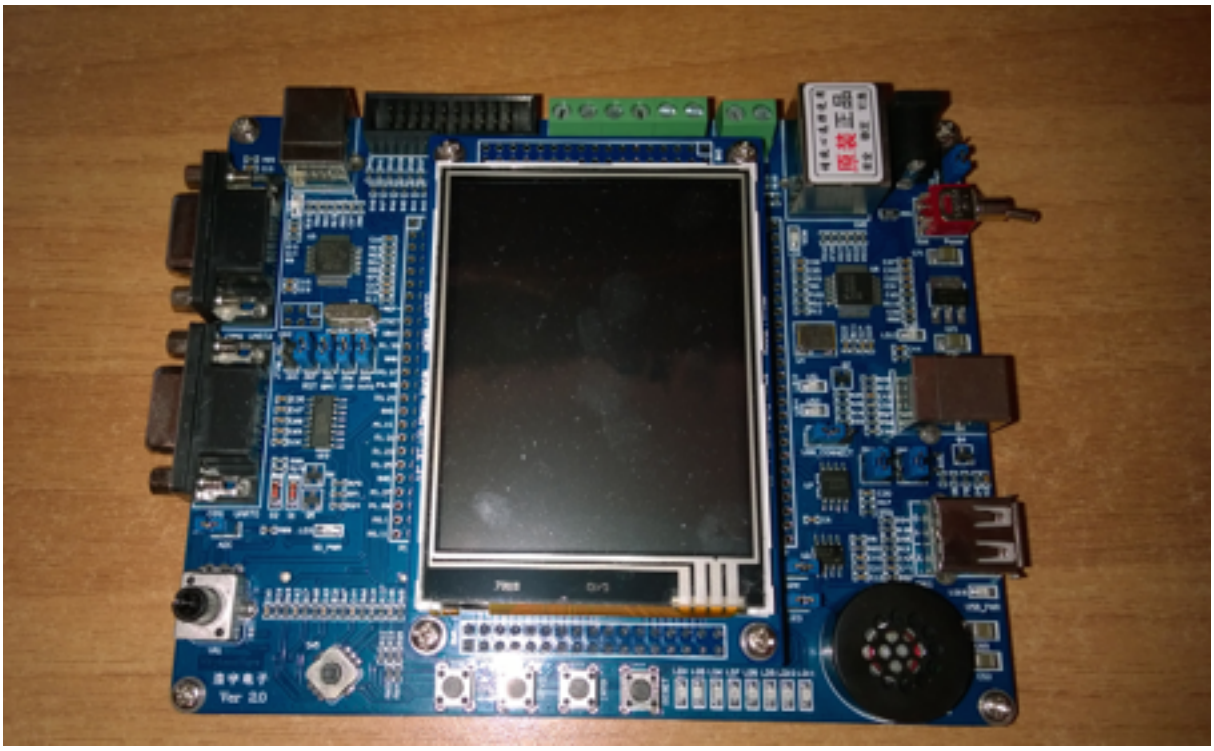
Istituto Statale di Istruzione Secondaria Superiore
“LUCIANO DAL CERO”
ESAME DI STATO 2014

Classe 5 Aa

Candidato

Schiavon Maikol

Programming server oriented embedded



Indice

Motivazioni

Il motivo che mi ha sollecitato ad avventurarmi in questo progetto consiste nel fatto che è sempre stato punto di mio interesse comprendere come avvenisse la comunicazione tra software e hardware o meglio quali conseguenze di carattere hardware possono nascere da un'operazione effettuata da un qualsiasi utente di fronte ad un computer.

Grazie a questo progetto appare evidente come l'interazione di un utente in un pagina web può assumere un aspetto hardware. Con questa frase voglio sottolineare come un'operazione inizialmente di tipo software può essere visivamente e sensibilmente colta.

L'idea al suo inizio era appunto quella di realizzare una applicazione non solamente di natura software ma anche di tipo hardware. Questa tesina vuole dimostrare la stretta relazione tra il software e l'hardware e quali possono essere i limiti qualora venisse a mancare un aspetto dei due.

Con l'obbiettivo di raggiungere questa premessa ho scelto di realizzare un'architettura client/server in questo caso la relazione tra software e hardware è di notevole importanza.

Personalmente risulta affascinate vedere come un' operazione, se vogliamo semplice come l'input generato dal premere un pulsante presente in una pagina web, ha dei risvolti di tipo hardware che si possono notare per esempio con l'accensione di un led.

Altrettanto sorprendente è rendersi conto come una serie di istruzioni in questo caso in linguaggio C, possano assumere un valore fisico che qualunque persona può vedere.

In alcune situazioni siamo portati a pensare che esista un parallelismo tra la comprensione di un sistema e il suo fine. Infatti quando siamo di fronte ad un sistema è probabile che ci chiediamo come sia possibile che esso generi un determinato output.

Personalmente sollecita molto interesse osservare come due sistemi di diversa natura possano comunicare mediante due linguaggi diversi e distinti.

Disarmante è osservare come la scelta di abbracciare un linguaggio di programmazione ci consenta di relazionarci con sistemi privi di voce. Impressionante vedere come per l'ennesima volta il valore che può assumere una lingua. Qualsiasi sia il costo per fare propria una lingua non sarà mai troppo alto rispetto alle capacità che grazie a essa possiamo acquisire, potremmo comunicare con sistemi e persone che ci possono regalare soddisfazioni o esperienze nuove.

Mappa concettuale

Programmazione web

Oggi nel 2014 il web non ha mai avuto un'importanza così rilevante. Si è verificato ciò che Steve Jobs aveva predetto ancora nel lontano 1983. A due giorni dal primo anniversario della morte di Steve Jobs è stato ritrovato l'audio integrale di un suo discorso tenuto nel 1983. Si tratta dell'intervento che Steve Jobs tenne all'International Design Conference di Aspen. L'intervento del visionario confortatore di Apple dura quasi un'ora ed è registrato su una cassetta magnetica intitolata "The future isn't what it used to be, Talk by Steve Jobs".

> "La strategia di Apple è molto semplice. Quello che vogliamo fare è mettere un computer incredibilmente grande in un libro che si possa portare con sé, e imparare a usarlo in 20 minuti. Questo è quello che vogliamo fare e vogliamo farlo in questo decennio. Vogliamo metterci un collegamento radio in modo che non ci si debba connettere a nulla per essere in connessione con tutti i grandi database e gli altri computer. […] Uno di questi giorni, quando avrete computer portatili con radio all'interno, camminerete per Aspen leggendo i messaggi."

In un'epoca ancora dominata da IBM e priva di internet, Steve Jobs si dice fiducioso che il computer possa diventare a breve un apparecchio personale, ma anche un potente mezzo di comunicazione. Così è stato ed ora qualsiasi attività lavorativa si sta attivando per essere conosciuta ma sopratutto presente nel web. E' inevitabile per cui lo sviluppo di siti web.

Ma in realtà questa è solo una delle possibili utilizzi del web, infatti sono nati diversi approcci alla rete o meglio ad internet che a sua volta hanno contribuito alla realizzazioni di differenti applicazioni.

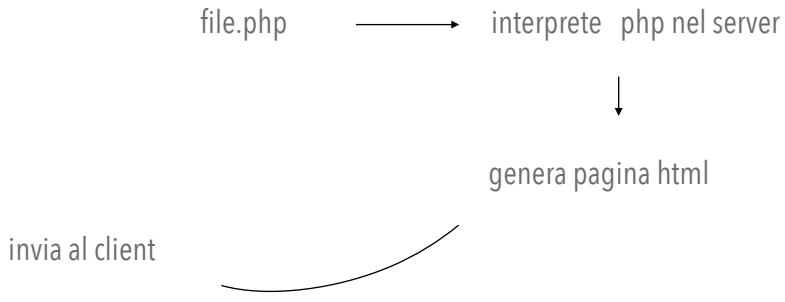
Page oriented

L'esempio riportato qui sopra è un chiaro riferimento alla programmazione page oriented in cui il linguaggio a padroneggiare è il PHP.

In particolare il sito web è una correlazione di pagine con estensione .php formate da una parte che cura la parte grafica della pagina realizzata in linguaggio html e le interrogazioni al database.

Il linguaggio html da solo non sempre soddisfa le varie richieste che una pagina web può soddisfare per questo motivo si ricorre a linguaggi di scripting lato server, in questo caso PHP. Un linguaggio interpretato dal server che rende dinamiche le pagine web.

Possiamo rappresentare la situazioni con questo schema:



Durante l'anno scolastico 2013-2014 ho avuto modo di sviluppare alcuni esempi di un approccio page oriented che si possono raggiungere al corrispettivo indirizzo web:

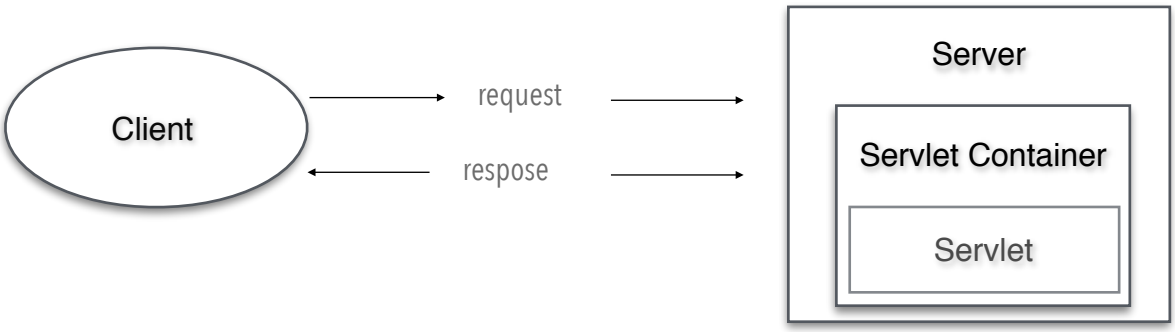
1. www.progettimaikol.altervista.org/VideotecaW1
2. www.carpediemm.altervista.org/Pro

Server oriented

Per la realizzazione di un sito web è possibile avere un diverso approccio al progetto ovvero server oriented ricorrendo all'utilizzo delle servlet.

Una Servlet è un componente software scritto in Java, gestito da un "container", che produce contenuto web dinamico.

Una Servlet interagisce con un web Client attraverso il paradigma di comunicazione request/response.



Il servlet container (o servlet engine) è un'estensione di un web server che fornisce l'ambiente di esecuzione ad una Servlet.

Il funzionamento consiste in una interazione tra client e servlet e avviene secondo questi punti:

1. Il Client fa una richiesta HTTP al web server

2. Il web server carica la servlet (solo la prima volta) e crea un thread per eseguirla

3. Il container esegue la servlet richiesta

4. La servlet genera la risposta

5. La risposta viene restituita al client

Quando si parla del linguaggio Java sono subito evidenti quali possono essere i vantaggi, per cui le servlet offrono:

1. Velocità
2. Persistenti
   - una volta caricata, una servlet rimane in memoria e può ottimizzare l'accesso alle risorse attraverso cachino
3. Implementation independence
   - usano una API standard supportata da molti web server
4. Vantaggi offerti dal linguaggio Java
   - Platform independence, Object Oriented programming, Garbage Collection, …

Attraverso l'utilizzo delle Servlet API che sono un framework di classi Java che offrono delle interfacce object oriented possiamo migliorare la comunicazione tra client e server a tale punto che possiamo interagire con i registri associati alle periferiche hardware.

Sistemi Embedded

Ho cercato di trovare un differente approccio alla rete. Partendo dalle conoscente acquisite durante gli anni di studio ho deciso di approfondire un settore a me poco conosciuto, ma di grande interesse industriale: i sistemi embedded, ovvero tutti quei sistemi elettronici di elaborazione a microprocessore che trovano una ampia applicazione pratica anche nella nostra quotidianità.

Contrariamente ai sistemi generici, quando si deve sviluppare un sistema embedded si conosce i suoi compiti a priori, che eseguirà dunque grazie ad una combinazione hardware/software specificamente studiata per tale applicazione.

Esistono svariati esempi di sistemi embedded, occorre solamente a pensare al settore automobilistico. Una moderna vettura di fascia media presenta mediamente un centinaio di microcontrollori, ognuno dei quali svolge dei compiti ben precisi, per citarne alcuni: abs, airbag.

Con tale sistemi dobbiamo abbandonare l'idea di utilizzare il linguaggio Java e abbracciare il linguaggio C.

Il vantaggio che potremmo cogliere immediatamente è un diretto controllo dell'hardware.

L'applicazione che svilupperò grazie a un sistema embedded è simulare in tutti gli effetti una vera e propria architettura Client / Server.

Client / Server

L'evoluzione esponenziale dell'elettronica e dei sistemi informatici hanno consentito all'hardware di subire un processo di evaporazione e di imprevedibile sublimazione a tale punto che oggi è diventato tutto ormai sorprendentemente piccolo ed esteticamente curato.

Grazie a tale fenomeno è stato possibile sviluppare questo progetto. L' obbiettivo è appunto simulare un server mediate un sistema embedded.

Dove è presente un server sicuramente possiamo trovare almeno un client, perciò svilupperemo nel nostro piccolo un sistema client e server. Questa architettura si caratterizza per il fatto che è formata da un minimo di due tipi di moduli: il client e il server, che generalmente sono due macchine distinte ma entrambe collegate alla rete.

Il server in realtà è un punto di accesso logico a funzionalità fornite da un insieme di moduli. Mentre un generico client è basato su un'applicazione, il browser, che deve supportare un insieme minimo di funzionalità, tra cui l'interpretazione del codice che si nasconde dietro ad ogni pagina web.

Nell'applicazione che presento la scheda elettronica assume il ruolo di server mentre il computer è il nostro client.

Ora che abbiamo capito il ruolo ricoperto dai due moduli, dobbiamo rispondere ad un altro interrogativo: chi fa cosa?

Il server svolge le operazioni necessarie per realizzare un servizio, ad esempio gestisce una banca dati, l'aggiornamento dei dati e la loro integrità.

Tipicamente la funzione del client è quella di interrogare il servizio, verificando i dati inseriti e provvedere ad inviare al server le richieste formulate dall'utente.

Se desideriamo che l'architettura funzioni correttamente, il server e il client devono eseguire delle funzioni. In particolare il ruolo del client consiste nel:

- Abilitare l'utente a spedire una richiesta al server
- Rendere comprensibile la richiesta al server
- Formattare la risposta del server in modo che l'utente possa leggerla

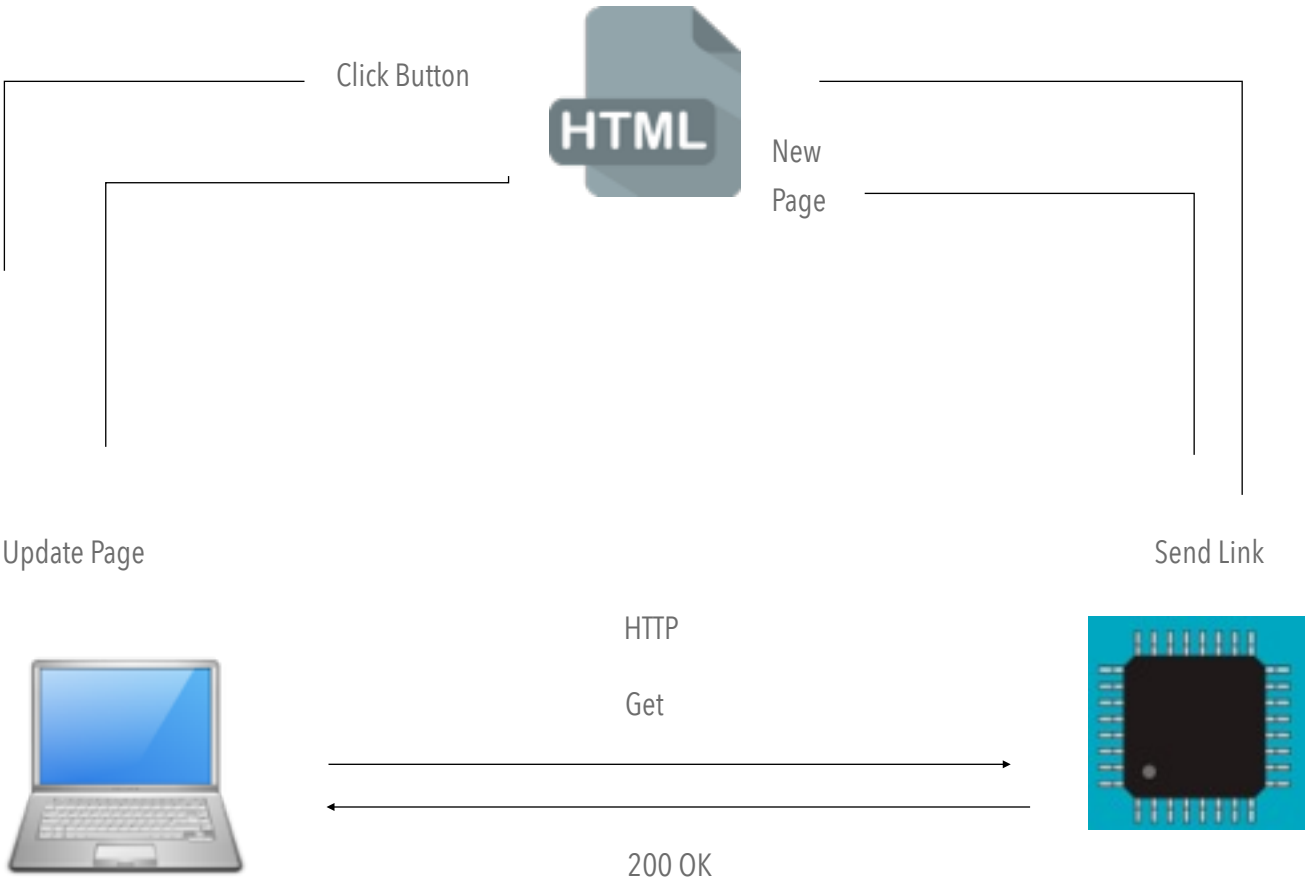
Il server deve a sua volta:

- Ricevere una richiesta da un client e processare la risposta
- Rispondere, spedendo l'informazione richiesta al client

I due moduli devono comunicare tra di solo e per farlo si rivolgono al protocollo HTTP, HyperTextTrasferProtocol, il cui compito principale consiste nella trasmissione delle informazioni sul web.

Un protocollo è un insieme di regole che permettono di trovare uno standard di comunicazione tra diversi computer attraverso la rete, dove per rete si intende un insieme di due o più computer connessi tra di loro ed in grado di condividere informazioni. Quando due o più computer comunicano tra di loro si scambiano una serie di informazioni. Per potersi scambiare informazioni, i vari computer devono avere dei protocolli che permettano di attribuire ad un determinato comando un significato univoco per tutte le macchine.

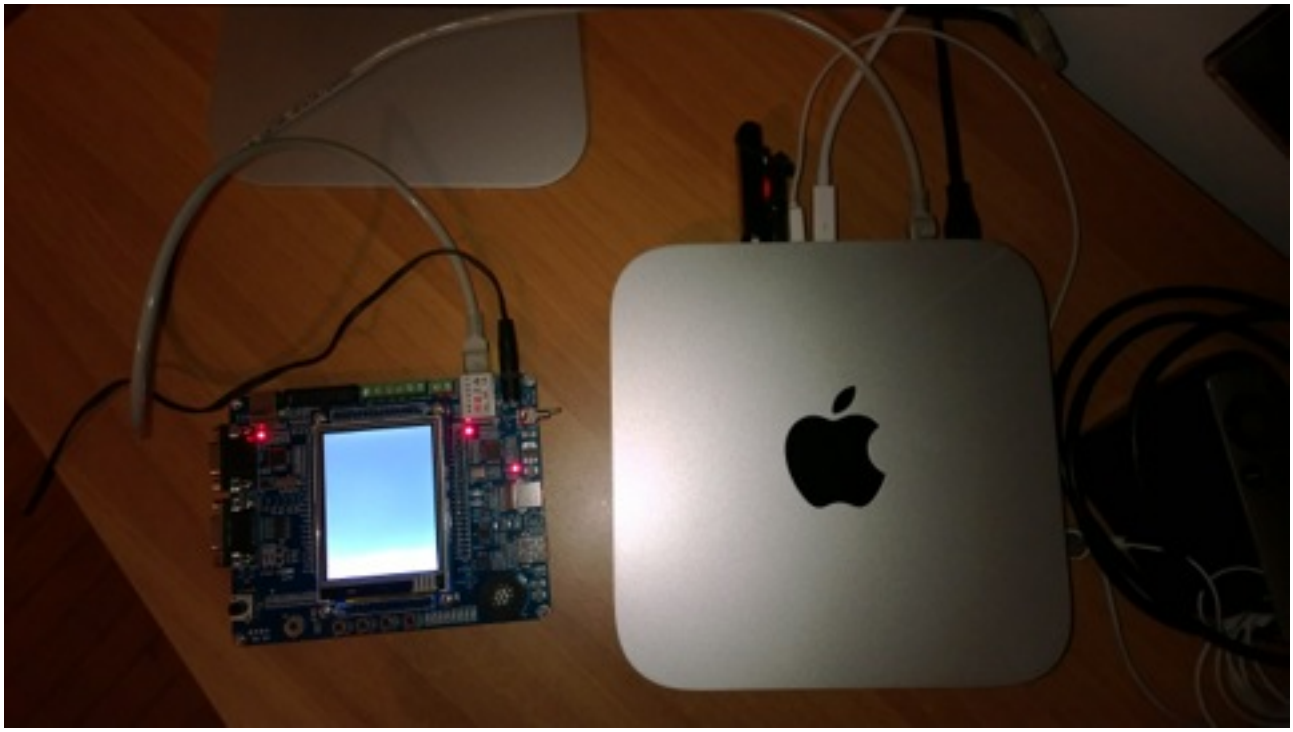
Lo schema di funzionamento del progetto proposto è il seguente:



Dopo ogni buona illustrazione segue un'esauriente dimostrazione, rispettando questa regola non scritta possiamo ora avventurarci al seguente punto.

Comunicazione

Per rendere ancora più interessante il progetto si è scelto che la connessione tra la scheda e il client avvenisse attraverso un cavo UTP nello standard ethernet.



Questa soluzione consente alla scheda di adottare una caratteristica fondamentale e di notevole importanza, che comunemente chiamiamo flessibilità. Un aspetto che giorno d'oggi risulta essere un tema di studio e di sviluppo per molti collosi dell'informatica.

Il sistema che stiamo sviluppando di qualsiasi natura sia per raggiungere il successo deve essere a disposizione di tutti per questo motivo deve possedere il minore numero di requisiti.

Con questa premessa appunto ho scelto che la comunicazione avvenisse attraverso un cavo UTP nello standard ethernet, ma non nascondo il fatto che questa decisione è stata intrapresa anche per consentire al mio progetto di avere notevoli future applicazioni.

Infatti al client non verrà richiesto di disporre di alcun'altra risorsa al di fuori di quelle che possiede di natura propria:

I. un comune browser

II. un'interfaccia ethernet

L'unica operazione richiesta al client consiste nell' indicare nella barra del url del proprio browser l'indirizzo IP della scheda, dopo però aver eseguito la correttamente la configurazione di rete.

Configurazione di rete

Se vogliamo che il nostro sistema funzioni correttamente dobbiamo preoccuparci che il client e il server si trovano nello stesso canale di comunicazione, in linguaggio tecnico, i due moduli devono appartenere alla medesima rete.

Per questo motivo il client e il server devono presentare questa configurazione di rete:

	Client	Server
Indirizzo IP	192.168.1.122	192.168.1.123
Subnet Mask	255.255.255.0	255.255.255.0

In una configurazione standard se vogliamo che la nostra stazione di lavoro raggiunga internet, per esempio il motore di ricerca www.google.it, è necessario che indichiamo anche il gateway.

Con il termine generico gateway si indica il servizio di inoltro dei pacchetti verso l'esterno. In questa topologia di rete il client e il server sono due peer che comunicano tra di loro attraverso un singolo cavo UTP, per questo motivo possiamo affermare che il gateway non è un parametro che ci interessa ai fini di questo progetto.

Wireshark

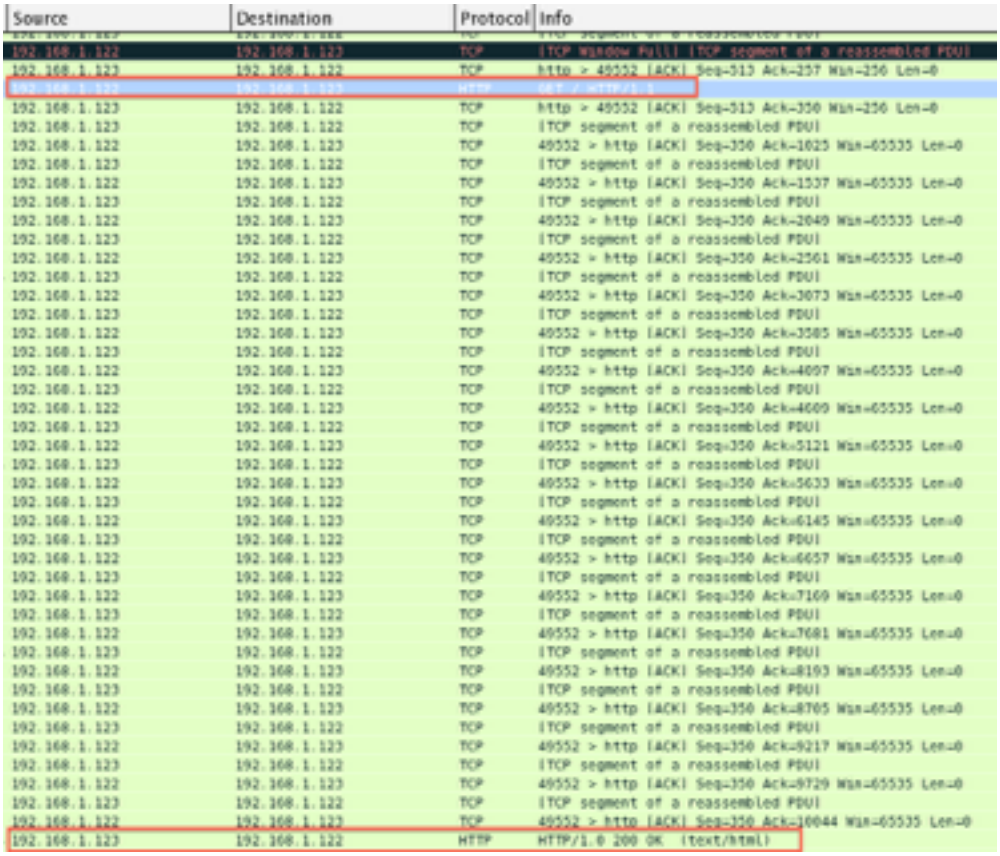
In questo punto si vuole proporre una rappresentazione del funzionamento di questo sistema sotto un punto di vista più tecnico, analizzando la comunicazione tra client e server. Con questa premessa e per rendere la comprensibile la dimostrazione a tutti ho scelto di ricorrere all'utilizzo del software Wireshark.

Wireshark è un eccellente analizzatore di protocollo o "packet sniffer" in grado di esaminare il contenuto di tutti i pacchetti in transito sull'interfaccia di rete attiva. La prerogativa di questo programma opensource, consiste nel fornire una panoramica dettagliata di tutto ciò che sta accadendo sulla rete locale proponendo un'interfaccia grafica di semplice utilizzo e di immediata comprensione.

Risulta evidente che la comunicazione tra client e il server avviene attraverso il protocollo HTTP. In particolare possiamo vedere che ad ogni operazione eseguita dal client due sono le funzioni che si ripetono ad ogni inizio e ad ogni fine.

La prima funzione risulta essere il metodo GET, il quale mi consente di inviare una differente variabile, in stretta relazione alla richiesta del client, al server, o meglio alla scheda che a sua volta elaborerà la stringa per restituire una corretta risposta.

La seconda funzione è il messaggio "200 OK" che viene inviato al client per informarlo che il server ha fornito correttamente il contenuto nella sezione body.



Source	Destination	Protocol	Info
192.168.1.122	192.168.1.123	TCP	[TCP Window Full] [TCP segment of a reassembled PDU]
192.168.1.123	192.168.1.122	TCP	http > 49552 [ACK] Seq=513 Ack=257 Win=256 Len=0
192.168.1.122	192.168.1.123	HTTP	GET / HTTP/1.1
192.168.1.123	192.168.1.122	TCP	http > 49552 [ACK] Seq=513 Ack=350 Win=256 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=1025 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=1537 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=2049 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=2561 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=3073 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=3585 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=4097 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=4609 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=5121 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=5633 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=6145 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=6657 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=7169 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=7681 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=8193 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=8705 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=9217 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=9729 Win=65535 Len=0
192.168.1.123	192.168.1.122	TCP	[TCP segment of a reassembled PDU]
192.168.1.122	192.168.1.123	TCP	49552 > http [ACK] Seq=350 Ack=10044 Win=65535 Len=0
192.168.1.123	192.168.1.122	HTTP	HTTP/1.0 200 OK (text/html)

Come abbiamo riportato prima, la comunicazione avviene attraverso il protocollo HTTP che si preoccupa solo del trasferimento delle informazioni tra il mittente e il destinatario, senza porre in relazione i dati tra le sessioni precedenti e successive. Questo in termini pratici significa minore quantità di dati da trasmettere e dunque maggior velocità ma nel medesimo momento non si gestiscono i dati inseriti con la dovuta riservatezza.

Di conseguenza attraverso l'utilizzo di Wireshark possiamo vedere con la massima trasparenza il valore prelevato dal metodo GET.

Nella situazione in cui viene richiesto all'utente di inserire la password, è possibile conoscere il valore inserito:

Source	Destination	Protocol	Info
192.168.1.123	192.168.1.122	TCP	http > 49553 [ACK] Seq=513 Ack=257 Win=256 Len=0
192.168.1.122	192.168.1.123	HTTP	GET /?pw=ciao HTTP/1.1

Anche se di minore importanza, quando l'utente difronte al pannello di controllo sceglierà con quale led interagire se noi ci troviamo all'interno della rete del client e del server possiamo conoscere quale led ha cambiato stato.

N	Source	Destination	Protocol	Info
576	192.168.1.123	192.168.1.122	TCP	http > 49565 [ACK] Seq=513 Ack=257 Win=256 Len=0
577	192.168.1.122	192.168.1.123	HTTP	GET /led?led=1 HTTP/1.1

N	Source	Destination	Protocol	Info
626	192.168.1.123	192.168.1.122	TCP	http > 49566 [ACK] Seq=513 Ack=257 Win=256 Len=0
627	192.168.1.122	192.168.1.123	HTTP	GET /led?led=1 HTTP/1.1

In questo caso non possiamo sapere se il led è stato acceso o spento perché l'informazione inviata dal client al server in entrambe le situazione risulta essere uguale e indica solamente il numero del led scelto dall'utente.

Client

Abilitare l'utente a spedire una richiesta al server

Il presupposto che dobbiamo prefissarci quando dobbiamo offrire un servizio o realizzare un sistema deve essere essenzialmente che tale sistema o servizio deve essere semplice, o meglio le operazioni richieste per raggiungere il risultato desiderato devono essere intuitive. Non dimentichiamo che il nostro sistema o servizio deve risultare accessibile a tutti gli utenti ai quali non deve essere richiesto di possedere rigorosamente delle precedenti conoscenze per utilizzare il sistema.

Nel nostro caso il mezzo migliore per fornire un'interfaccia semplice nella navigazione tra i vari elementi è proprio quella di sviluppare una pagina web.

In questa circostanza dobbiamo ricorrere all'utilizzo dei linguaggi **html e css**, conosciuti anche come markup languages.

HTML is a high lavel language which includes a series of rules and instructions for the description and construction of web pages. It know as markup languages because use instructions, known as markup tags. In particular Html stands for, **h**yper**T**ext**m**arkup **l**anguage, consists of commands called tags. Tags use to tell the web browser how to display text or pictures.

There are a number of different elements that you can use on web page. Is possibility you see the code behind in web pages.

Today we are experiencing the rapid success of many mobile devices, for this reason is very importance develop a web page compatible with all the different display sizes.

The web pages display in one resolution with height 100% and width 100%, because the page can be display with no problem in another screen.

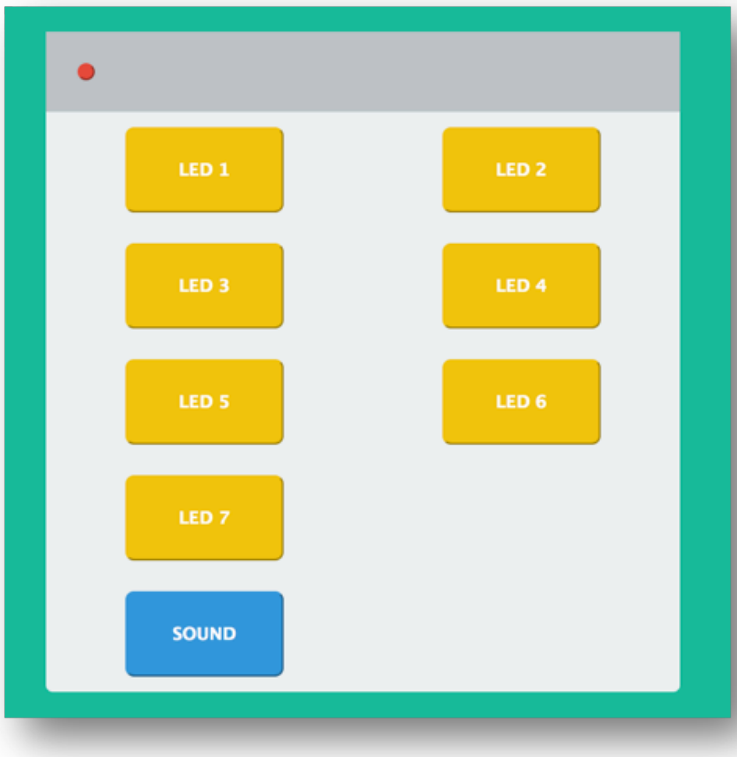
Each aspect graphic you can see that is linked to a css rules .Below I have listed some of the html code of the first web page. The tag input indicate the space where the client insert the password for enter the control panel. I the tag button have connect it one rule, it name is button-invia.

```
<input type="text" name="pw" placeholder="password">
<button type="submit" id="button-invia">Invia</button>
```

In CSS exist one rule, its name is button-invia, where shows the rule of its graphics aspect. In the second part of this code add to the button another color when you with pointer go over in the button.

```
#button-invia{
border-color:#1ABC9C;
background:#1ABC9C;
}
#button-invia:hover{
background-color:#2fe2bf;
border-color:#2fe2bf;}
```

In the second page is possibility to see the control pannel, where you can power led or reproduced the sound. Both the web pages have been developed for be compatibily all display, for this reason all graphics elements have width and height in percentage.



All bottons are in one container, its name is space:

```
.spazio{
width:50%;
height:50%;
text-align:center;}
```

All botton follow one rule, its name is led:

```
#led{
width:50%;
height:100px;
background-color:#f1c40f;
text-align:center;
font-size:20px;
border-color:#f1c40f;}
```

The code of these web pages is very simply and effective. All code was study to occupy the least possible space, all elements have been optimized.

Rendere comprensibile la richiesta al server

La richiesta del client al server avviene attraverso il protocollo HTTP.

Il protocollo HTTP si basa sul modello domanda/risposta. La comunicazione ha sempre inizio dal client che si collega ad un server per formulare una richiesta. Il server a sua volta invia la risposta e la transazione è conclusa.

In questo progetto la richiesta al server generata dal client in un primo momento consiste in una stringa la quale attribuiamo l'importanza di una password.

Dal punto di vista del client la richiesta viene formulata riportando la password digitata dall'utente nell'indirizzo url della pagina web.

Per eseguire questa operazione oltre alla parte grafica che abbiamo analizzato prima, dobbiamo elaborare un corretto funzionamento della pagina sottostando alle dure limitazioni poste dal linguaggio HTML.

Risulta essenziale quindi realizzare un evento: quando l'utente preme il pulsante dopo aver inserito la password, la stringa digitata quasi magicamente viene visualizzata nell' indirizzo url.

Tradotto in linguaggio HTML:

```
<form action="" method="get">
    <input type="text" name="pw" placeholder="password">
<button type="submit" id="button-invia">Invia</button>
</form>
```

Ad riportare la password nell'indirizzo è il metodo GET che consente di inviare dati usando appunto il protocollo HTTP. Secondo la specifica del protocollo HTTP di questo metodo i dati sono preceduti dall'indirizzo della pagina richiesta e un punto interrogativo.

Questo progetto ha solo un compito illustrativo per cui la riservatezza della password inserita sarà un criterio che verrà sviluppato successivamente, introducendo una crittografia e un'autenticazione attraverso il protocollo HTTPS.

Il risultato che otterremo una volta che l'utente effettuerà l'opportuno evento risulterà questo:

http://192.168.1.123/?pw=Gianni93

Possiamo notare che il protocollo utilizzato è proprio HTTP, 192.168.1.123 è l'indirizzo della scheda che interroghiamo, subito dopo viene riportato il risultato generato dal metodo GET ovvero '/?pw=Gianni93' in particolare Gianni93 è la password corretta che è stata inserita nel input della pagina html.

L'utente una volt ache ha eseguito il login, viene indirizzato alla pagina web del pannello di controllo dove può accendere o spegnere un led o avviare la riproduzione di un suono.

Per inviare la richiesta al server utilizziamo il medesimo metodo esaminato qui sopra ovvero il GET attribuendo ad ogni bottone un valore il quale indica quale led deve essere acceso o spento.

```
<form action=led method="get" name=led>
     <button type="submit" name="led" value=1 id="led">LED 1</button>
</form>
```

Il risultato che otterremo sarà questo:

http://192.168.1.123/led?led=6

Dove appunto il valore 6 indica su quale led la scheda deve eseguire l'operazione di accensione o spegnimento.

Formattare la risposta del server in modo che l'utente possa leggerla

Unfortunately for us, computer can't understand spoken English or any natural language. The only language they can understand directly is machine code, which consists of 1s and 0s, binary code.

Il problema che si deve appunto risolvere è quello di trovare una soluzione che consente al client e al server di avviare un dialogo che entrambi comprendono.

In un primo momento la soluzione sembra non esistere ma in realtà se sono qui a parlarvi del mio progetto probabilmente un ignoto metodo deve esistere.

L'idea sviluppata consiste nel prendere tutto il codice html e css che formano la pagina web e inserirlo in un contenitore comprensibile al server.

Ricordando che il nostro server elabora la richiesta e restituisce una risposta grazie al linguaggio C, il contenitore che dovremmo utilizzare si chiama array.

In questo modo quando avviene un fenomeno viene restituito un apposito array al client o altrimenti viene inviato al client un altro array.

In particolare il server dispone di due array, che per il client sono due pagine web. L'utente ai fini pratici deve inserire la corretta password all'interno dell'input html della pagina web. Quando avviene questo fenomeno il server comprende che il client gli pone una richiesta controllando la lunghezza del indirizzo della pagina.

Se la password inserita è corretta il server restituirà un'array che agli occhi dell'utente finale risulta essere il pannello di controllo mentre se la password digitata è errata l'utente si troverà nuovamente alla pagina di accesso.

La soluzione adottata è un perfetto modo per rendere compatibile ma soprattutto comprensibile il linguaggio Html e css al nostro programma in C.

Server

Il secondo modulo della architettura è indubbiamente il server che viene interrogato dal client e di conseguenza deve eseguire le corrette operazioni per restituire la corretta risposta. In un primo momento quando l'utente si trova ancora nella pagina del login la scheda deve controllare che la password digitata sia corretta e in tale situazione deve restituire o meno al client la pagina del pannello di controllo (1), in un secondo istante una volta che l'utente è riuscito ad raggiungere il pannello di controllo deve riuscire ad interagire con la scheda stessa(2).

Ho scelto di enumerare questi due step e di analizzarli come è stato fatto con il client analizzando però in questo caso le principali funzionalità del server.

Ricevere una richiesta da un client e processare la risposta

1
password

La richiesta inoltrata dal client al server risulta essere una stringa che ha la funzione di password che viene riportata nel url del browser.

La password corretta è presente all'interno del programma C della nostra scheda, quello che realmente avviene è un confronto tra le due stringhe.

Per avvicinarsi alla complessità del progetto, illustro l'algoritmo con cui la scheda eseguirà questo controllo.

Desk code:

Inizializzo due interi che mi aiutano a scorrere il nostro array multidimensionale.

```
int PW_search()
{

int i = 0;
int j = 0;
```

inizializzo due char array, al primo associo una grandezza pari alla lunghezza della password memorizzata nella scheda, mentre al secondo gli attribuisco la password memorizzata.

```
char pw_buffer[PW_LENGTH -1];
char pw[] = PW;
```

indirizzo che il server ricevere è il medesimo riportato nell'url per cui cerco in carattere '?'

```
while(TCP_RX_BUF[i] != '?')
      i++;
```

ora posso prelevare la password inserite dal nostro utente, per riuscire in questo intento dobbiamo scorrere l'intero indirizzo fino quando troviamo 'pw='.

```
if(TCP_RX_BUF[++i] == 'p' && TCP_RX_BUF[++i] == 'w' && TCP_RX_BUF[++i] == '=')
      {
```

Solo ora posso conoscere la password inserita. In particolare copio all'interno del buffer pw_buffer tutto ciò che si trova dopo i 'pw=' del buffer TCP_RX_BUF, fino alla lunghezza della nostra password (PW_LENGHT-1)

```
strncpy(pw_buffer, (char *)&TCP_RX_BUF[i], PW_LENGTH-1);
```

confronto carattere per carattere la password ricevuto con quella prestabilita

```
for(j = 0;j < PW_LENGTH -1; j++)
                  if(pw_buffer[j] != pw[j])
                        return 0;

            return 1;
      }

      return 0;
}
```

2
led

Una volta eseguito correttamente il login, l'utente si troverà di fronte al pannello di controllo. Ora quando si preme il pulsante ci si aspetta che avvenga qualche cosa.

In particolare quando l'utente preme il pulsante LED1 si dovrà accendere il LED numero 1 mentre se preme per una seconda volta il bottone LED1 il led si spegne, lo stesso fenomeno si verificherà con gli altri rimanenti sette led.

Per la gestione dei led utilizzeremo un array con una grandezza pari a 7 e lo inizializzeremo tutto a 0. Ogni bit rappresenterà un nostro led fisico.

Esaminiamo l'algoritmo che ci consente di eseguire questa operazione:

```
int led_flag[7] = {0};

for(i = 0; i < 7; i++)

      led_flag[i] = 0;
```

LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
0	0	0	0	0	0	0	0

Il valore 0 significa che il led è spento mentre 1 il led è accesso, ora risulta intuitivo che se vogliamo accendere il LED3 dobbiamo portare ad 1 il rispettivo bit. In pratica dobbiamo ottenere questo risultato:

LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
0	0	1	0	0	0	0	0

Dopo questa deduzione vediamo come fare diventare realtà questa ipotesi:

```
void LED_On (unsigned int num) {

      LPC_GPIO2->FIOPIN |= led_mask[num];
}


void LED_Off (unsigned int num) {

      LPC_GPIO2->FIOPIN &= ~led_mask[num];
}
```

Alla funzione LED_On o LED_Off passo il numero del led interessato e nel primo caso riporto il numero 1 nella posizione rispettiva del led, mentre con la seconda funzione il bit del LED3 diventa 0, in realtà il nostro array risulterà essere così:

LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
1	1	0	1	1	1	1	1

Come possiamo osservare abbiamo richiamato led_mask[] ovvero la maschera che mi consente di spostare il valore 1 alla rispettiva posizione.

```
#define LED_NUM     8    /* Number of user LEDs                */

const unsigned long led_mask[] = { 1UL<<0, 1UL<<1, 1UL<<2, 1UL<< 3,
           1UL<< 4, 1UL<< 5, 1UL<< 6, 1UL<< 7 };
```

Passaggio fondamentale perché tutto funzioni è definire che i led che in realtà si chiamano `LPC_GPIO2` sono elementi di output

```
LPC_GPIO2->FIODIR |= 0x000000ff;
```

Disponendo di otto led è possibile consentire all'utente attraverso il pannello di controllo di scegliere quale led accendere o spegnere per questo motivo dobbiamo controllare quale led il nostro utente predilige.

Per conoscere questa informazione dobbiamo rivolgerci nuovamente all'indirizzo url dove viene riportato il numero del led preferito.

http://192.168.1.123/led?led=3

Dovremmo prelevare il numero del led che come possiamo osservare viene riportato dopo 'led=', procediamo con il linguaggio

```
int LED_action()
{
      int i = 0;

if(TCP_RX_BUF[++i] == 'l' && TCP_RX_BUF[++i] == 'e' && TCP_RX_BUF[++i] == 'd')
            if(TCP_RX_BUF[++i] == '?')
                  if(TCP_RX_BUF[++i] == 'l' && TCP_RX_BUF[++i] == 'e' && TCP_RX_BUF[++i] == 'd' && TCP_RX_BUF[++i] == '=' )
```

Estrapolando il valore presente dopo 'led=' possiamo affermare che conosciamo il numero del led, ma purtroppo in realtà è di tipo char ovvero un carattere e non un intero. Per questo motive dobbiamo convertirlo in intero. Questa scelta ci aiuterà molto ad ottimizzare il codice per una migliore gestione dei led.

La conversione risulta essere alquanto sofisticata, infatti per compiere questo passaggio dobbiamo ricorrere all'utilizzo della tabella ascii:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Come possiamo osservare nella tabella ascii il primo intero risulta essere il 48 per cui la conversione avviene sottraendo 48 alla variabile che abbiamo estrapolato dall'indirizzo della pagina. Ora il numero del led è a tutti gli effetti un intero.

```
return TCP_RX_BUF[++i] - 48;

}
```

Ora conosciamo il numero del led con il quale l'utente vuole interagire e non è più un carattere ma un intero.

speaker

Gestire una richiesta per riprodurre un suono è molto simile alla richiesta precedentemente studiata. Ovvero devo trovare all'interno della barra dell'indirizzo la parola sound che si trova dopo il carattere '/'

```
int i = 0;


while(TCP_RX_BUF[i] != '/')
          i++;
if(TCP_RX_BUF[++i] == 's' && TCP_RX_BUF[++i] == 'o' && TCP_RX_BUF[++i] ==
'u'  && TCP_RX_BUF[++i] == 'n'  && TCP_RX_BUF[++i] == 'd')
          return 1;
return 0;
}
```

logout

La terza funzione che mi sembrava corretta inserire dal punto di vista funzionale è il logout. Attraverso il pulsante rosso in alto a sinistra della pagina del pannello di controllo l'utente può richiamare questa funzione:

```
int logout()
{
     int i = 0;

     while(TCP_RX_BUF[i] != '/')
          i++;

     if(TCP_RX_BUF[++i] == 'l' && TCP_RX_BUF[++i] == 'o' && TCP_RX_BUF[+
+i] == 'g')
          return 1;

     return 0; }
```

Il logaritmo in pratica cerca la stringa 'log' all'interno del indirizzo url e restituisce 1 se trova la stringa altrimenti 0.

1
password

Una volta che siamo riusciti a estrapolare la password, la scheda inizierà ad effettuare i dovuti controlli. In particolare a segnalarci se la password inserita è corretta è una variabile, la quale assumerà dei valori, 0 se la stringa inserita è sbagliata, 1 se è corretta. Questo controllo lo esegue PW_search() che abbiamo riportato nel precedente punto.

Ora dobbiamo eseguire una totale gestione della richiesta formulata dal client e restituire una corretta risposta.

Esaminando dal punto di vista pratico e funzionale il progetto, all'utente viene richiesto l'inserimento della password per accedere ad un pannello di controllo che gli consente di interagire con la scheda elettronica, il nostro server. La risposta inviata dal server al client è in realtà un array che per il client risulterà essere una pagina web.

L'elaborazione che il server è chiamato ad eseguire nel linguaggio comprensibile alla scheda risulta essere:

```
void gestisci_richiesta()
{
```

Se non ho ricevuto ancora la password o l'ho ricevuta sbagliata, il flag pw_flag è uguale a 0

```
      if(pw_flag == 0)
```

Richiamo il controllo che verifica che la password inserita è corretta

```
            if(PW_search() == 1)
            {
                  pw_flag = 1;
```

Ora posso attribuire al pw_flag il valore 1. In tale situazione devo restituire al client l'array contenete la pagina web relativa al pannello di controllo.

Per compiere questa operazione utilizzerò la funzione memcpy la cui sintassi mi impone:

memcpy (contenitore, sorgente, lunghezza del sorgente);

```
memcpy(WebSide,WebSide_con_password,sizeof(WebSide_con_password));
                  logout_flag = 1;
            }
```

2

led

Per una migliore comprensione del codice si è scelto di inserire il risultato di LED_action(), appena esaminato nella funzione precedente del server, in una variabile

```
req = LED_action();
```

Il primo controllo che effettuo consiste nel vedere se l'utente ha premuto un pulsante, se req è diverso da 0 allora vuol dire che un pulsante è stato premuto per ciò dovrò accendere o spegnere fisicamente il led.

```
if(req != 0)
      {
      if(led_flag[req])
            LED_On(req);
      else
            LED_Off(req);

      led_flag[req] = !led_flag[req];
      }
```

Come è possibile vedere vengono chiamate le funzioni che prima abbiamo anlizzato. L'ultima riga indica che il valore di un rispettivo led non deve essere uguale in questo modo abbiamo la certezza che sia 0 o 1 ovvero spento o acceso.

logout

Come abbiamo visto precedentemente la funzione logout ci restituisce 1 se l'utente ha premuto il pulsante per uscire dal pannello di controllo altrimenti 0.

Se l'utente desidera effettuare il logout allora verrà indirizzato alla pagine del login. Per eseguire questa operazione restituiremo al client l'array contenente la pagine del login:

```
if(logout() == 1 && logout_flag == 1)
      {

            memcpy(WebSide,WebSide_senza_password,sizeof(char) *
sizeof(WebSide_senza_password));

            for(i = 0; i < 7; i++)
                  led_flag[i] = 0;

            pw_flag = 0;
```

```
            logout_flag = 0;
      }
```

Oltre a questo passaggio risulta importante visto che il controllo della scheda si può avere solo dal pannello di controllo dovremmo procede con il spegnimento di tutti i led attribuendoli tutti il valore 0.

Dovremmo anche azzerare la variabile che mi indica che è stato effettuato il login per consenti una volta tornati alla pagine del login di effettuare nuovamente l'accesso al pannello di controllo.

speaker

Una volta che ho ricevuto la richiesta indico al mio convertitore digitale analogico di riprodurre un suono che graficamente risulta essere un'onda quadra che assume un valore di +255 per 1000 volte e un valore -255 per 1000 volte.

```
if(SOUND_action())
      {
            int j = 100;
            while(j--){
                  for(i = 0; i < 1000; i++)
                        LPC_DAC->DACR = 255;

                  for(i = 0; i < 1000; i++)
                        LPC_DAC->DACR = -255;
}
```

Approfondiamo questo aspetto …

Elaborazione numerica dei segnali

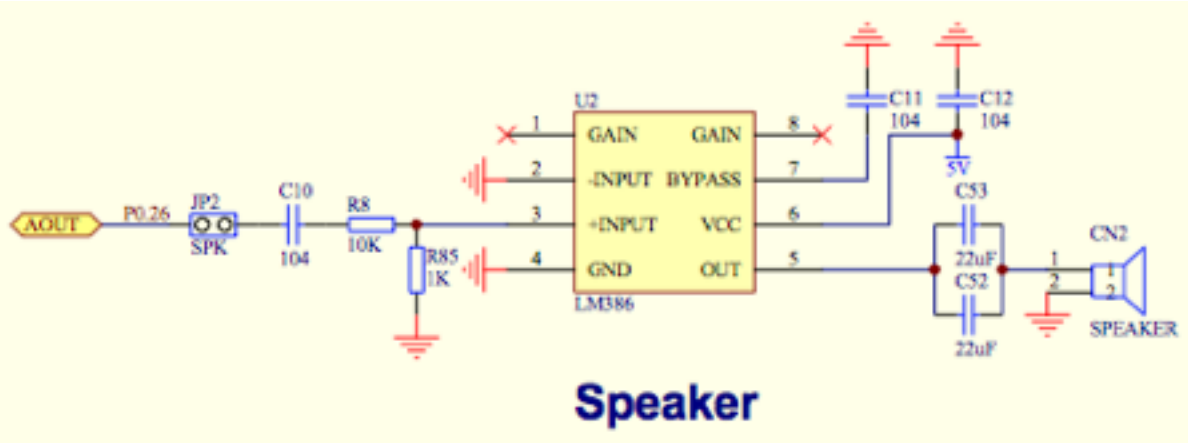
Attraverso un'interfaccia web possiamo scegliere di fare riprodurre un suono alla scheda.Precedentemente abbiamo esaminato come questa funzione possa essere realtà solamente da un punto di vista software.

Fino a questo momento infatti abbiamo potuto vedere che indicando al nostro Digital to Analog Converter, DAC, un valore numerico esso riesce ad emanare un suono.

In realtà il DAC è solo uno dei blocchi funzionari fondamentali per riprodurre il suono. Il suo compito consiste nella conversione di un valore digitale o meglio numero, ricevuto in input, in un valore analogico che restituisce in output.

Ora proviamo ad approfondire questa operazione avvicinandoci fisicamente alla scheda elettronica. In pratica possiamo rappresentare l'intera funzione con il seguente schema funzionale:

Mentre lo schema elettrico effettivo si presenta così:



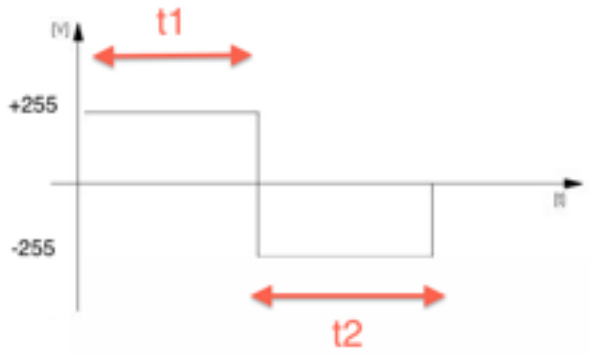
Conversione Digitale / Analogica

In natura, quasi tutte le grandezze fisiche sono analogiche, ovvero non numerabili e di conseguenza non possono essere direttamente elaborati dai calcolatori che hanno natura finita. Questo è il principale motivo per cui i dati analogici devono essere convertiti in formato digitale.

Il principio di funzionamento del convertitore D/A consiste nell' associare ad un dato digitale una grandezza analogica, in generale una tensione.

Il dato digitale è un numero, che nel caso dei calcolatori viene rappresentato secondo le regole binarie. I valori numerici usati nella numerazione binaria sono solamente due (zero e uno), anche i valori di tensione ad essi associati saranno solamente due (zero o circa zero per indicare il valore numerico *zero*; Vcc o circa per indicare il valore numerico *uno*).

Il segna in uscita possiamo rappresentarlo con il seguente grafico:



Con l'algoritmo che troviamo nel paragrafo 'speaker' con il quale effettuiamo una gestione dello speaker abbiamo indicato tutti i parametri qui sopra presenti. Ovvero t1 e t2 sono due cicli for di una durata pari a 1000, mentre Vx e -Vx sono i rispettivi valori assegnati al DAC ovvero +255 e -255.

Glossario

Indirizzo IP: numero di 32 bit scritti convenzionalmente nella notazione decimale puntata, che identifica in modo univoco ciascuna stazione sulla rete.

Subnet Mask: o “maschera di sottorete”, indica il metodo utilizzato per definire il range di appartenenza di un host all’interno di una sottorete IP al fine di ridurre il traffico di rete e facilitare la ricerca e il raggiungimento di un determinato host con relativo indirizzo IP della stessa.

Cavo UTP: è l’acronimo di Unshielded Twisted Pair e identifica un cavo non schermato utilizzato comunemente per il collegamento nelle reti ethernet. La lunghezza massima di un cavo UTP nello standard ethernet è di 100 m. Un cavo UTP la cui termina con connettori di tipo RJ-45 che si innestano direttamente nell’interfaccia del dispositivo.

Array: è un’insieme di variabili, tutte dello stesso tipo, identificato da un nome unico. Gli elementi dell’array sono disposti in memoria in posizioni consecutive. Per accedere ai singoli elementi di un array, è necessario specificare il nome della variabile array e la posizione dell’elemento di interesse tramite un valore intero che si definisce indice.

Stringa: è un’insieme di caratteri quali parole, nomi, frasi …

Algoritmo: è una successione finita di passi elementari che risolvono il problema dato.

Processo: l’esecuzione sequenziale, una dopo l’altra, di istruzioni elementari. Un processo è anche un programma in esecuzione.

Thread: o processo leggero è una sequenza di istruzioni di un programma in corso di esecuzione.

Sitografia

http://www.iet.unipi.it/a.bechini/master.it/docs/01_Client_server.pdf

http://www.dsi.unifi.it/~poneti/Lezione4C.pdf

http://www.dii.unisi.it/~benelli/scienze_della_comunicazione/dispense/2005_06/ClientServer.pdf