webdyn

flexitron group

ROUTER TITAN

Nota de aplicación 81

Implementando una pasarela TRAPS SNMP a mensajes SMS

Implementando una pasarela TRAPS SNMP a mensajes SMS

Detalles del escenario

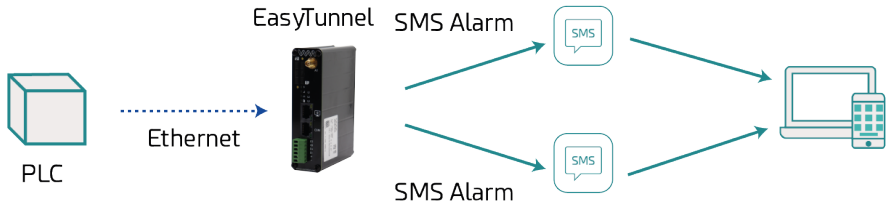
Los router Titan disponen de todas las funcionalidades típicas de un router 4G/3G/2G pero además cuentan con una serie de prestaciones adicionales que lo convierten en uno de los routers con más prestaciones del mercado.

Una de las prestaciones adicionales es la capacidad de implementar una pasarela de TRAPS SNMP a mensajes SMS. Es decir, el router Titan, tras recibir un TRAP SNMP, puede enviar un mensaje SMS hacia uno o varios destinatarios.

Como siempre, se ilustrará esta capacidad con un ejemplo sencillo. En este caso la pasarela debe realizarse mediante un SCRIPT.

1. Descripción del ejemplo

En este ejemplo se va a configurar un dispositivo WEBDYN-EASY-TUNNEL para actuar como una pasarela de TRAP a mensajes SMS. Se dispone de un SAI que envía un TRAP SNMPv2C con OID 1.2.3.4.5.6.7.8.11 cuando detecta una pérdida de alimentación 220Vac y envía también un TRAP SNMPv2c con OID 1.2.3.4.5.6.7.12 cuando su temperatura interna supera los 50ºC. Se pretende que los TRAPS SNMP puedan ser recogidos por el dispositivo WEBDYN-EASY-MODEM a través de su puerto Ethernet y, una vez recibido envíe un mensaje SMS de alarma apropiado (en función del OID recibido, “ALARMA 220V” ó “ALARMA TEMPERATURA”) a los números de teléfono +34666123456 y +34666123457

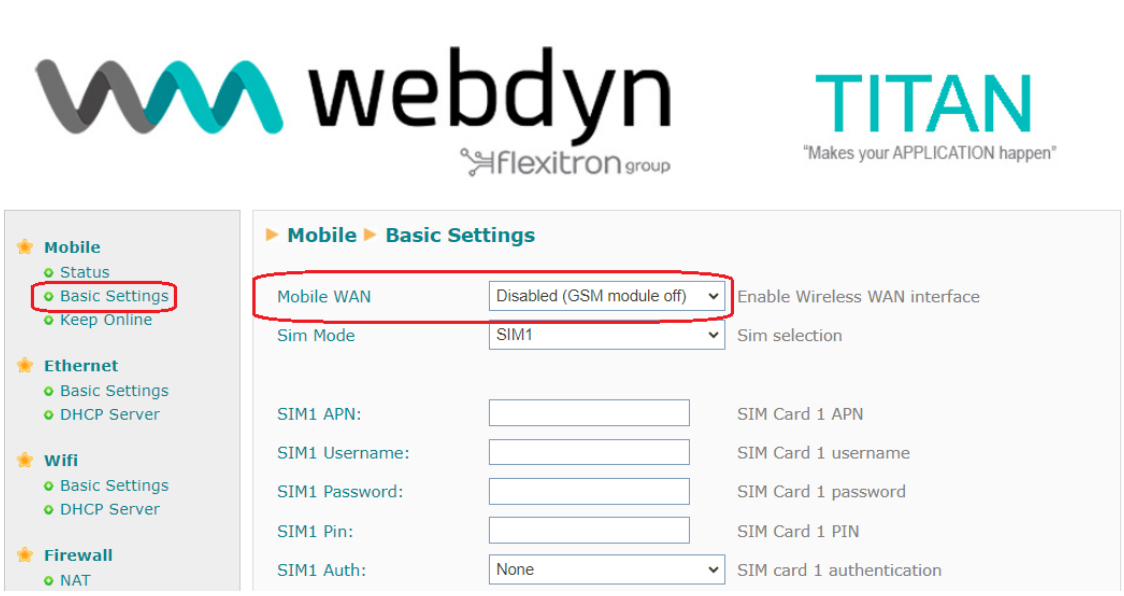


2. Configuración del WEBDYN-EASY-TUNNEL

A continuación se describirá en detalle cómo configurar cada sección del WEBDYN-EASY-TUNNEL para llevar a cabo dicho escenario.

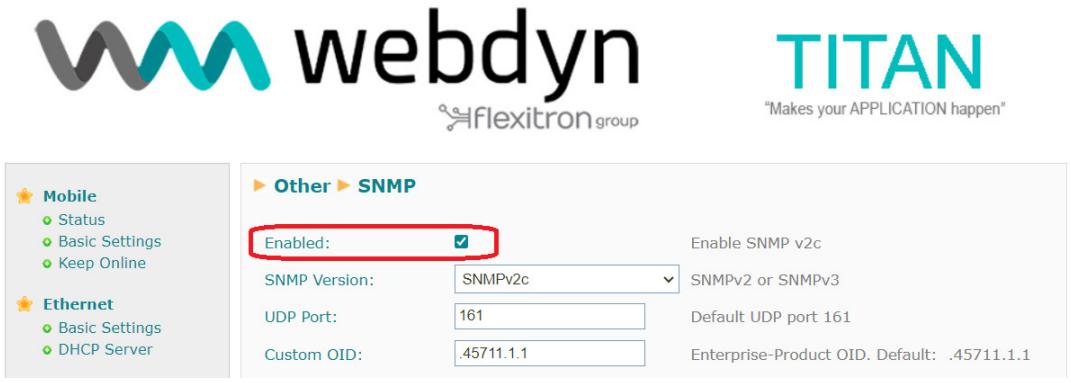
2.1 Configuración Mobile

Para este ejemplo no se precisa uso de tarjeta SIM, por lo que no se va a utilizar. Como el equipo no va a utilizar tarjeta SIM debe configurarse apropiadamente el dispositivo. Para ello, desde el menú de configuración “Mobile - Basic Settings”, en la casilla “Mobile WAN” especificaremos el valor “Disabled (GSM module off)”.



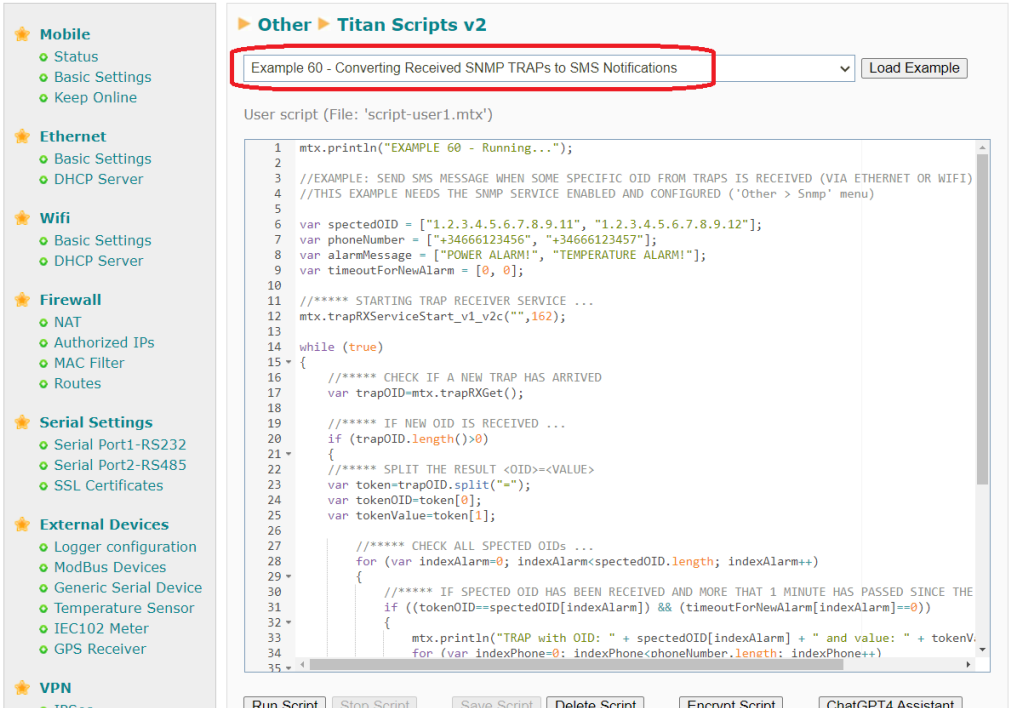
2.2 Configuración del Servicio SNMP

Es necesario también activar el servicio SNMP v2c. Para ello debe acudirse al menú de configuración “Other - SNMP” y configurar la parte apropiada para v2c.



2.3 Configuración del Script de conversión de TRAPS a mensajes SMS

El siguiente paso es programar el script de para la conversión de los TRAPS SNMP recibidos a mensajes SMS. El presente ejemplo es muy similar al ejemplo 60, disponible en el propio equipo, por lo que se recomienda su carga y modificación.



A continuación el script explicado paso a paso.

```
6  var spectedOID = ["1.2.3.4.5.6.7.8.9.11", "1.2.3.4.5.6.7.8.9.12"];
7  var phoneNumber = ["+34666123456", "+34666123457"];
8  var alarmMessage = ["POWER ALARM!", "TEMPERATURE ALARM!"];
9  var timeoutForNewAlarm = [0, 0];
10
```

En el array spectedOID se introducen los OIDs de los TRAPS que se pretenden detectar. En el caso del presente ejemplo el OID “1.2.3.4.5.6.7.8.9.11”, que representa el TRAP de alarma de cuando se pierde la alimentación de 220Vac y el OID “1.2.3.4.5.6.7.8.9.12” que representa el TRAP de alarma de temperatura.

En el array phoneNumber se encuentran los númétos de teléfono a los que enviar las alarmas SMS cuando se detecte un TRAP de alarma.

En el array alarmMessage se encuenta el texto que se incluirá en el SMS para cada alarma. Es decir, para el OID “1.2.3.4.5.6.7.8.9.11” se utiliza el mensaje “POWER ALARM” y para el OID “1.2.3.4.5.6.7.8.9.12” el mensaje “TEMPERATURE ALARM”.

El array timeoutForNewAlarm se usa para almacenar un temporizador de segundos para cada tipo de TRAP OID. Se utilizará para impedir que si un equipo envía varios TRAPS con la misma alarma cada pocos segundos se envíe un SMS por cada uno de ellos. Más adelante se configurará el script para no más de un mensaje SMS por minuto para cada tipo de alarma.

```
11  //***** STARTING TRAP RECEIVER SERVICE ...
12  mtx.trapRXServiceStart_v1_v2c("",162);
```

Esta línea activa el servicio de recepción de TRAPS SNMP de tipo V1 y V2C, sin especificar el community (para recogerlos todos) y en el puerto UDP 162.

```
16      //***** CHECK IF A NEW TRAP HAS ARRIVED
17      var trapOID=mtx.trapRXGet();
```

La función mtx.trapRXGetOID() comprueba si se ha recibido un nuevo TRAP. En caso afirmativo devuelve, uno a uno, los OID del TRAP recibido.

```
19      //***** IF NEW OID IS RECEIVED ...
20      if (trapOID.length()>0)
21      {
22      //***** SPLIT THE RESULT <OID>=<VALUE>
23      var token=trapOID.split("=");
24      var tokenOID=token[0];
25      var tokenValue=token[1];
26
27          //***** CHECK ALL SPECTED OIDs ...
28          for (var indexAlarm=0; indexAlarm<spectedOID.length; indexAlarm++)
29          {
30              //***** IF SPECTED OID HAS BEEN RECEIVED AND MORE THAT 1 MINUTE HAS PASSED SINCE THE
31              if ((tokenOID==spectedOID[indexAlarm]) && (timeoutForNewAlarm[indexAlarm]==0))
32              {
33                  mtx.println("TRAP with OID: " + spectedOID[indexAlarm] + " and value: " + tokenV
34                  for (var indexPhone=0; indexPhone<phoneNumber.length; indexPhone++)
35                  {
36                      mtx.println("Sending Alarm SMS to " + phoneNumber[indexPhone]);
37                      mtx.smsSend(phoneNumber[indexPhone], alarmMessage[indexAlarm]);
38                  }
39                  timeoutForNewAlarm[i] = 600;
40              }
41          }
42      }
```

Este bloque de código muestra las acciones que se ejecutan si se ha detectado un nuevo OID.

Lo primero que se realiza es el split del OID recibido, que se hace con el comando trapOID.split(“=”) ya que el OID recibido se recibe en una estructura <codigoOID>=<valor>. El <códigoOID> se almacena en la variable tokenOID y el <valor> en la variable tokenValue.

Después, básicamente, si el tokenOID recibido coincide con algún OID de los especificados anteriormente en el array “spectedOID” y además la variable timeout contenida en el array “timeoutForNewAlarm” vale 0 (es decir, no se ha enviado un SMS para este tipo de alarma desde hace más de 1 minuto) entonces se envía un mensaje SMS, con el texto apropiado especificado en el array alarmMessage, a todos los números de teléfono contenidos en el array phoneNumber. Seguidamente se actualiza el valor del timeout a 600 (en décimas de segundo) para que no vuelva a enviar mensajes SMS del mismo tipo en ese tiempo.

```
39      //***** UPDATING TIMEOUTS ...
40      for (var i=0;i<timeoutForNewAlarm.length;i++)
41          if (timeoutForNewAlarm[i]>0) timeoutForNewAlarm[i]--;
42
43      //***** PAUSE 100 MILISECONDS
44      mtx.pause(100);
45  }
```

Por último, en el final del bucle while, se actualizan las variables de timeout, restando 1 cada 100ms (aproximadamente).

¿Más dudas?

Escríbenos tus consultas a **iotsupport@mtxm2m.com**