------------------
             //-- NetMask Arduino --
             case 'N':
                netmask[0]=Extraer(Cadena,2);   // Primer byte NM
                netmask[1]=Extraer(Cadena,6);   // Segundo byte NM
                netmask[2]=Extraer(Cadena,10);   // Tercer byte NM
                netmask[3]=Extraer(Cadena,14);  // Cuarto byte NM
                break;
             //--------------------
             //-- Gateway Arduino --
             case 'G':
                gateway[0]=Extraer(Cadena,2);    // Primer byte GW
                gateway[1]=Extraer(Cadena,6);    // Segundo byte GW
                gateway[2]=Extraer(Cadena,10);   // Tercer byte GW
                gateway[3]=Extraer(Cadena,14);  // Cuarto byte GW
                break;
             //--------------------
             //-- NNTP Server (Hora) --
             case 'H':
```

```
               // nntp[0]=Extraer(Cadena,2);    // Primer byte NTP
               // nntp[1]=Extraer(Cadena,6);    // Segundo byte NTP
               // nntp[2]=Extraer(Cadena,10);   // Tercer byte NTP
               // nntp[3]=Extraer(Cadena,14);  // Cuarto byte NTP
               break;
            //--------------------
            //-- MAC del Arduino --
            case 'M':
               mac[0]=Extraer(Cadena,2);    //
               mac[1]=Extraer(Cadena,6);    // Segundo byte MAC
               mac[2]=Extraer(Cadena,10);   //  Tercer byte MAC
               mac[3]=Extraer(Cadena,14);   // Cuarto byte MAC
               mac[4]=Extraer(Cadena,18);   // Cuarto byte MAC
               mac[5]=Extraer(Cadena,22);   // Cuarto byte MAC
               break;
            //-------------------------
            //-- Equipos a verificar --
            case 'D':

equipo[x]=(Extraer(Cadena,4)*0x1000000)+(Extraer(Cadena,8)*0x10000)+(Extraer(Cadena,12)*0x100)+Extraer(
Cadena,16);
               FlagError[x]=0;    // Poner FlagError a '0'
               x++;
               break;
            //--------------------
            case ';':
               // Ignorar línea, se trata de comentarios...
               break;
            }
         //---------------- Fin análisis --------------
         Cadena[0]=' ';
         Puntero=255;    // Como puntero se incrementa lo pongo a 255 ¡¡¡ Cahpuz !!!
         }
       Puntero++;
      }
    }
  else {
      Serial.println(".CFG no existe.");
    }
  //--------------------------- Finalizado CONFIG.ARD, configuramos Ethernet...
  Serial.println(x);            // Ver número de dispositivos hallados
  myFile.close();
  // Cerrar fichero SD
  // Asignar la IP y la MAC al Arduino
  Ethernet.begin(mac, ip, gateway, netmask);
  server.begin();
  // Udp.begin(localPort);
  // Una vez ajustada la tarjeta de Red, se puede comenzar el bucle de exploración
  //-----------------------------------------------------------------------------------------------
}
//-----------------------------------------------------------------------------------------------

//-------------------------------------------------------------------------
void loop(){
   //-------------------------------------------------------------
   // Recorrer la matriz de direcciones a explorar si toca hacerlo
   //-------------------------------------------------------------
   if((hora + INTERVAL) < millis()){
       //=====================================================
       // Verificar IP señalada por el puntero
       // Solo si se ha superado el INTERVALO entre ping
       ICMPPing ping(pingSocket);
       ipahora[3] = (byte) equipo[y];
       ipahora[2] = (byte)(equipo[y] >> 8);
       ipahora[1] = (byte)(equipo[y] >> 16);
       ipahora[0] = (byte)(equipo[y] >> 24);
       ping(1, ipahora, Cadena);
       Serial.println(Cadena);
```

```
      //-------------------------------------
      // Tratamiento en caso de error de PING
      //-------------------------------------
      if(Cadena[0]=='R' && Cadena[1]=='e' && Cadena[2]=='q'){
         // Si hay error, incrementar el número de estado
         if(estado[y]<5){ estado[y]++; }
         // Se ha llegado al límite de reintentos???
         if((estado[y]==5) && (FlagError[y]==false)){ // Si hay error repetido y es primera vez....
            // Averiguar la fecha/hora actual ( I2C PCF8583 ? )
            //-------------------------------------------------------------------------------
            // Grabar SD
            //-----------
            myFile = SD.open("eventos.txt", FILE_WRITE);
            // Si está disponible, escribir dato de ERROR en dispositivo
            if(myFile){
               myFile.print("Error IP: ");
               myFile.print((byte)(equipo[y] >> 24));
               myFile.print('.');
               myFile.print((byte)(equipo[y] >> 16));
               myFile.print('.');
               myFile.print((byte)(equipo[y] >> 8));
               myFile.print('.');
               myFile.println((byte)equipo[y]);
               myFile.close();
               }
            //-----------------------------------------------------------------
            FlagError[y]=true;       // Evitar re-entrada en alarma....
            }   // Hemos llegado a 5, marcamos error
          }
      //-------------------------------------
      // Tratamiento en caso PING correcto
      //-------------------------------------
      if(Cadena[0]=='R' && Cadena[1]=='e' && Cadena[2]=='p'){
         // Ha habido errores anteriormente ??? , si es así decrementar
         if(estado[y]>0){ estado[y]--; }                    // Decrementamos y chequeamos situación
         if((estado[y]==0) && (FlagError[y]==true)){
            // Averiguar la fecha/hora actual ( ver tratamiento socket)
            // Enviar correo y marcar (anulación alarma)
            //------------
            // Grabar SD
            //------------
            myFile = SD.open("eventos.txt", FILE_WRITE);
            // Si está disponible, escribir dato de ERROR en dispositivo
            if(myFile){
               myFile.print("IP reactivada: ");
               myFile.print((byte)(equipo[y] >> 24));
               myFile.print('.');
               myFile.print((byte)(equipo[y] >> 16));
               myFile.print('.');
               myFile.print((byte)(equipo[y] >> 8));
               myFile.print('.');
               myFile.println((byte)equipo[y]);
               myFile.close();
               }
             //-----------------------------------------------------------------
            FlagError[y]=false;       // Evitar re-entrada en finalarma....
            }
        }
      //-------------------------------------------------------------------------------------------
-
      y++;                      // Controlar puntero
      if(y==x){ y=0;}           // Hemos alcanzado el final de la lista
      hora=millis();            // Actualizar contador de intervalo
      //-------------------------------------------------------------------
      // Nos preguntamos si hay que activar el aviso
      boolean Bandera=false;
      for(int w = 0; w < x; w++){ Bandera=(Bandera || FlagError[w]);} // Recorrer  los dispositivos
      if(Bandera){ digitalWrite(Sirena, HIGH);} else {digitalWrite(Sirena, LOW);} // Ajustar 'Sirena'
```

```
       //---------------------------------------------------------------
       }                        // Bucle de intervalo para ping
      //================================================================================================
      //================================================================================================
      // Entre PING y PING escuchar clientes... Web sencillo
      EthernetClient client = server.available();
      if (client){
         // Petición http
         boolean currentLineIsBlank = true;
         while(client.connected()){
            if(client.available()){
               char c = client.read();
               // Si hay una linea en blanco, la petición http ha terminado
               // Es hora de contestar...
               if (c == '\n' && currentLineIsBlank) {
                   // Enviar respuesta estandar http
                   client.println("HTTP/1.1 200 OK");
                   client.println("Content-Type: text/html");
                   client.println();
                   //---------------------------------------
                   // Salidas relacionadas con la petición....
                   //---------------------------------------
                   contentPrinter(client, KBCRA);
                   contentPrinter(client, INICIO);
                   for(int z = 0; z < x; z++) {    // Recorrer todos los dispositivos
                       if(estado[z]==0){ contentPrinter(client, divverde); }
                       if(estado[z]>4) { contentPrinter(client, divroja); }
                       if((estado[z]>0) && (estado[z]<5)) { contentPrinter(client, divama); }
                       client.print("Dispositivo: ");
                       client.print((byte)(equipo[z] >> 24));
                       client.print('.');
                       client.print((byte)(equipo[z] >> 16));
                       client.print('.');
                       client.print((byte)(equipo[z] >> 8));
                       client.print('.');
                       client.print((byte)equipo[z]);
                       client.print("</td></tr>");
                       delay(1);
                       }
                   client.println("</TABLE>");
                   break;
                   }
                if (c == '\n') {
                   // Comenzar una nueva línea
                   currentLineIsBlank = true;
                   }
                else if (c != '\r') {
                   // you've gotten a character on the current line
                   currentLineIsBlank = false;
                   }
                }
              }
           // Dar tiempo al navegador para recibir los datos.
           delay(1);
           // Cerrar conexión
           client.stop();
       }
      //*************************************************************************************
      // delay(500);         // Sustituido por millis() al inicio del bucle

   //-----------------------------------------------------------------------------------
}
//-------------------------------------------------------------------------------------------
```

Observaciones

La idea inicial era usar ICMP para explorar los dispositivos, un cliente NTP para establecer la hora exacta de los fallos en el sistema y usar Webduino para implementar una página Web y el protocolo necesario para enviar un correo de aviso. Aparte de esto, los sucesos se escribirían en la tarjeta SD en un fichero.

El protocolo ICMP es imprescindible para el proceso, pero al agregar el cliente NTP, Arduino se queda sin recursos, con lo cual se hubo de renunciar a la conexión con un servidor NTP para obtener la hora.

El segundo intento de recopilar la hora sin incluir otro Arduino, fue la conexión de un sistema basado en I2C, el chip PCF8583, algo que no fue posible ya que al incluir las librerías Wire, también se alcanzaba el límite del Arduino Uno.

Existe la posibilidad de conectar un reloj en tiempo real SPI, ya que las librerías están ya en uso para la memoria SD y la conexión Ethernet, pero no disponía de un chip RTC para SPI.

Las librerías de Webduino también ocupan un espacio muy importante, por lo que la página Web ha habido que implementarla con un protocolo más simple.

Se ha intentado minimizar el uso de RAM grabando gran parte de las cadenas en memoria de programa PROGMEM, ya que en varias ocasiones el Arduino se ha quedado sin recursos, ya que parece tratar las cadenas de impresión generales como memoria RAM.

Sería interesante usar un Arduino con más capacidad para poder acceder a un servidor de hora y enviar correo de avisos de incidencias vía Webduino. Hoy en día la mayoría de los correos usan comunicaciones seguras que no están implementadas en Arduino, pero se puede usar un servidor Web de Internet como ‘relé’, de forma que se le manda a una página php genérica mediante un POST los datos de servidor SMTP, usuario, contraseña, destinatario, Objeto y texto y el servidor Web mediante PHP, capture los datos y genere el envío del correo usando SSL. Esta era la idea original completa.