Progetto Sistemi di Elaborazione

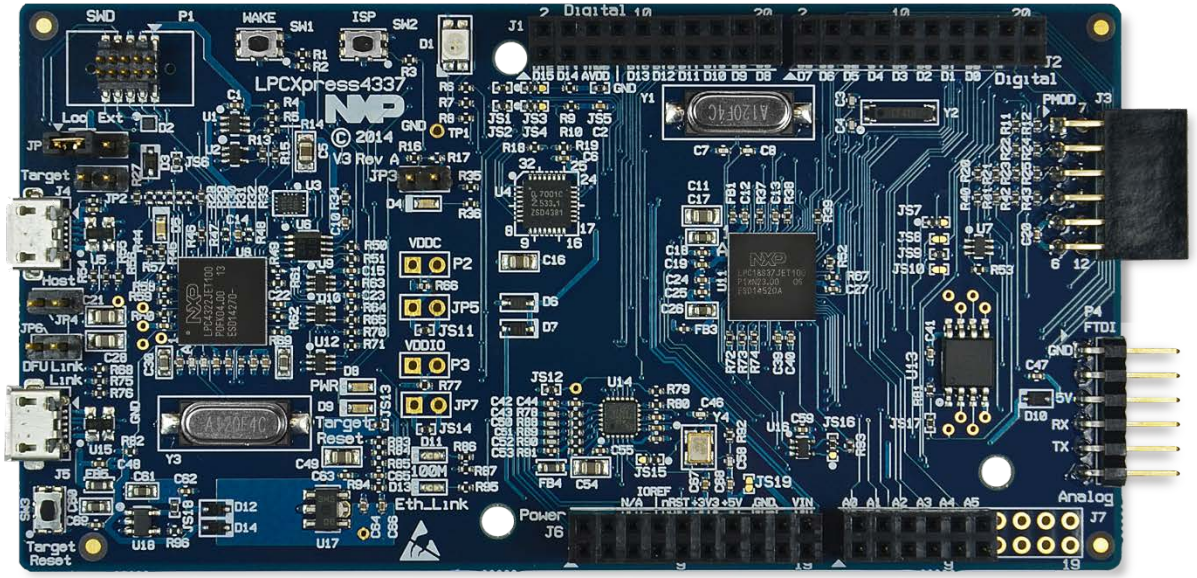
LPC18s37 ARM Cortex-M3

Lorenzo Campioni 111440
Michele Fraccaroli 111140

Indice

- Enviroment
- Obbiettivi
- Sviluppo
- Test
- Conclusioni

LPC18s37 ARM Cortex-M3



Specifiche tecniche

- 180MHz CPU clock
- 1 MB di flash
- 136 kB SRAM
- 16 kB EEPROM
- Integrated AES engine
- ARM Cortex-M3 Harvard architecture

LPCXpresso IDE

L’intero sviluppo e building del progetto sono stati interamente fatti su LPCXpresso, un IDE basato sulla piattaforma Eclipse che permette il debugging tramite SWD e JTAG.

LPCXpresso offre features per lo sviluppo per microcontrollori Cortex-M based LPC e Kinetics.

Indice

- Enviroment
- Obbiettivi
- Sviluppo
- Test
- Conclusioni

FreeRTOS

FreeRTOS è un sistema operativo real time per sistemi embedded.

Il kernel è composto da file in C che forniscono metodi per lo sviluppo multitasking tramite primitive di sincronizzazione, di gestione dei task e creazione di strutture dati condivise.

lwIP - Lightweight IP

lwIP è uno stack TCP/IP open source progettato per sistemi embedded .

Questo stack implementa la maggior parte dei protocolli di livello 2, 3 e 4 dello stack OSI e implementa delle socket API simili alle Berkeley socket.

HTTP

Al di sopra delle API fornite da lwIP, il protocollo applicativo preso in esame per questo progetto è stato HTTP.

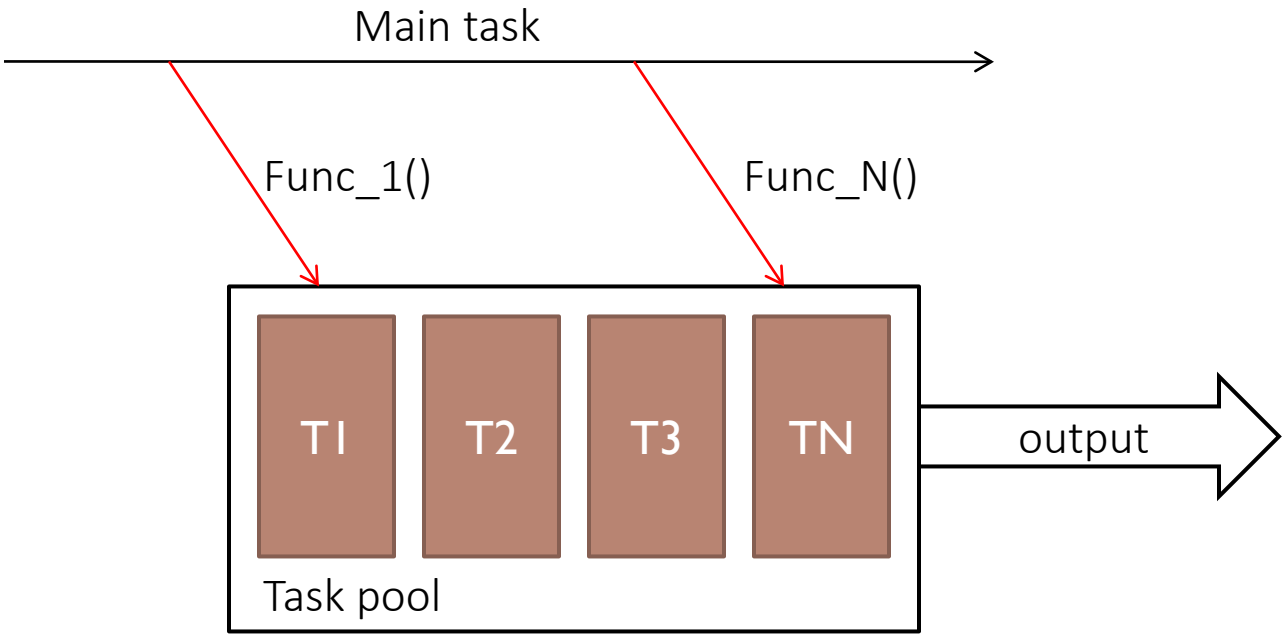
In particolare, per creare un web server, è stato implementato un parser testuale per poter gestire delle semplici richieste HTTP.

Indice

- Enviroment
- Obbiettivi
- Sviluppo
- Test
- Conclusioni

Worker tasks

Il lavoro su FreeRTOS si è incentrato sulla creazione di un pool di task concorrenti che svolgono i compiti che un task principale (main task) gli assegna.



Comunicazione tra task

Per implementare un corretto sincronismo nella comunicazione tra i task abbiamo utilizzato le queue.

Le queue sono dei costrutti forniti da FreeRTOS che permettono l'inserimento e l'estrazione (default FIFO) di dati presenti in esse.

Acquisizione dei parametri HTTP

Il lavoro principale svolto sul web server si è incentrato sul filtraggio dei parametri HTTP.

Il metodo HTTP usato per il passaggio di parametri è il GET che prevede il passaggio di questi nell'URL.

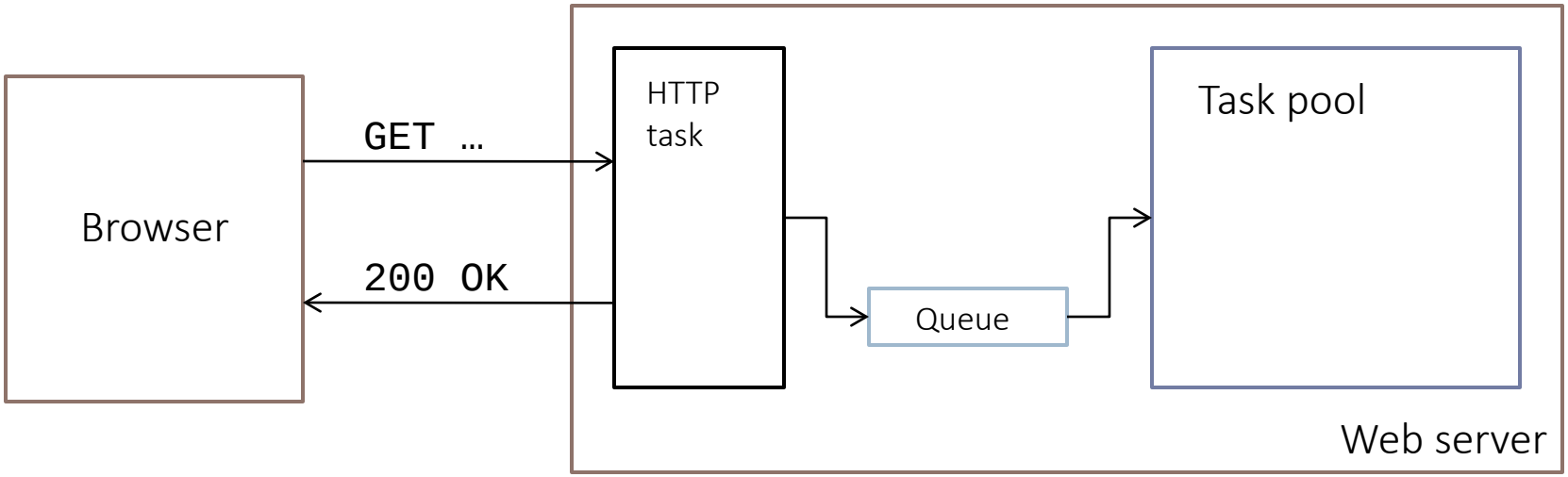
Formato richiesta:

```
GET /index.html?par1=val1&par2=val2 HTTP/1.1
```

Web server

I parametri acquisiti sono poi utilizzati come decisori di quali funzioni, il web server, dovrà svolgere.

Per evitare il rallentamento del task che risponde alla richiesta HTTP, abbiamo sfruttato il pool di task precedentemente sviluppato durante il lavoro su FreeRTOS.



Indice

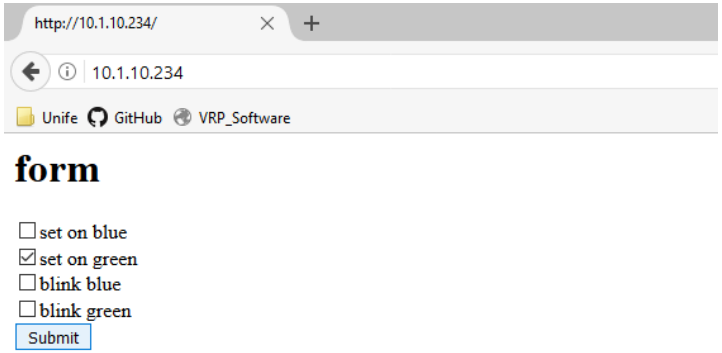
- Enviroment
- Obbiettivi
- Sviluppo
- Test
- Conclusioni

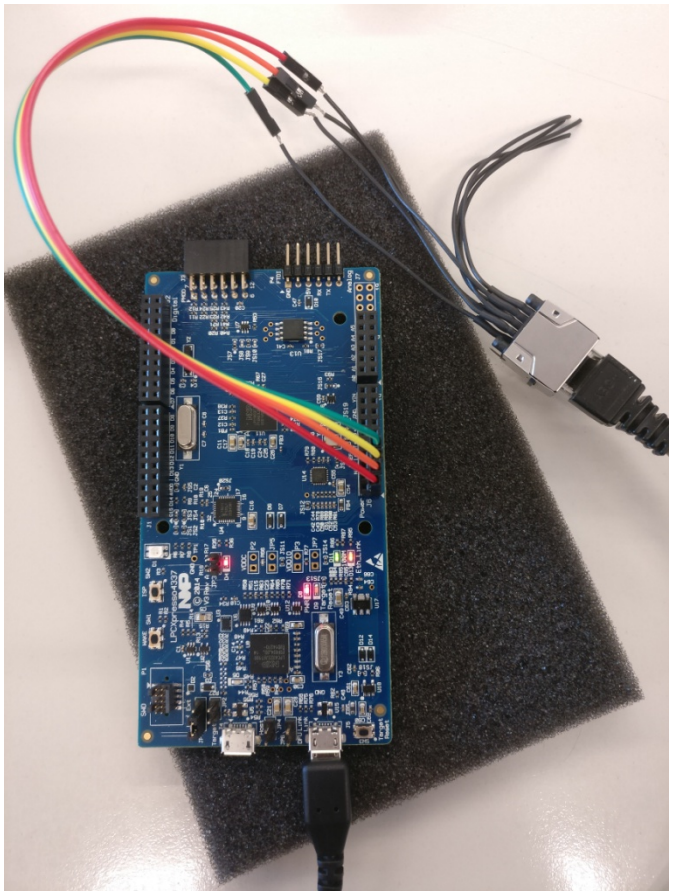
Test del sistema

Per testare il web server abbiamo scelto di utilizzare un browser come client e dei semplici led come elementi da controllare da remoto.

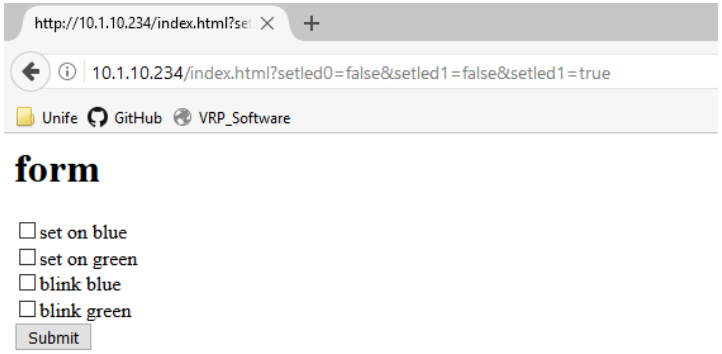
Questa scelta ci ha permesso di testare in modo efficace il funzionamento complessivo di tutti gli elementi che compongono il web server.

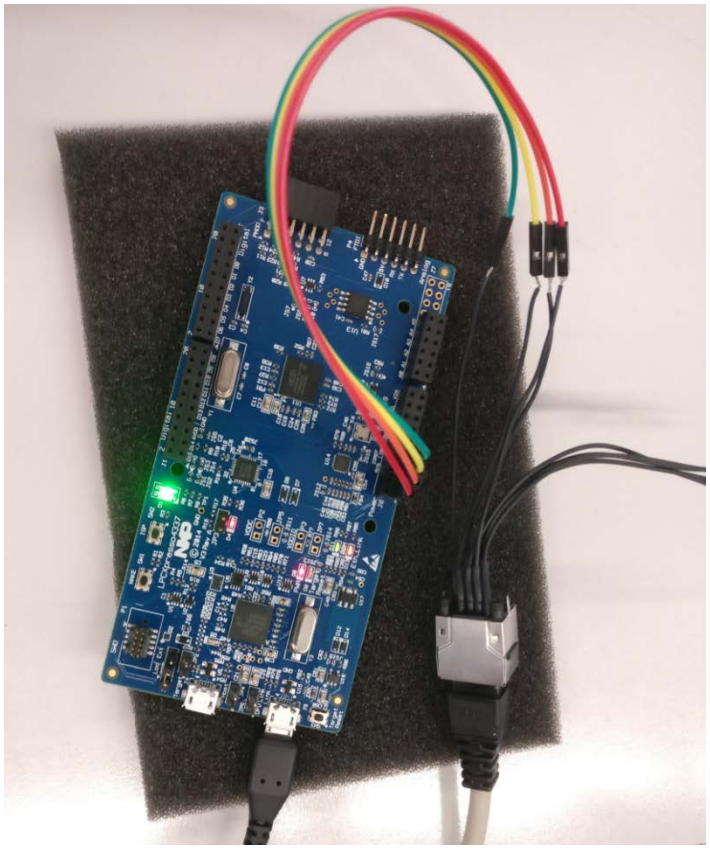
Test del sistema





Test del sistema





Indice

- Enviroment
- Obbiettivi
- Sviluppo
- Test
- Conclusioni

Conclusioni

A causa di una limitata capacità computazionale, il dispositivo in esame non è ottimale per gestire una vera e propria sessione HTTP (cifratura a chiave pubblica, gestione cookies, sessioni, etc).

Tuttavia, la capacità di utilizzare protocolli di alto livello (e.g. HTTP) per gestire l'eterogeneità dei dati lo rendono adatto per essere impiegato come nodo intermedio tra un gateway (che fornirà poi il front-end dell'applicazione) e una rete di nodi constrained (e.g. IoT).