

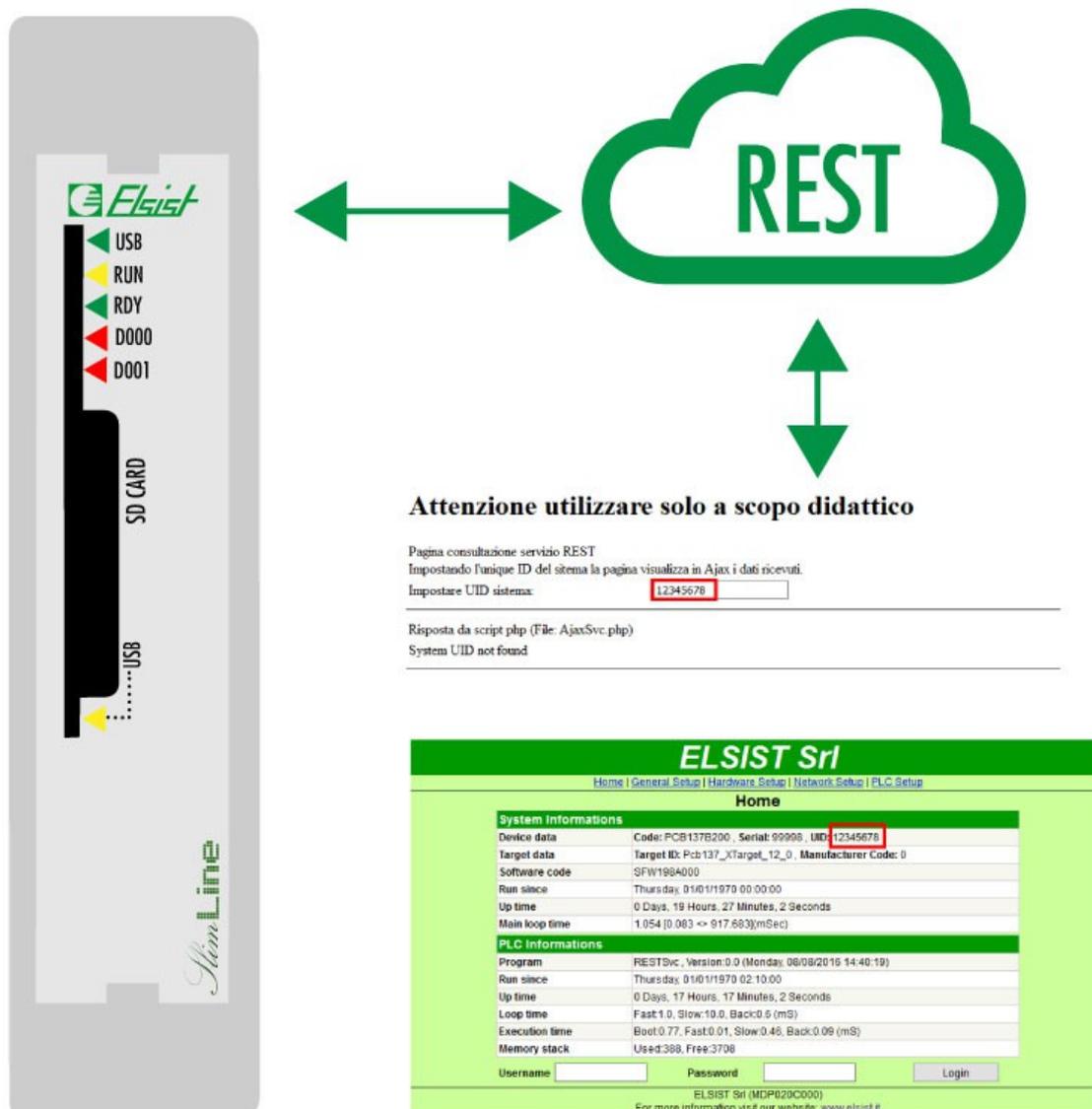
1.1 Libreria gestione servizio REST (eLLabRESTSvcLib)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'[import delle librerie](#).

REST non è un'architettura nè uno standard, ma un insieme di linee guida per la realizzazione di un'architettura di sistema. Tutta la comunicazione tra il client ed il server avviene in HTTP quindi il server REST può essere facilmente ospitato in una server farm. Proprio per testare il servizio abbiamo pubblicato un server su un servizio di hosting gratuito <http://www.slimline.altervista.org>.

Utilizzando il FB **RESTWSvcClient** il sistema invia gli eventi generati nel sistema, ed in assenza di eventi, ciclicamente un frame di heartbeat al server nel cloud e riceve come risposta eventuali comandi dal server. Utilizzando una architettura client si superano tutti i problemi di IP pubblici il sistema funziona anche su reti NATtate in quanto è lui ad aprire la connessione verso l'IP pubblico del server nel cloud.

Appoggiandosi su di un FIFO gli eventi da inviare sono bufferizzati in locale con il timestamp relativo, se c'è connessione l'evento viene inviato al server nel cloud, ma se manca connessione il sistema tenta la connessione ed invia l'evento a connessione ripristinata. Insieme all'evento viene inviato anche il relativo timestamp e questo permette al server di collocare correttamente l'evento nel tempo.



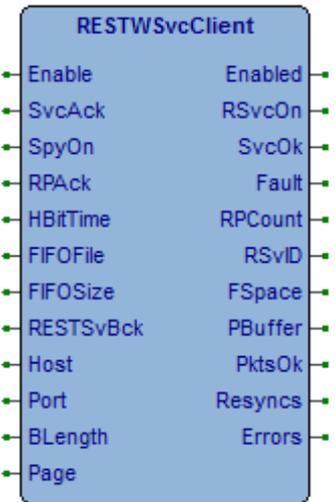
1.1.1 RESTWSvcClient, connects to a REST web service

Type	Library
FB	eLLabRESTSvcLib_A220

Questo blocco funzione gestisce la connessione verso un server REST, per la gestione del protocollo HTTP viene utilizzato il FB [HTTPGetPage](#). Per garantire il buffering dei dati su eventi molto rapidi, il servizio utilizza uno stack FIFO su di un file. In **FIFOFile** ed in **FIFOSize** si può definire quale file utilizzare per appoggiare lo stack FIFO e che dimensione dare allo stack.

E' prevista la gestione dell'invio di un messaggio di heartbeat verso il server ad ogni intervallo di tempo definito in **HBitTime**. Questo permette di controllare il corretto funzionamento del servizio anche in assenza di messaggi da inviare al server. L'uscita **RSvcOn** se attiva indica che il servizio è attivo.

Non essendo possibile gestire all'interno di un FB variabili ritentive, e dovendo gestire i registri dello stack FIFO mantenendoli allo spegnimento del sistema, occorre fornire in **RESTSvBck** l'indirizzo di una variabile di tipo **RESTSERVICEBACKUP** che dovrà essere allocata in area ritentiva.



Enable (BOOL)	Comando attivazione connessione al servizio REST.
SvcAck (BOOL)	Acknowledge ricezione dati. Và attivato per un loop da programma utente sulla ricezione dei dati dal server REST.
SpyOn (BOOL)	Se attivo permette di spiare il funzionamento della FB.
RPAck (USINT)	Numero dati ricevuti da server. Và valorizzato da programma utente contemporaneamente a SvcAck con il numero di parametri ricevuti dal server REST.
HBitTime (UINT)	Tempo invio heartbeat al server (S). Ogni tempo definito viene inviato un messaggio di heartbeat al server.
FIFOFile (STRING[32])	Percorso e nome del file di appoggio dello stack FIFO.
FIFOSize (UDINT)	Dimensione in bytes dello stack FIFO.
RESTSvBck (@RESTSERVICEBACKUP)	Indirizzo allocazione struttura RESTSERVICEBACKUP . La struttura và allocata in memoria ritentiva.
Host (@USINT)	URL del server REST a cui connettersi.
Port (UINT)	Numero di porta del server REST a cui connettersi (Default 80).
BLength (UDINT)	Dimensione buffers richiesta e risposta REST.
Page (@USINT)	Pagina gestione servizio REST.
Enabled (BOOL)	Blocco funzione abilitato.
RSvcOn (BOOL)	Servizio REST attivo. Si attiva su comunicazione con il server REST attiva, viene disattivato se si interrompe la comunicazione, errore su invio di messaggi o heartbeat.
SvcOk (BOOL)	Ok ricezione da server REST. Si attiva su ricezione risposta da server REST e rimane attivo fino a che il programma utente non attiva SvcAck . Con il bit attivo sono disponibili nel buffer puntato da PBuffer i dati ricevuti dal server.
Fault (BOOL)	Attivo per un loop di programma se errore gestione.
RPCount (USINT)	Numero di parametri ricevuti dal server REST.
RSvID (UDINT)	REST service ID. Identificativo servizio REST, da fornire alle FB collegate al servizio.
FSpace (UDINT)	Byte liberi nel file di appoggio FIFO. Indica quanto spazio è rimasto libero nel buffer di appoggio FIFO.
PBuffer (@USINT)	Pointer buffer pagina REST ricevuta. Buffer allocato nel FB tramite funzione RMAIloc , verrà rilasciato alla attivazione di SvcAck .
PktsOk (UDINT)	Counter pacchetti REST correttamente scambiati con il server. Raggiunto il valore massimo il conteggio riprende da 0.

- Resyncs** (UDINT) Counter resincronizzazioni con il server REST. Raggiunto il valore massimo il conteggio riprende da 0.
I messaggi scambiati tra FB e server hanno un identificativo che ne permette il controllo. In caso di disallineamento (Perdita di un messaggio) viene eseguita una resincronizzazione.
- Errors** (UDINT) Counter errori comunicazione con il server REST. Raggiunto il valore massimo il conteggio riprende da 0.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

Codice	Descrizione
10057020	FB eseguita in una task diversa dalla task di background.
10057030	RESTSvBck non è stato definito.
10057050	Errore esecuzione FB HTTPGetPage .
10057100~7	Errore lettura da file FIFO.
10057110	Errore dato letto da file FIFO. Il file FIFO viene reinizializzato.
10057200	Messaggio in FIFO da inviare al server più lungo di BLength (Il messaggio è cancellato).
10057210	Messaggio heartbeat da inviare al server più lungo di BLength (Il messaggio non viene inviato).
10057300	Errore ricezione risposta HTTP da server REST.
10057310	Messaggio ricevuto da server non ha identificativo messaggio "MID".
10057311	Messaggio ricevuto da server non ha numero parametri "RP".
10057312	Messaggio ricevuto da server non ha dati pagina "Page".
10057500	Host ha chiuso la connessione.
10057510	Timeout richiesta pagina.
10057520	Campo Request troppo lungo.
10057550~2	Errore ricezione lunghezza pagina.

1.1.2 RESTSendFct, sends a REST message

Type	Library
FB	eLLabRESTSvcLib_A220

Questo blocco funzione può essere eseguito come una funzione, permette di inserire un messaggio nel buffer FIFO del client REST (FB [RESTWSvcClient](#)) collegato. Il collegamento di questo FB con il FB client avviene con il parametro **RSvID** a cui bisogna fornire il valore in uscita del corrispondente parametro del FB [RESTSvcClient](#).



In **Add** occorre indicare l'indirizzo della stringa messaggio da inviare, eseguendo il FB il messaggio viene inserito immediatamente nel buffer FIFO di inoltro ed il client REST provvederà ad inviarlo al server appena possibile.

- Add** (@USINT) Indirizzo stringa messaggio da inviare.
- RSvID** (UDINT) REST service ID. Identificativo servizio REST, occorre passare alle FB collegate al servizio.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

Codice	Descrizione
10058020	FB eseguita in una task diversa dalla task di background.
10058030	RSvID non è stato definito.
10058040	RSvID non è corretto.
10058100	Il messaggio da inviare è troppo lungo, supera dimensione BLength di FB RESTWSvcClient . Il messaggio viene troncato alla lunghezza definita.
10058110	Non vi è più spazio in buffer FIFO per contenere il messaggio, il messaggio viene perso.
10058200~4	Errore scrittura in file FIFO.

Come testare il servizio REST

Il servizio REST gestito dal sistema provvede a dialogare tramite Internet con un server cloud, quindi per spiegarne il funzionamento abbiamo realizzato un server cloud su un servizio di hosting gratuito <http://www.slimline.altervista.org>.

L'architettura del server è basata su script php mentre la pagina di consultazione htm si appoggia a JQuery ed usa una tecnica Ajax per l'aggiornamento in tempo reale.

L'utilizzo del server cloud è libero a tutti in pratica basta caricare sul sistema SlimLine il programma dimostrativo PTP135A000 che è già configurato per inviare i dati al server cloud.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	DInp	USINT	Auto	[0..1]		..	Digital input byte
2	DOut	USINT	Auto	No		..	Digital output byte
3	i	INT	Auto	No		..	Auxiliary counter
4	LastError	UDINT	Auto	No		..	Last execution error number
5	REST	RESTWSvcClient	Auto	No		..	REST service FB
6	RESTSBf	STRING	Auto	[64]		..	REST Send buffer
7	RxPage	STRING	Auto	[64]		..	Received page (Only for debug)

Esempio ST

```
(* ----- *)
(* ESEGUO INIZIALIZZAZIONI *)
(* ----- *)
(* Eseguo inizializzazione variabili. *)

IF (SysFirstLoop) THEN
  REST.SpyOn:=TRUE;
  REST.RESTSVbck:=ADR(RESTSVbck); (* REST service backup pointer*)
  REST.FIFOFile:='Storage/REST.bin'; (* Path and name of file where to log *)
  REST.FIFOSize:=10000; (* FIFO file size *)

  REST.Host:=ADR('www.slimline.altervista.org'); (* Hostname servizio REST *)
  REST.Page:=ADR('/Mdp095a100/Ptp135a000/RESTSvc.php'); (* Pagina servizio REST *)
  REST.Port:=80; (* Porta servizio REST *)

  REST.HBitTime:=5; (* Heartbeat time (S) *)
  REST.BLength:=512; (* REST Request/Answer buffers length *)
END_IF;

(* ----- *)
(* GESTIONE SERVIZIO REST *)
(* ----- *)
(* Eseguo gestione servizio REST. *)

REST(Enable:=TRUE); (* Eseguo gestione servizio REST *)
RESTSend(RSvID:=REST.RSvID, Add:=NULL);
REST.SvcAck:=FALSE; (* REST service acknowledge *)
IF (REST.Fault) THEN LastError:=SysGetLastError(TRUE); END_IF;

(* ----- *)
(* INVIO INGRESSI DIGITALI AL SERVER CLOUD *)
(* ----- *)
(* Eseguo compattazione ingressi su byte. *)

DInp[0]:=16#00; (* Digital input byte *)
IF (Di00CPU) THEN DInp[0]:=DInp[0] OR 16#01; END_IF;
IF (Di01CPU) THEN DInp[0]:=DInp[0] OR 16#02; END_IF;
IF (Di02CPU) THEN DInp[0]:=DInp[0] OR 16#04; END_IF;
IF (Di03CPU) THEN DInp[0]:=DInp[0] OR 16#08; END_IF;
IF (Di04CPU) THEN DInp[0]:=DInp[0] OR 16#10; END_IF;
IF (Di05CPU) THEN DInp[0]:=DInp[0] OR 16#20; END_IF;

(* Il FB "RESTWSvcClient" invia un messaggio di heartbeat al server ogni *)
(* tempo impostato in "HBitTime". Per informare il server dello stato *)
(* degli ingressi digitali lo invio su ogni variazione. *)

IF (DInp[0] <> DInp[1]) THEN
  DInp[1]:=DInp[0]; (* Digital input byte *)
  i:=SysVsprintf(ADR(RESTSBf), SIZEOF(RESTSBf), 'DInp=%d', USINT_TYPE, ADR(DInp[0]));
  RESTSend(Add:=ADR(RESTSBf));
```

```

END_IF;

(* ----- *)
(* RICEZIONE USCITE DIGITALI DA SERVER CLOUD *)
(* ----- *)
(* Su ricezione Ok da servizio REST controllo l'Ok ricevuto. *)

IF NOT(REST.SvcOk) THEN RETURN; END_IF;
REST.SvcAck:=TRUE; (* REST service acknowledge *)

(* Trasferisco pagina ricevuta in buffer. Solo per debug permette di *)
(* visualizzare il contenuto della pagina ricevuta. *)

i:=TO_INT(Systemmove(ADR(RxPage), REST.PBuffer, SIZEOF(RxPage)));

(* Esego acquisizione valore uscite ricevuto da server. *)

REST.RPAck:=0; (* REST parameters acknowledge *)
IF (SysVarsscanf(SysStrFind(REST.PBuffer, ADR('DOut='), FIND_GET_END), '%d', USINT_TYPE, ADR(DOut))) THEN
REST.RPAck:=REST.RPAck+1; END_IF;

(* Esego gestione uscite digitali. *)

Do00CPU:=TO_BOOL(DOut AND 16#01); //Do00 CPU module
Do01CPU:=TO_BOOL(DOut AND 16#02); //Do01 CPU module
Do02CPU:=TO_BOOL(DOut AND 16#04); //Do02 CPU module
Do03CPU:=TO_BOOL(DOut AND 16#08); //Do03 CPU module

(* [End of file] *)

```

Come si vede viene parametrizzato il FB **REST** con i parametri per la connessione al nostro server cloud di prova.

```

REST.Host:=ADR('www.slimline.altervista.org'); (* Hostname servizio REST *)
REST.Page:=ADR('/Mdp095a100/Ptp135a000/RESTSvc.php'); (* Pagina servizio REST *)
REST.Port:=80; (* Porta servizio REST *)

```

Il FB **REST** invia un messaggio al server cloud ogni tempo definito in **HbitTime**. Su variazione stato ingressi digitali il FB **RESTSend** comanda l'invio al server cloud dello stato degli ingressi.

Ad ogni ricezione di un messaggio REST il server cloud registra i dati ricevuti in un file ini di appoggio ed invia al sistema il valore di comando delle uscite. Riporto di seguito listato del file **RESTSvc.php** a cui si connette il FB REST anche se il file sorgente si trova nel programma dimostrativo.

```

<?php
// *****
// FUNZIONI CONVERSIONE DATI RICEVUTI
// *****
// Funzioni per conversione dati.

function RxBYTE($Rx, $Ofs) {return(intval(substr($Rx, $Ofs, 2), 16));}
function RxWORD($Rx, $Ofs) {return(intval(substr($Rx, $Ofs, 4), 16));}
function RxDWORD($Rx, $Ofs) {return(intval(substr($Rx, $Ofs, 8), 16));}
function RxREAL($Rx, $Ofs) {$Pk=pack("L", intval(substr($Rx, $Ofs, 8), 16)); $Uk=unpack("f", $Pk);
return($Uk[1]);}

// -----
// INCLUSIONE FILES
// -----
// Inclusione files.

$HomeDir=substr($_SERVER['SCRIPT_FILENAME'],0,-strlen($_SERVER['SCRIPT_NAME'])); //Home directory
require_once($HomeDir."/Mdp095a100/Ptp135a000/Include.php"); //Inclusioni generali

// -----
// CONTROLLO RICHIESTA IN ARRIVO
// -----
// La richiesta deve contenere i campi, MID, UID, MV, RP. Se errore esco.

if (!CkReqPars(array("MID", "UID", "MV", "RP"))) exit("Wrong REST parameters");
if (!is_numeric($_REQUEST['UID'])) exit("Wrong system UID");

// Per ogni sistema (Riconoscibile dal suo "UID") esiste un file dati ne eseguo
// lettura e compilazione array globale.

ReadINIFile($_REQUEST['UID']); //File dati di sistema
$GLOBALS['Dt']['PollTime']=GetuTime()-$GLOBALS['Dt']['Heartbeat']; //Tempo poll sistema

```

```

$GLOBALS['Dt']['Heartbeat']=GetuTime(); //Data/Ora ultimo heartbeat (UTC)

// Salvo messaggio ricevuto.

$GLOBALS['Dt']['MV']=$_REQUEST['MV']; //Message version
$GLOBALS['Dt']['RP']=$_REQUEST['RP']; //Received parameters
$GLOBALS['Dt']['RxMessage']="MID={$GLOBALS['Dt']['MID']}, UID={$_REQUEST['UID']}, MV={$GLOBALS['Dt']['MV']},
RP={$GLOBALS['Dt']['RP']}"; //Last request
if (isset($_REQUEST['Data'])) $GLOBALS['Dt']['RxMessage'].=", Data={$_REQUEST['Data']}";

// -----
// CONTROLLO ID MESSAGGIO
// -----
// Controllo se ricevuto l'acknowledge dallo SlimLine del messaggio REST
// inviato precedentemente dal server. Controllo se il MID ricevuto è
// corretto (Successivo al MID del messaggio precedente).

if ((($_REQUEST['MID']-$GLOBALS['Dt']['MID'])&0xFFFF) == 1)
{
    // Ricevuto MID successivo messaggio corretto (Nessun messaggio è
    // andato perso) utilizzo MID ricevuto.

    $GLOBALS['Dt']['MID']=$_REQUEST['MID']; //Message ID
}
else
{
    // Errore ricezione messaggi, occorre eseguire una resincronizzazione
    // sistema, viene inviato un numero random che sarà utilizzato dal
    // sistema come prossimo MID.

    $GLOBALS['Dt']['MID']=rand(0, 65535); //Message ID
    $GLOBALS['Dt']['Resyncs']++; //REST resincronizations
}

// -----
// ACQUISIZIONE INFORMAZIONI DAL MESSAGGIO DATI
// -----
// Un messaggio dati contiene un campo "Data" composto da diversi campi, ogni
// byte occupa due caratteri ascii. I dati sono in Big endian, MSB ... LSB.
// +---+---+---+---+---+---+...+---+
// | Length|0|0| Epoch | Value |
// +---+---+---+---+---+---+...+---+
//
// Length: Lunghezza record (2 byte)
// Epoch: Epoch time (4 byte)
// Value: Stringa con valore (Lunghezza variabile)
// -----
// Se messaggio ricevuto contiene campo "Data" eseguo acquisizione dati campo.

if (!isset($_REQUEST['Data'])) goto SENDDATA;
$GLOBALS['Dt']['Length']=RxWORD($_REQUEST['Data'], 0); //Lunghezza record dati
$GLOBALS['Dt']['Epoch']=RxDWORD($_REQUEST['Data'], 8); //Epoch time relativo al record dati
$GLOBALS['Dt']['Value']=substr($_REQUEST['Data'], 16, ($GLOBALS['Dt']['Length']-8)); //Stringa valore record
dati

// Nel campo "Value" il sistema SlimLine invia le variabili indicandole nel modo
// "Var1=Valore|Var2=Valore|..." controllo che vi sia almeno un "=".
// Nel nostro esempio vi sarà una sola variabile "DInp=xx".

if (strpos($GLOBALS['Dt']['Value'], "=") === false) goto SENDDATA;

// Estraggo nome e valore delle variabili creando un array associativo.
// Mi troverò con $DtArray('DInp' => xx, ...)

$DtArray=array(); //Array associativo dati ricevuti
$Variables=explode("|", $GLOBALS['Dt']['Value']); //Variables array
foreach ($Variables as &$Variable)
{
    if (strpos($Variable, "=") !== false)
    {
        $VNameValue=explode("=", $Variable); //Array Nome/Valore variabile
        $DtArray=array_merge($DtArray, array($VNameValue[0] => $VNameValue[1]));
    }
}

// Appoggio stato ingressi digitali.

$GLOBALS['Dt']['DInp']=$DtArray['DInp']; //Stato ingressi digitali

```

```
// -----
// INVIO DATI AL SISTEMA
// -----
// Eseguo scrittura file per storicizzare i dati sul server.

SENDDATA:
WriteINIFile($_REQUEST['UID']); //Scrittura dille ini

// Inserisco la definizione dei campi da impostare, separo ogni campo con
// lo spazio per permettere nel sistema alla scanf di interrompersi sulla
// acquisizione di valori stringa. Nel nostro esempio vi è un solo campo.

$RetPars=sprintf("DOut=%d ", $GLOBALS['Dt']['DOut']);

// -----
// RITORNO PAGINA AL CLIENT
// -----
// Compilo messaggio di risposta che inizia con il MID. Il valore ritornato
// è calcolato sommando il valore di UID. In questo modo si garantisce che
// il sistema che riceve il messaggio possa verificarlo utilizzando il suo
// unique ID.

$GLOBALS['Dt']['TxMessage']=sprintf("MID=%d", ($GLOBALS['Dt']['MID']+$_REQUEST['UID'])&0xFFFF); //Return page
$GLOBALS['Dt']['TxMessage'].=sprintf("&RP=%d", 0); //Return page
$GLOBALS['Dt']['TxMessage'].=sprintf("&Page=%s", $RetPars); //Return page
echo $GLOBALS['Dt']['TxMessage'];
?>
```

Come si vede dal listato ho volutamente realizzato una gestione molto semplice utilizzando per l'appoggio un file ini, ma in realtà chi ha dimestichezza con applicazioni web troverà molto più efficiente appoggiarsi ad un database.

Come funziona il servizio

Come abbiamo visto sistema SlimLine invia i dati al server cloud (Esegue lo script RESTSvc.php inviando in GET i dati) che in base all'unique ID del sistema controlla se esiste già sul server un file ini dedicato. Se esiste ne esegue lettura in caso contrario viene creato un nuovo file, nel file sono contenuti tutti i dati necessari alla gestione del servizio. Ecco un listato di esempio del file ini.

```
MID="19867"
MV="1.0"
RP="1"
Length="14"
Epoch="1470474898"
Value="DInp=2"
DInp="2"
DOut="1"
RxMessage="MID=19866, UID=10978974, MV=1.0, RP=1"
TxMessage=
Resyncs="1"
PollTime="5.3649678230286"
Heartbeat="1470474724.041"
```

Come si vede è memorizzato il MID cioè l'ID progressivo del messaggio che permette di effettuare il controllo sui messaggi ricevuti. Sono poi memorizzati tutti gli altri campi al solo fine di debug, in questo modo è possibile capire i meccanismi di funzionamento del servizio.

I dati sono aggiornati ad ogni ricezione di messaggio dal sistema (Tempo di heartbeat o su variazione ingressi) e nel campo **PollTime** lo script php calcola il tempo intercorso tra gli aggiornamenti.

Per la visualizzazione dei dati vedere la pagina htm al link <http://www.slimline.altervista.org>. Come si vede accedendo alla pagina avremo qualcosa di simile.



Attenzione utilizzare solo a scopo didattico

Pagina consultazione servizio REST
 Impostando l'unique ID del sistema la pagina visualizza in Ajax i dati ricevuti.

Impostare UID sistema:

MID: 19988
 PollTime: 5.3876609802246
 Heartbeat: 1470475372.1045

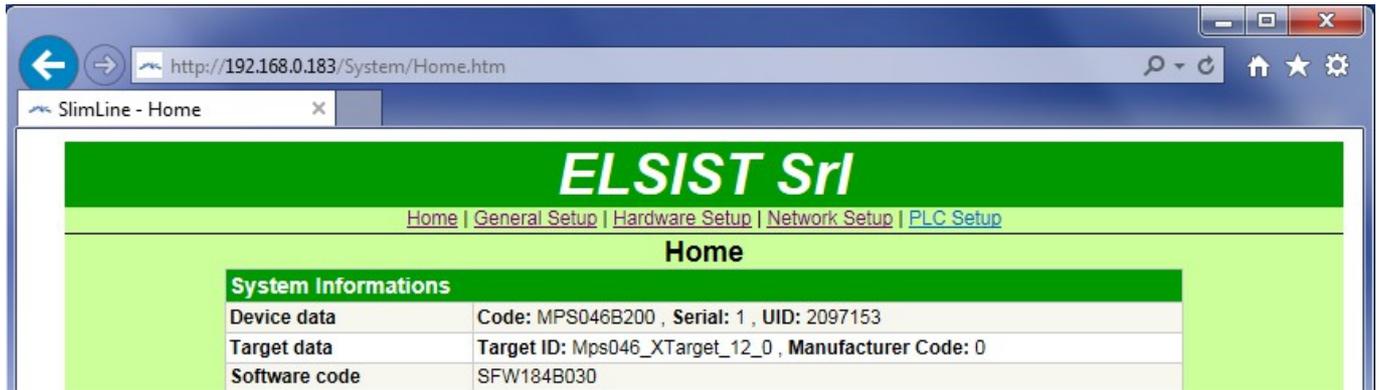
Ingressi digitali:

Uscite digitali:

Come si vede vi è un avvertimento di utilizzare il servizio a solo scopo didattico in quanto sono assenti password di accesso ed altre protezioni. Quindi risulta evidente che inserendo UID di sistema a caso è possibile trovare l'UID del vostro sistema e comandarne le uscite.

Tutto questo è stato fatto volutamente per limitare la complessità del dimostrativo, se si desidera utilizzarlo è certamente possibile partendo dai programmi sorgenti inserire più funzioni e password di accesso.

Ricordo che l'**UniqueID** (UID) del sistema si può visualizzare in debug da LogicLab ma è anche visibile dalla pagina web.



Come dicevo la pagina htm utilizza JQuery ed è aggiornata in tempo reale con chiamate Ajax. Non mi dilungo sulla spiegazione dei TAGs html in quanto sono facilmente intuibili. Quasi interamente la pagina si basa su campi div che sono aggiornati da Javascript ecco il sorgente.

```
// -----
// FUNZIONE ESEGUITA SU LOAD PAGINA
// -----
// Sul load della pagina attivo ajax.

$(document).ready(function()
{
    AjaxCall(); //Eseguo chiamata ajax su load pagina
    setInterval(AjaxCall, 5000); //Imposto chiamata ciclica ajax
});

// -----
// RICHIESTA AJAX
// -----
// Viene eseguita la richiesta ajax. Eseguo lo script "AjaxSvc.php" passando
// i parametri in POST.

function AjaxCall()
{
    // Compongo byte di gestione comando output.

    var DOut=0x00; //Digital output byte command
    if ($("#Do00CPU").is(':checked')) DOut+=0x01;
    if ($("#Do01CPU").is(':checked')) DOut+=0x02;
    if ($("#Do02CPU").is(':checked')) DOut+=0x04;
    if ($("#Do03CPU").is(':checked')) DOut+=0x08;

    // Eseguo invio richiesta ajax con parametri in POST.

    $.ajax(
    {
        type:"POST",
        url:"/Mdp095a100/Ptp135a000/AjaxSvc.php",
        data:"UID="+$("#UID").val()+"&DOut="+DOut,
        dataType:"html",

        // Funzione eseguita su successo della chiamata.

        success:function(Answer)
        {
            // Copio stringa ricevuta da script php per visualizzazione.

            $("#div#Answer").html(urldecode(Answer));

            // Suddivido campi, sono separati dal carattere "|".

            var AArray=Answer.split('|'); //Answer array
            for(var i=0; i<AArray.length; i++)
            {
                // Suddivido i campi tra nome campo e valore. Valorizzo il
                // div di visualizzazione valore.

                var VArray=AArray[i].split('=');
```

```

    $("div#" + VArray[0]).html(encodeURIComponent(VArray[1]));
}

// Eseguo valorizzazione radio button stato ingressi digitali.

$("#Di00CPU").prop("checked", ($("#div#DInp").html() & 0x01 ? true : false));
$("#Di01CPU").prop("checked", ($("#div#DInp").html() & 0x02 ? true : false));
$("#Di02CPU").prop("checked", ($("#div#DInp").html() & 0x04 ? true : false));
$("#Di03CPU").prop("checked", ($("#div#DInp").html() & 0x08 ? true : false));
$("#Di04CPU").prop("checked", ($("#div#DInp").html() & 0x10 ? true : false));
$("#Di05CPU").prop("checked", ($("#div#DInp").html() & 0x20 ? true : false));
},

// Funzione eseguita su errore chiamata.

error: function()
{
    $("div#Answer").html("Call error");
}
});
}

// -----
// DECODIFICA CARATTERI URL
// -----
// Per evitare che nel campo valore vi siano caratteri "=" che possono dare
// fastidio nello split converto campo dati in entit  URL e con questa
// funzione ne eseguo la decodifica.

function urldecode (str) {return decodeURIComponent((str + '').replace(/\+/g, '%20'));}
</script>

```

E' interessante notare che le variabili stringa per evitare il problema di trovarsi caratteri "=" all'interno sono state codificate come richieste URL dallo script php e quindi devono essere decodificate da Javascript.

La richiesta Ajax esegue sul server lo script **AjaxSvc.php** ecco il listato.

```
// -----
// INCLUSIONE FILES
// -----
// Inclusione files.

$HomeDir=substr($_SERVER['SCRIPT_FILENAME'],0,-strlen($_SERVER['SCRIPT_NAME'])); //Rilevo Home directory
require_once($HomeDir."/Mdp095a100/Ptp135a000/Include.php"); //Inclusioni generali

// -----
// CONTROLLO RICHIESTA IN ARRIVO
// -----
// La richiesta deve contenere i campi, UID, DOut. Se errore esco.

if (!CkReqPars(array("UID", "DOut"))) exit("Wrong REST parameters");
if (!is_numeric($_REQUEST['UID'])) exit("Wrong system UID");

// -----
// ESEGUO LETTURA FILE DATI DI SISTEMA
// -----
// Per ogni sistema (Riconoscibile dal suo "UID") esiste un file dati ne eseguo
// lettura e compilazione array globale.

if (!ReadINIFile($_REQUEST['UID'])) exit("System UID not found");

// Salvo byte comando uscite digitali.

$GLOBALS['Dt']['DOut']=$_REQUEST['DOut']; //Comando uscite digitali

// Eseguo scrittura file.

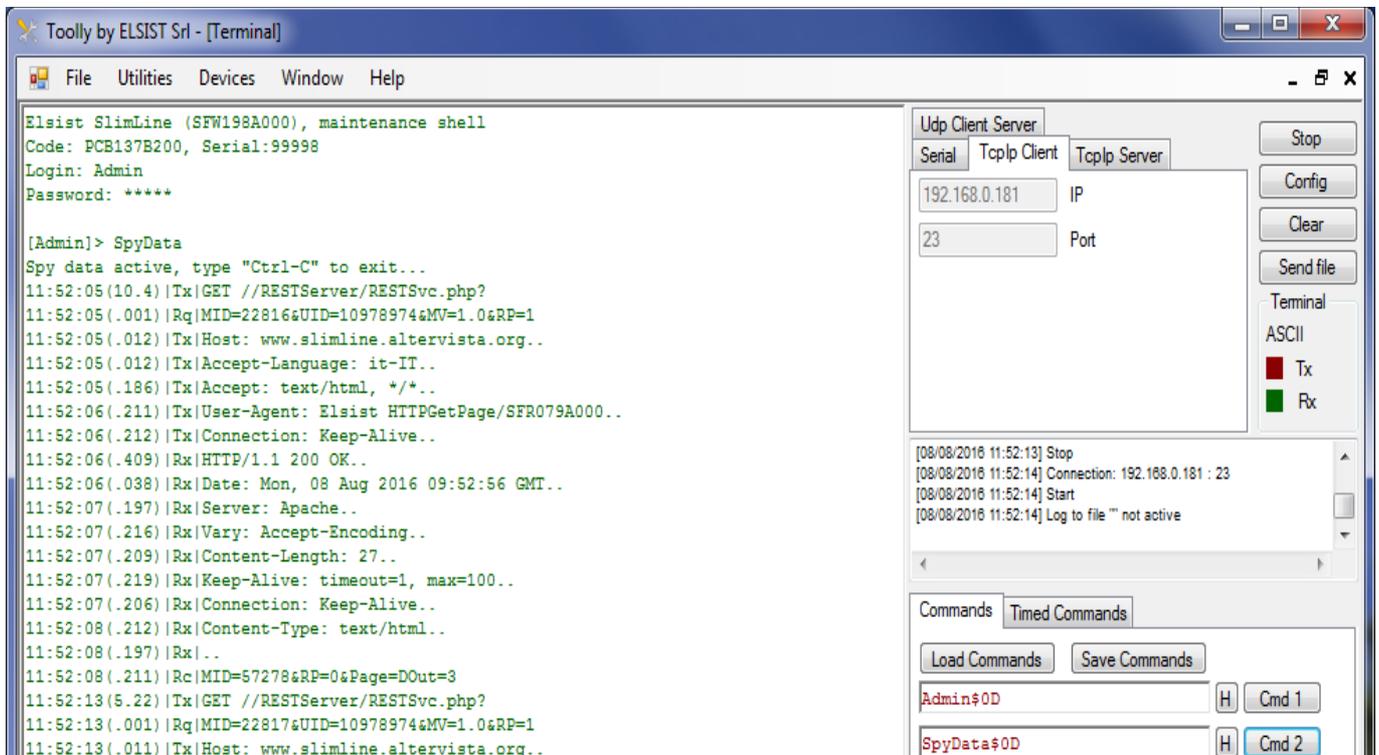
WriteINIFile($_REQUEST['UID']);

// -----
// RITORNO DATI A PAGINA WEB
// -----
// Ritorno dati a pagina web. Per evitare che nel campo valore vi siano
// caratteri "=" che danno fastidio nello split converto stringhe in entità URL.

$return="MID={$GLOBALS['Dt']['MID']}"; //Message ID
$return.="|MV={$GLOBALS['Dt']['MV']}"; //Message version
$return.="|RP={$GLOBALS['Dt']['RP']}"; //Received parameters
$return.="|Length={$GLOBALS['Dt']['Length']}"; //Lunghezza record dati
$return.="|Epoch={$GLOBALS['Dt']['Epoch']}"; //Epoch time relativo al record dati
$return.="|Value=".urlencode($GLOBALS['Dt']['Value']); //Stringa valore record dati
$return.="|DInp={$GLOBALS['Dt']['DInp']}"; //Stato ingressi digitali
$return.="|DOut={$GLOBALS['Dt']['DOut']}"; //Comando uscite digitali
$return.="|RxMessage=".urlencode($GLOBALS['Dt']['RxMessage']); //Messaggio ricevuto
$return.="|TxMessage=".urlencode($GLOBALS['Dt']['TxMessage']); //Messaggio trasmesso
$return.="|Resyncs={$GLOBALS['Dt']['Resyncs']}"; //REST resynchronizations
$return.="|PollTime={$GLOBALS['Dt']['PollTime']}"; //Tempo poll sistema
$return.="|Heartbeat={$GLOBALS['Dt']['Heartbeat']}"; //Data/Ora ultimo heartbeat (UTC)
echo $return;
```

Come testare il dimostrativo

Per testare il dimostrativo basta caricare il programma su di un sistema SlimLine connesso alla rete aziendale ed opportunamente configurato per poter accedere ad Internet (Impostare Gateway e DNS server). Il programma è già configurato con l'URL del servizio su Altervista e con l'indirizzo dello script da eseguire. Come si vede dal sorgente è stato attivato lo Spy del FB **RESTWSvcClient** quindi dalla console di spionaggio è possibile verificare tutti i pacchetti in scambio tra il sistema ed il server cloud. Ecco la schermata di Tolly con l'attivazione della console di spionaggio totale **SpyData<CR>** (Tutti i pacchetti sono spiati).



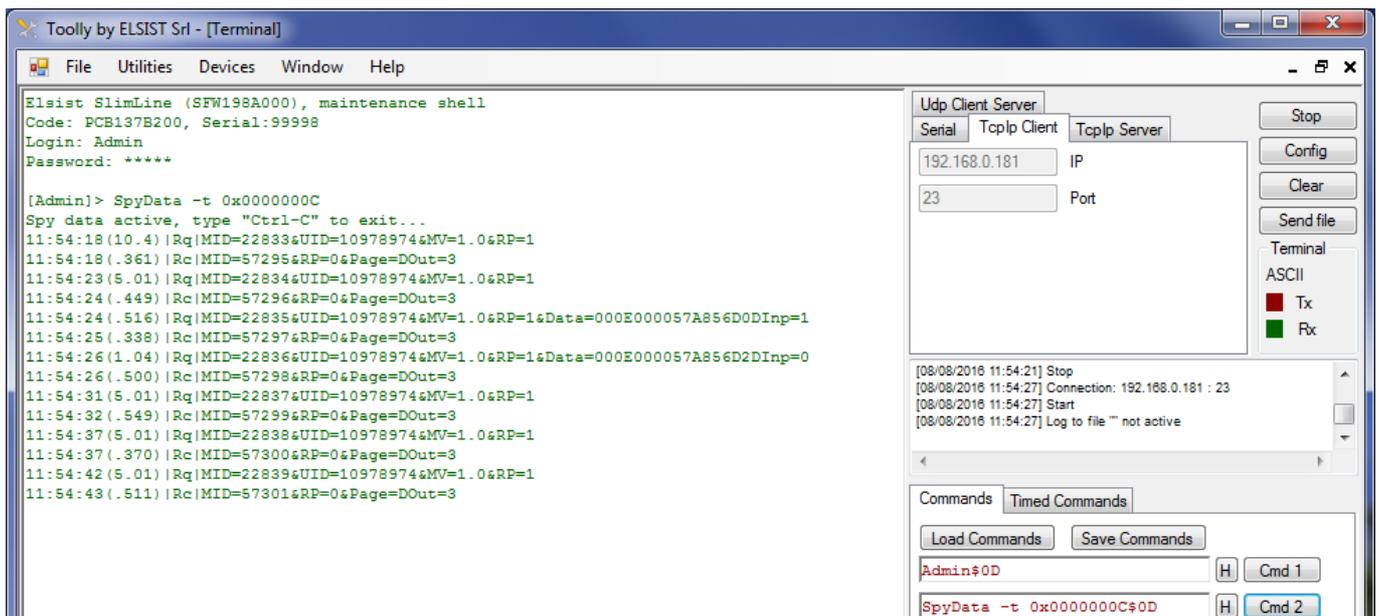
```

Elsist SlimLine (SFW198A000), maintenance shell
Code: PCB137B200, Serial:99998
Login: Admin
Password: *****

[Admin]> SpyData
Spy data active, type "Ctrl-C" to exit...
11:52:05(10.4)|Tx|GET //RESTServer/RESTSvc.php?
11:52:05(.001)|Rq|MID=22816&UID=10978974&MV=1.0&RP=1
11:52:05(.012)|Tx|Host: www.slimline.altervista.org..
11:52:05(.012)|Tx|Accept-Language: it-IT..
11:52:05(.186)|Tx|Accept: text/html, */*..
11:52:06(.211)|Tx|User-Agent: Elsist HTTPGetPage/SFR079A000..
11:52:06(.212)|Tx|Connection: Keep-Alive..
11:52:06(.409)|Rx|HTTP/1.1 200 OK..
11:52:06(.038)|Rx|Date: Mon, 08 Aug 2016 09:52:56 GMT..
11:52:07(.197)|Rx|Server: Apache..
11:52:07(.216)|Rx|Vary: Accept-Encoding..
11:52:07(.209)|Rx|Content-Length: 27..
11:52:07(.219)|Rx|Keep-Alive: timeout=1, max=100..
11:52:07(.206)|Rx|Connection: Keep-Alive..
11:52:08(.212)|Rx|Content-Type: text/html..
11:52:08(.197)|Rx|..
11:52:08(.211)|Rc|MID=57278&RP=0&Page=DOut=3
11:52:13(5.22)|Tx|GET //RESTServer/RESTSvc.php?
11:52:13(.001)|Rq|MID=22817&UID=10978974&MV=1.0&RP=1
11:52:13(.011)|Tx|Host: www.slimline.altervista.org..
    
```

In Tx le stringhe inviate al server ed in Rx quelle ricevute, a fianco di ogni riga vi è il tempo trascorso dalla esecuzione della riga precedente. I dati inviati sono etichettati con Rq in Rc quelli ricevuti.

Attivando il trigger sul comando di spionaggio **SpyData -t 0x0000000C<CR>**, visualizzeremo solo i dati inviati e ricevuti senza visualizzare tutte le stringhe di gestione del protocollo HTTP.



```

Elsist SlimLine (SFW198A000), maintenance shell
Code: PCB137B200, Serial:99998
Login: Admin
Password: *****

[Admin]> SpyData -t 0x0000000C
Spy data active, type "Ctrl-C" to exit...
11:54:18(10.4)|Rq|MID=22833&UID=10978974&MV=1.0&RP=1
11:54:18(.361)|Rc|MID=57295&RP=0&Page=DOut=3
11:54:23(5.01)|Rq|MID=22834&UID=10978974&MV=1.0&RP=1
11:54:24(.449)|Rc|MID=57296&RP=0&Page=DOut=3
11:54:24(.516)|Rq|MID=22835&UID=10978974&MV=1.0&RP=1&Data=000E000057A856D0DInp=1
11:54:25(.338)|Rc|MID=57297&RP=0&Page=DOut=3
11:54:26(1.04)|Rq|MID=22836&UID=10978974&MV=1.0&RP=1&Data=000E000057A856D2DInp=0
11:54:26(.500)|Rc|MID=57298&RP=0&Page=DOut=3
11:54:31(5.01)|Rq|MID=22837&UID=10978974&MV=1.0&RP=1
11:54:32(.549)|Rc|MID=57299&RP=0&Page=DOut=3
11:54:37(5.01)|Rq|MID=22838&UID=10978974&MV=1.0&RP=1
11:54:37(.370)|Rc|MID=57300&RP=0&Page=DOut=3
11:54:42(5.01)|Rq|MID=22839&UID=10978974&MV=1.0&RP=1
11:54:43(.511)|Rc|MID=57301&RP=0&Page=DOut=3
    
```

Dalla analisi vederemo che normalmente ogni 5 secondi viene inviato un pacchetto di heartbeat, ma quando si ha la variazione di un ingresso digitale viene immediatamente inviato un messaggio con il campo **Data** che contiene lo stato degli ingressi.