Davide Bagnoli (DADO91)

WEB STAZIONE METEO CON ARDUINO [7] : BUG FIXING

5 February 2012

Rieccomi, il progetto è terminato, però durante il suo sviluppo, sono saltate all'occhio alcuni problemi o mancanze sia nel firmware, che nel sito e quindi in questo articolo volevo elencarle e spiegare brevemente come sono state risolte.

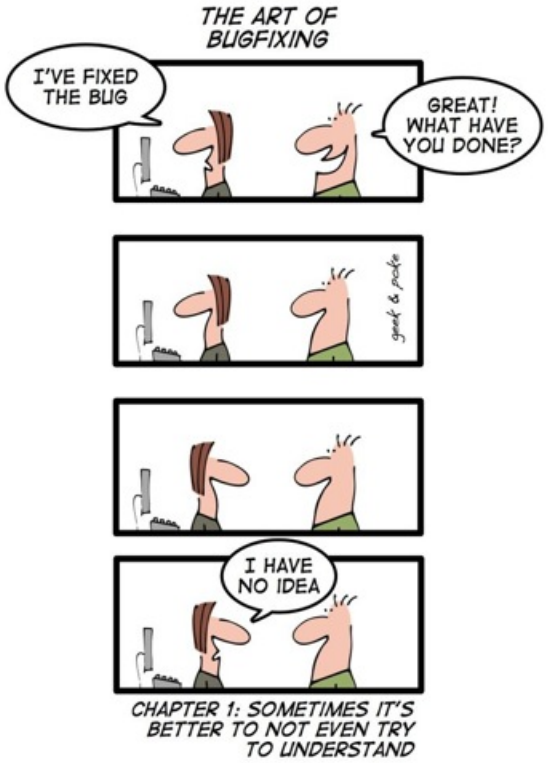
le Sviste e i Problemi

Ecco rapidamente l'elenco:

- Dopo un giorno e mezzo di funzionamento la comunicazione con il sensore si interrompe.
- Monitor 16x1 troppo piccolo, se usato anche per la notifica dei problemi.
- Comunicazione in chiaro, nell'URL, dei dati (tra cui username e password) con il metodo GET.
- Poca flessibilità degli array di char.
- Necessità di stabilire una connessione e richiamare la pagine per la raccolta dei dati per ogni sensore.
- Possibili attacchi SQL Injection senza un opportuno controllo dei parametri ricevuti.

A queste sviste/problemi è già stata data una soluzione. Rimangono alcune cose, perlopiù migliorie, da fare, una su tutte la ricezione di un codice dalla pagina di invio dei dati per notificare tramite arduino possibili problemi del sito, ma ci stiamo lavorando.

Le Soluzioni



bugfixing.jpg

Errore comunicazione con DHT

Dopo aver messo a lavoro il TempDuino a tempo pieno, ho notato che dopo circa un giorno e mezzo, mi notificava l'errore "TimeOut" generato dalla libreria del sensore quando rimane in attesa di un dato, ma questo non arriva. La soluzione è presto detta, invece che dichiarare come globale l'oggetto che gestisce il sensore e quindi allocarlo solo all'avvio del programma, adesso lo alloco in maniera dinamica all'inizio della lettura, per poi essere deallocato a lettura eseguita. Questo accorgimento permette di far stabilire da capo la comunicazione con il sensore ogni volta che viene letto, e per adesso non ha più dato problemi. Ecco il nuovo codice della funzione di raccolta dati :

```
if(time > ReadTime + Periodo_Lettura_Sensore)
 {
   ReadTime = millis();
   dht* DHT = new dht();
   chk = (*DHT).read22(DHT22_PIN);//leggo dal sensore e ritorno un valore corrisponden
   if(chk == 0)
   {
```

```
      t = (*DHT).temperature + temp_adj;
      u = (*DHT).humidity + umid_adj;
      accum_temp += t;
      accum_umid += u;
      n_camp++;
      StampaLCD(t,u);
    }
    else
      {
        StampaErrore(chk);
      }
    free(DHT);
  }
```

Vediamo infatti, all'inizio l'allocazione dinamica dell'oggetto DHT e la deallocazione tramite free(DHT), alla fine del codice. Ovviamente adesso quando usiamo l'oggetto dobbiamo dereferenziarne il puntatore facendo con "*".

Monitor troppo piccolo

A questo si risolve velocemente, ho utilizzato un monitor 16x2, impiegando la prima riga allo stesso modo di prima e la seconda per notificare eventuali errori, il momento dell'invio, e l'ip assegnato a arduino dal DHCP.

Comunicazione in chiaro & poca flessibilità degli array di char

La comunicazione in chiaro è stata risolta usando il metodo POST, invece che il GET, per l'invio delle informazioni. La poca flessibilità degli array di char, è sorta proprio quando sono passato al metodo POST, il motivo è presto detto. Per una richiesta POST, il web server richiede maggiori informazioni riguardo il messaggio inviato, tra cui la lunghezza, cioè il numero di caratteri che compongono la stringa contenente le informazioni, per ottenere questo dato, le alternative erano :

- Crearsi una funzione che lavorasse sugli array di char
- Passare agli oggetti di tipo String, che implementano una funziona apposita per ottenere questo dato (nomestringa.lenght();).

Ovviamente ho optato per la seconda. Una richiesta POST si presenta a questo modo:

```
POST www.miosito.it/pagina.php HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Connection: close
```

```
Content-Length: 25
saluto=ciao&saluto2=salve
```

La stringa dei dati vediamo che ha la stessa forma di quella utilizzata per il GET, però senza il "?" che serviva per delimitare l'url dai dati. La funzione nuova la trovate al prossimo passo, perché necessita di un'altra modifica

Una connessione per ogni invio

Per come era strutturato l'invio delle informazioni in prima battuta, per ogni sensore dovevo connettermi e fare l'opportuna richiesta alla pagina, va da se che la cosa risulta alquanto ridondante e lenta, soprattutto se abbiamo molti sensori. La soluzione adottata è la seguente: Invece che inviare un solo dato alla volta, vengono inviati tutti assieme, aumentando in modo dinamico i parametri della pagina di ricezione. In pratica fornisco alla pagina il numero di lettura che andrò ad inviare e componendo i dati dei sensori a questo modo:

sensore0=IdSensore0&lettura0=LetturaSensore0&sensore1=IdSensore1&lettura1=LetturaSensore1
...
sensoreN-1=IdSensoreN-1&letturaN-1=LetturaSensoreN-1&sensoreN=IdSensoreN&letturaN=Lettur

Ecco il nuovo codice, gli viene passato un vettore con gli ID e uno con le corrispondenti letture.

```
 void InvioHttpSeriale (String server, int porta, String pagina, String username, Strin
{
    String data = "username=" + username + "&password=" + password + "&ndati=" + String
    for(int i=0; i<len ; i++)
      data += "&sensore" + String(i) + "=" + String(id[i]) + "&dato" + String(i) + "="

    String request = "POST http://" + server + ":" + String(porta) + "/" + pagina + " H
    String headerHost = "Host: " + server;

    client.println( request );
    client.println( headerHost );
    client.println( "Content-Type: application/x-www-form-urlencoded" );
    client.println( "Connection: close" );
    client.print( "Content-Length: " );
    client.println( data.length() );
    client.println();
    client.print( data );
    client.println();
}
```

Attacchi SQL Injection

Per capire di cosa si parla vi mando a questo articolo, che spiega molto bene il fenomeno : http://msdn.microsoft.com/it-it/library/cc185099.aspx

In sostanza è il modo di sfruttare la concatenazione delle stringhe per comporre le query, per produrre query totalmente diverse utilizzando dei parametri di un certo formato. Per eliminare questo problema basta filtrare i dati recuperati dagli array superglobali dei dati, che siano $_GET o $_POST o altri, con queste due funzioni :

```
$dato =  mysql_real_escape_string(stripslashes($_POST[‘dato’]));
```

- stripslashes(); Si occupa di togliere gli slash dalle stringhe che gli vengono passate, maggiori informazioni : http://php.net/manual/en/function.stripslashes.php
- mysql_real_escape_string(); Si occupa di convertire i caratteri speciali in caratteri comprensibili a MySQL : http://php.net/manual/en/function.mysql-real-escape-string.php

Usando le due funzioni si riesce a eliminare eventuali stringhe che con una certa formattazione permetterebbero di fare questo genere di attacchi. Qui un piccolo articolo che parla di come prevenire l’SQL Injection : http://php.html.it/guide/lezione/2988/evitare-ogni-tipo-di-sql-injection/

Conclusioni

Ringrazio di nuovo gohan e angus, che mi hanno dato lo spunto e le giuste dritte per la comunicazione via POST e per serializzare il tutto

Breve articolo per farvi sapere le varie modifiche che sono entrate a far parte del progetto finale, comunque lo sviluppo non finisce qui…

Link agli articoli del progetto

Web Stazione Meteo con Arduino [1] - L'idea

Web Stazione Meteo con Arduino [2] - La Rilevazione dei Dati

Web Stazione Meteo con Arduino [3] - Il Primo Circuito di Prova

Web Stazione Meteo con Arduino [4] - Il circuito definitivo e la comunicazione http

Web Stazione Meteo con Arduino [5] - Appendice 1 : Le Reti spiegate a mia Nonna

Web Stazione Meteo con Arduino [6] : Il Sito Web

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Dado91:web-stazione-meteo-con-arduino-7-bug-fixing"