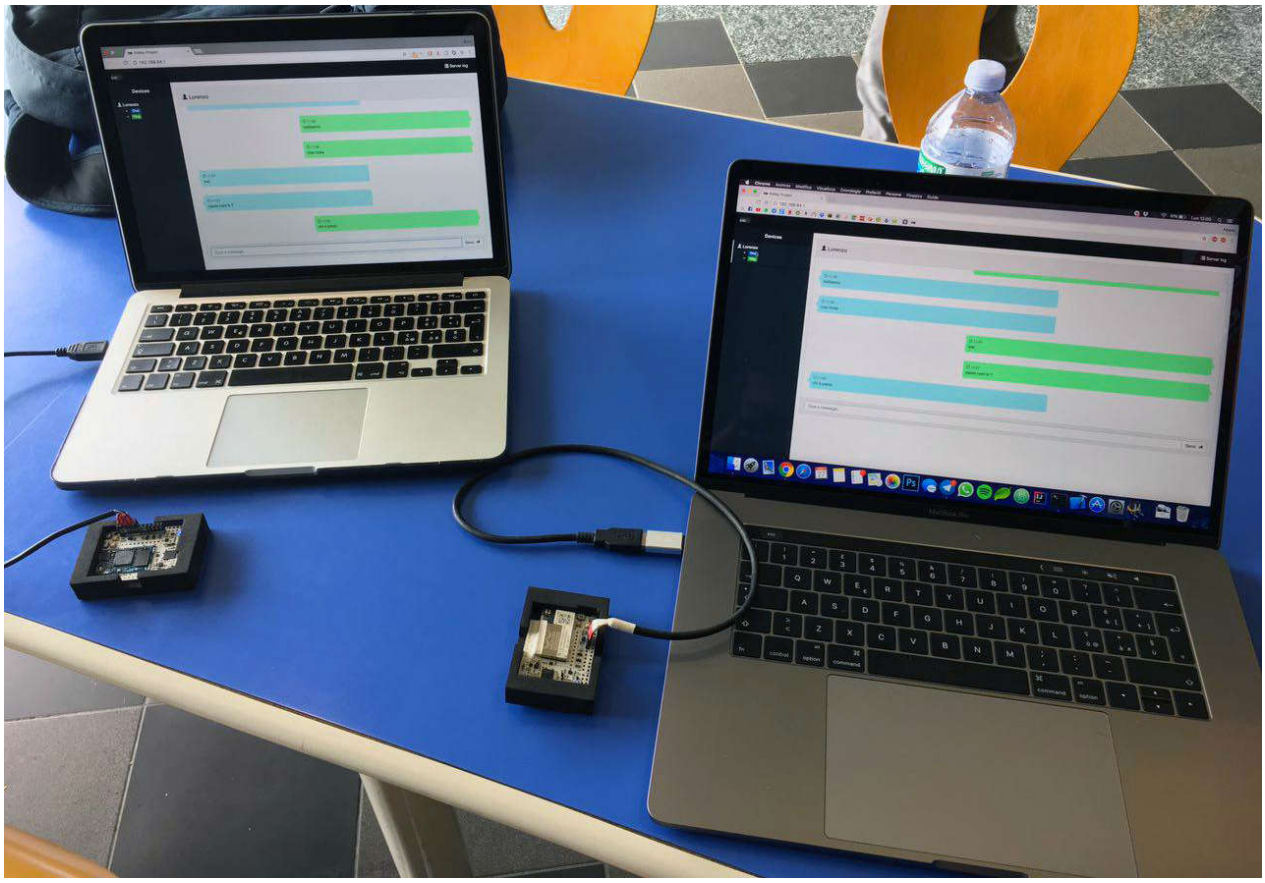
Progetto 64K



Servizio Chat attivo su due pc interconnessi tramite una rete mesh

Descrizione

Il progetto che abbiamo deciso di intraprendere all'inizio di questo semestre aveva come obiettivo quello di rendere possibile la condivisione di contenuti di vario genere su più computer attraverso una rete mesh.

Siamo riusciti a raggiungere i nostri scopi collegando un dispositivo specifico (con un cavo USB) ad ogni computer e avviando una pagina web che viene servita dal dispositivo stesso, già collegato con tutti gli altri dispositivi in rete, e quindi con gli altri computer.

L'idea di base non aveva particolari requisiti applicativi bensì era volta ad implementare in maniera funzionale ed efficiente una comunicazione tra più device proprio attraverso una rete mesh.

La componente software, che si sarebbe servita di tale layer comunicativo, era inizialmente volta solo ad implementare una chat ma, nel corso del progetto, è stata rivista e si è giunti a rendere disponibile un'interfaccia semplice che permettesse agli sviluppatori futuri di creare servizi generici come condivisione di documenti, chat, lavori di gruppo real-time e molto altro.

Ad oggi i dispositivi, che verranno descritti successivamente, quando vengono accesi entrano a far parte di una rete mesh.
Stabiliscono una connessione di livello 4 (TCP) gli uni con gli altri, in modo da essere pronti a scambiare dati.

Sopra a questa implementazione, scritta in Python e che agisce come un Router di pacchetti, c'è una seconda parte applicativa scritta in Javascript, CSS e HTML che implementa la GUI web con i vari servizi disponibili.

Il dispositivo è configurato per apparire al computer host come un adattatore Ethernet, formando una rete IP con esso. In questo modo, il computer può accedere all'interfaccia web presente sul dispositivo, che sarà in grado di comunicare con il software Python tramite WebSocket, e quindi anche con qualsiasi altro dispositivo collegato alla rete mesh.

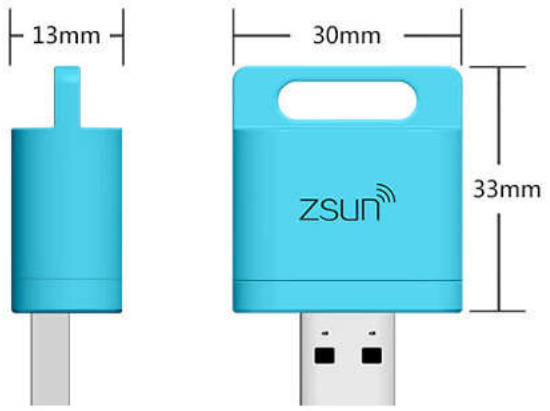
Prima di entrare nel dettaglio è bene sottolineare cosa si intenda per alcuni termini che ricorreranno spesso:

- Rete mesh: Una rete mesh è una rete IP Wireless tra più dispositivi che non necessita di alcun access point. Non è quindi presente un dispositivo centrale che gestisce i vari dispositivi connessi in quanto ogni dispositivo è collegato direttamente agli altri.
  La rete mesh offre quindi un grande vantaggio sotto questo punto di vista poiché può essere create ed utilizzata in qualsiasi zona, bastano due dispositivi.
  Inoltre, essendo una rete multi-hop, anche se un dispositivo C non dovesse essere direttamente visibile dall'interfaccia wireless di A, è comunque possibile raggiungerlo passando per un dispositivo B posto tra i due e raggiungibile da entrambi.
- Servizio applicativo: Un servizio applicativo, così come è stato inteso in questo progetto, è un modulo scritto con un linguaggio di scripting (JavaScript) che viene caricato da un browser. Il servizio è un'entità di alto livello, per così dire, che può comunicare con gli altri dispositivi che ne possiedono un'implementazione valida. Quindi, ad esempio, se due dispositivi hanno il servizio chat installato, è possibile chattare insieme.

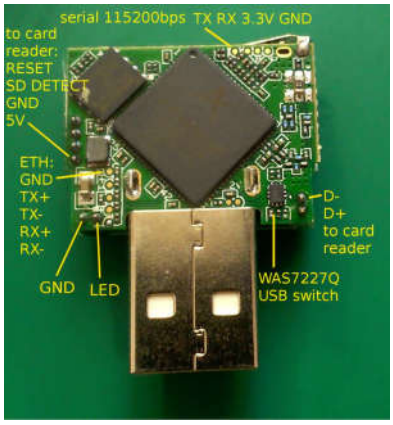
Il dispositivo utilizzato

Inizialmente abbiamo deciso insieme al professore di lavorare su un modulo molto compatto.

ZSun Card reader.



Come appare lo ZSun card reader



ZSun card reader PCB

Nonostante sia stato pensato per leggere delle micro SD e renderne disponibili le foto tramite un access point, ha catturato la nostra attenzione in quanto era costruito col SoC MIPS AR9331. Questo

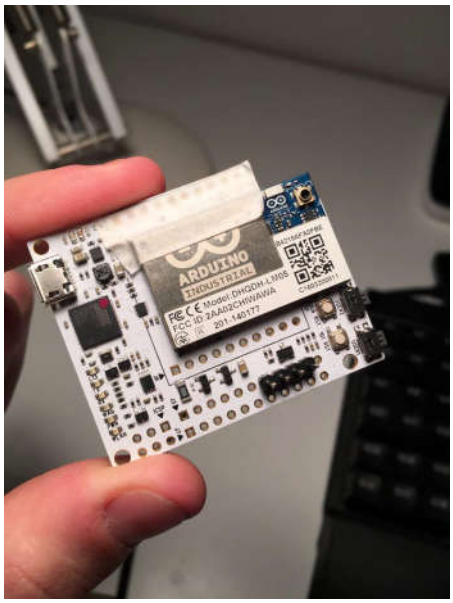
piccolo SoC possiede un'interfaccia wireless a 2.4GHz che ci è tornata molto utile per i nostri obiettivi e, inoltre, è ampiamente supportato dalla leggerissima distribuzione per router OpenWRT.

Shell seriale sulla nuova immagine di OpenWRT installata sullo ZSun

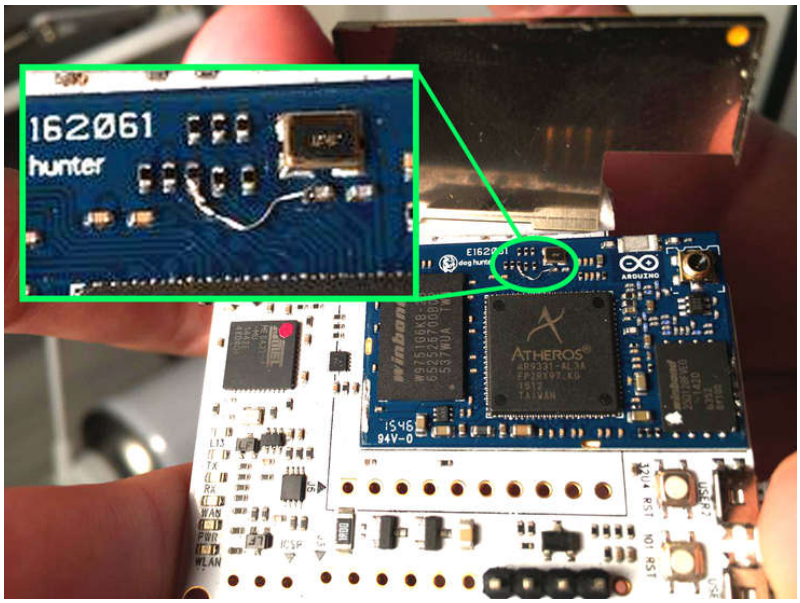
Dopo esserci connessi direttamente al modulo in questione attraverso l'interfaccia seriale e aver flashato la nuova immagine di OpenWRT abbiamo cominciato ad osservare i primi problemi.
Nello specifico non era possibile comunicare direttamente col PC tramite la USB in quanto questa era connessa con uno switch hardware allo slot SD e al SoC e non eravamo in grado di farla funzionare come Slave per sembrare, agli occhi dei pc, un'interfaccia di rete.

Dopo molti vani tentativi abbiamo quindi optato per un dispositivo diverso che superasse questa limitazione ma che adoperasse lo stesso SoC (AR9331):

Arduino 101 Industrial



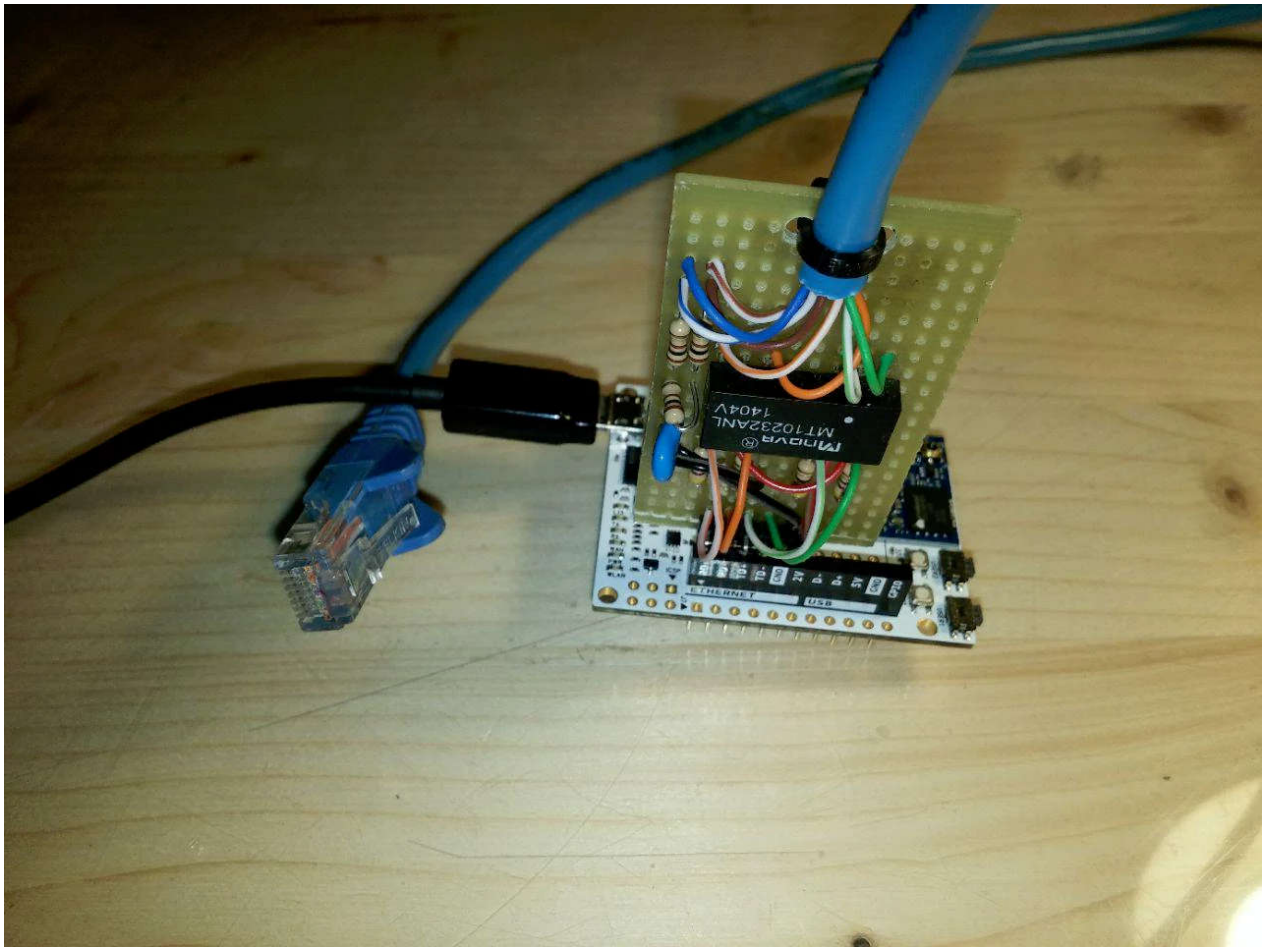
Arduino Industrial 101



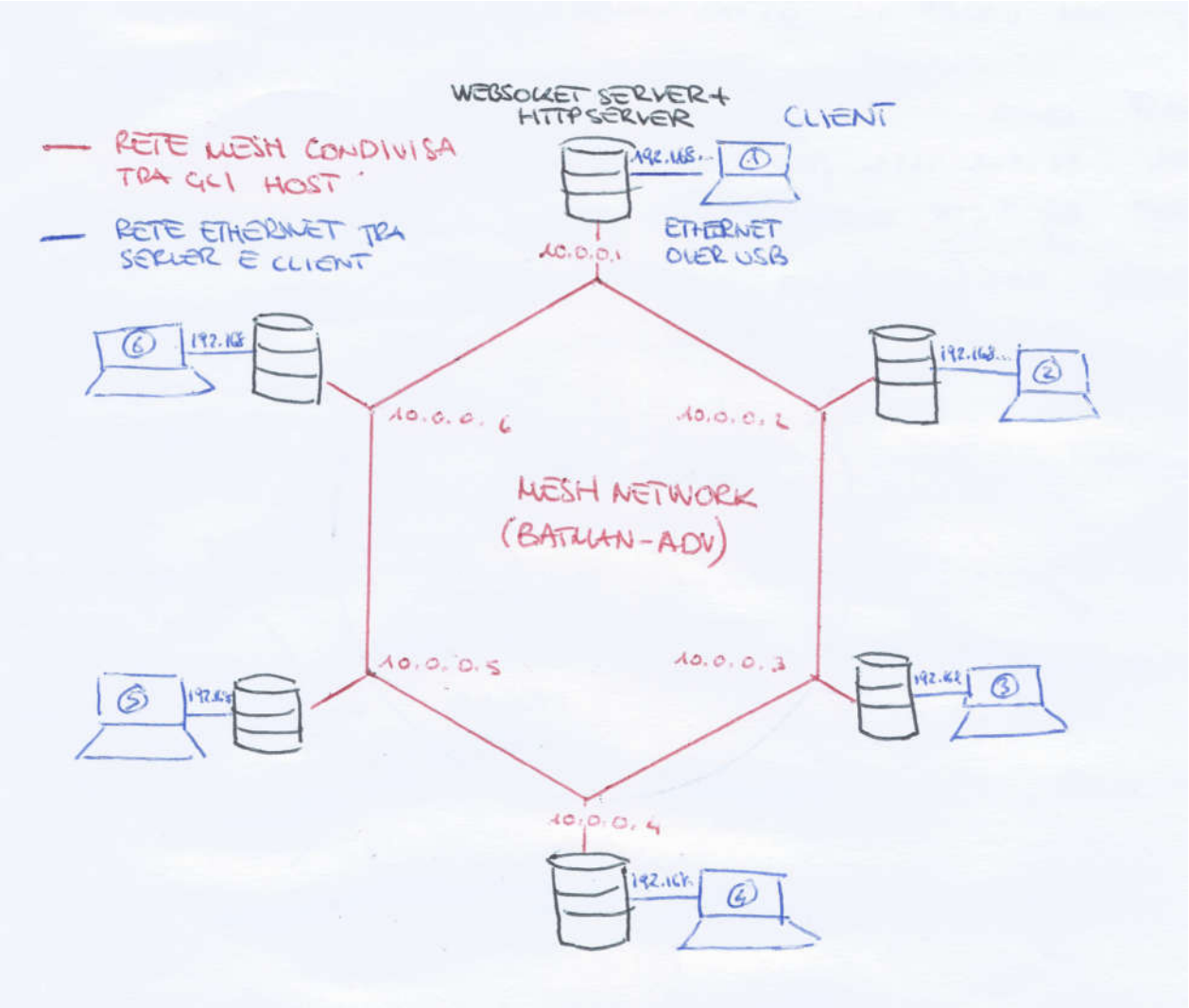
Il jumper aggiunto per impostare la USB in modalità slave.

Questa volta, con l'ausilio di un piccolo jumper, siamo riusciti a forzare la modalità di operazione (a livello hardware) della USB a slave e far funzionare, dopo aver creato e compilato un'immagine ad-hoc

di OpenWRT, l'interfaccia di rete su USB.
Da questo momento per accedere alla console non potevamo più utilizzare la USB e quindi abbiamo dovuto utilizzare una delle due interfacce ethernet presenti sull'AR9331.
È stato realizzato un piccolo circuito aggiuntivo con i componenti necessari al funzionamento dell'interfaccia Ethernet, visto che i trasformatori non sono presenti sulla scheda.



La struttura di rete



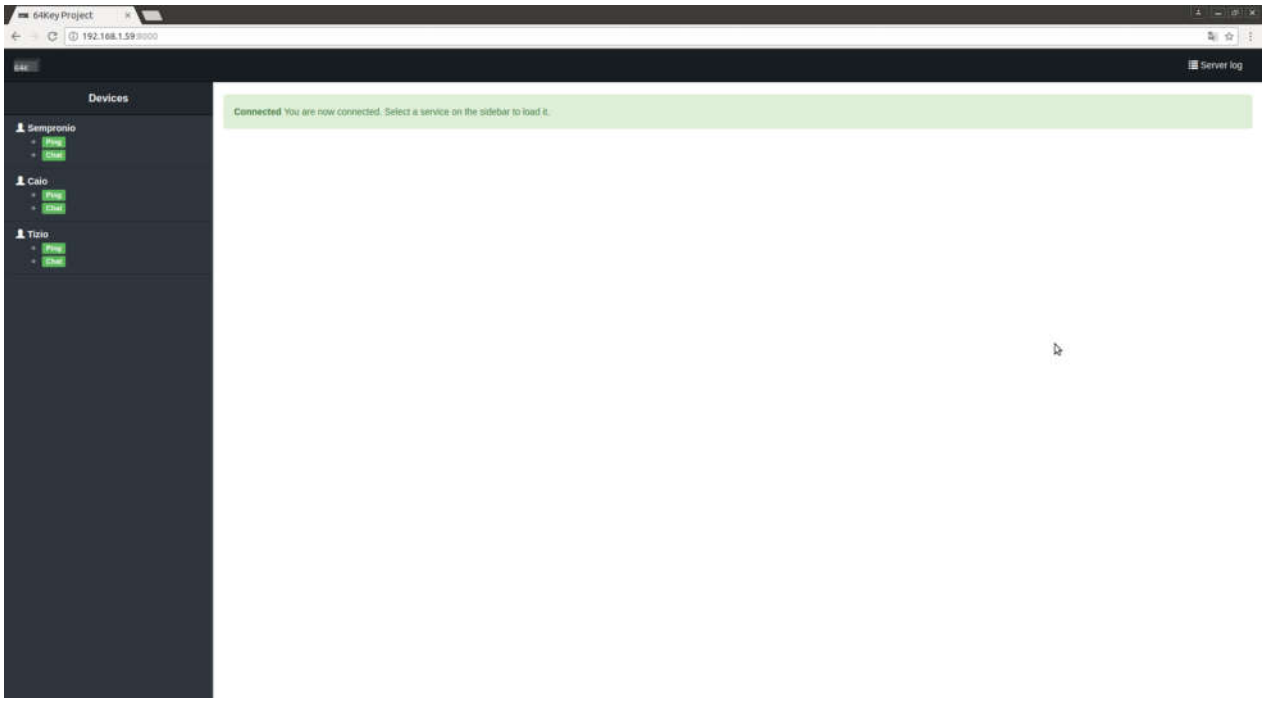
I vari dispositivi (Schematizzati in Nero) fanno parte di una rete mesh (Colorata in rosso) che viene creata attraverso l'uso di un potente modulo kernel per linux chiamato B.A.T.M.A.N Advanced che opera a livello 2 (Layer MAC).
A differenza di molti altri protocolli di routing wireless che operano a livello 3 (IP), batman invia frame ethernet raw, operando sullo strato ISO/OSI e garantisce molta affidabilità.

La rete mesh ha una propria classe di indirizzi IP 10.0.0.0/8 che può intendersi come una WAN per un normale computer, nella quale sono presenti tutti gli altri host.

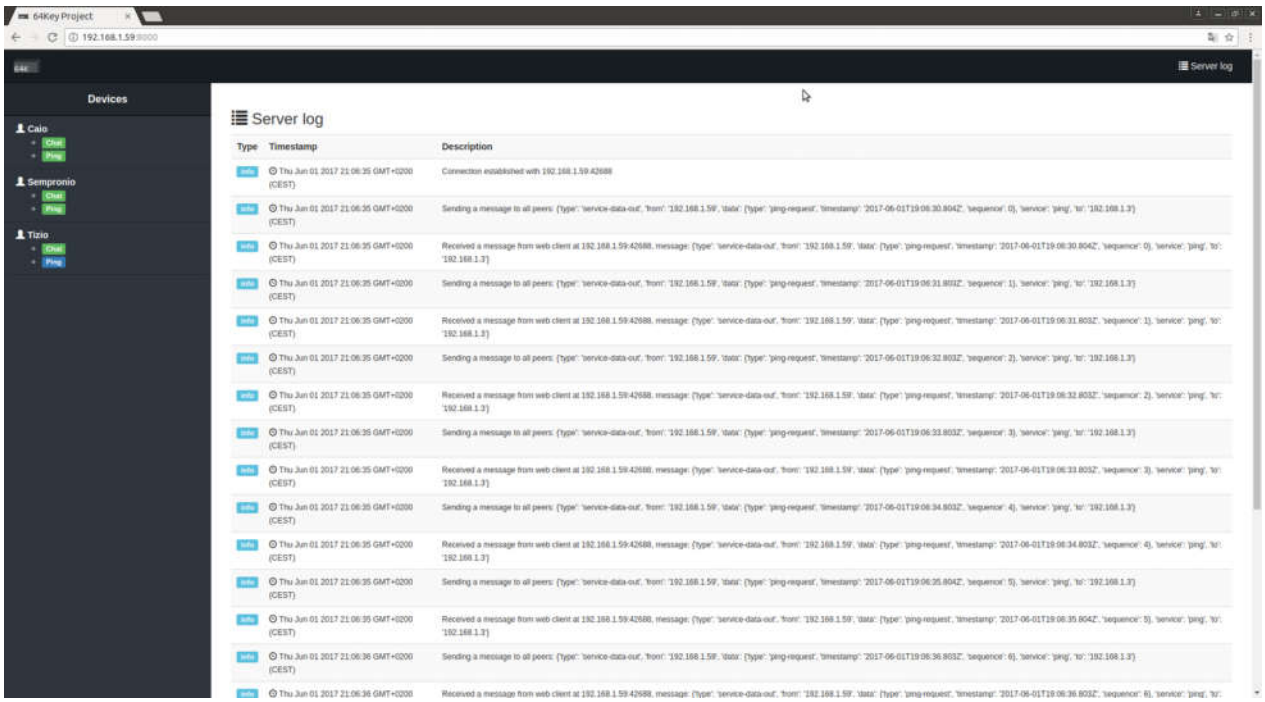
Ogni dispositivo è collegato a un computer con un generico sistema operativo (Linux, macOS, Windows) attraverso un cavo USB su cui viene implementato il protocollo ethernet-over-usb.
In questo modo il dispositivo si presenta come adattatore Ethernet USB per il computer, che si collega in automatico e ottiene un indirizzo IP tramite DHCP.
Questa seconda rete, identica per ogni coppia Computer-Dispositivo, ha un'altra classe IP 192.168.64.1/24 che è sconnessa dalla rete mesh.

Il dispositivo quindi, una volta connesso al computer, avvia un server HTTP che inizia a servire le pagine web con sopra i vari servizi a quest'ultimo.
Nel contempo agisce come una sorta di router e indirizza i pacchetti generati dalla pagina caricata dal browser al destinatario richiesto.
Infatti gli applicativi permettono l'interazione con uno o più host nella rete e conseguentemente le informazioni vanno trasferite correttamente.

La GUI

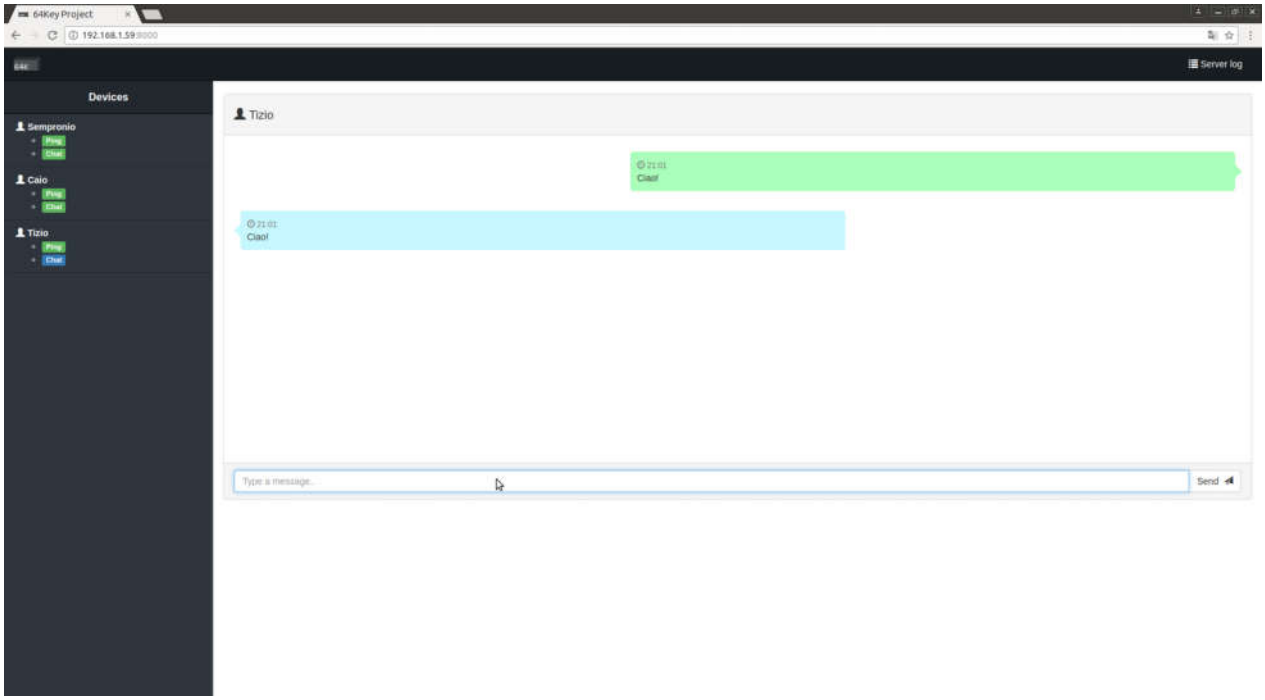


La GUI si presenta come una normalissima pagina web.
E' stata realizzata con l'ausilio del framework Bootstrap, realizzato da Twitter, che nella sua semplicità rende l'esperienza utente dinamica e funzionale.
Una sidebar laterale mostra i vari devices collegati alla rete mesh mentre al centro c'è lo spazio per visualizzare il servizio attivo.
Ad ogni device è associato un nome, sotto al quale compaiono i vari servizi disponibili su quel dispositivo.
Ogni dispositivo può avere servizi diversi installati, ma due device potranno interagire tra loro attraverso lo stesso servizio solo se sarà stato implementato su entrambi: in questo caso la label relativa al servizio sarà di colore verde altrimenti rossa. Nel caso due device abbiano versioni diverse dello stesso servizio, il colore sarà giallo.
Ma cos'è un servizio ? Come funziona e perché è chiamato in questo modo?
Prima di entrare nello specifico si evidenzia la presenza di una server log, che permette all'utente di verificare come si stia comportando il Server.

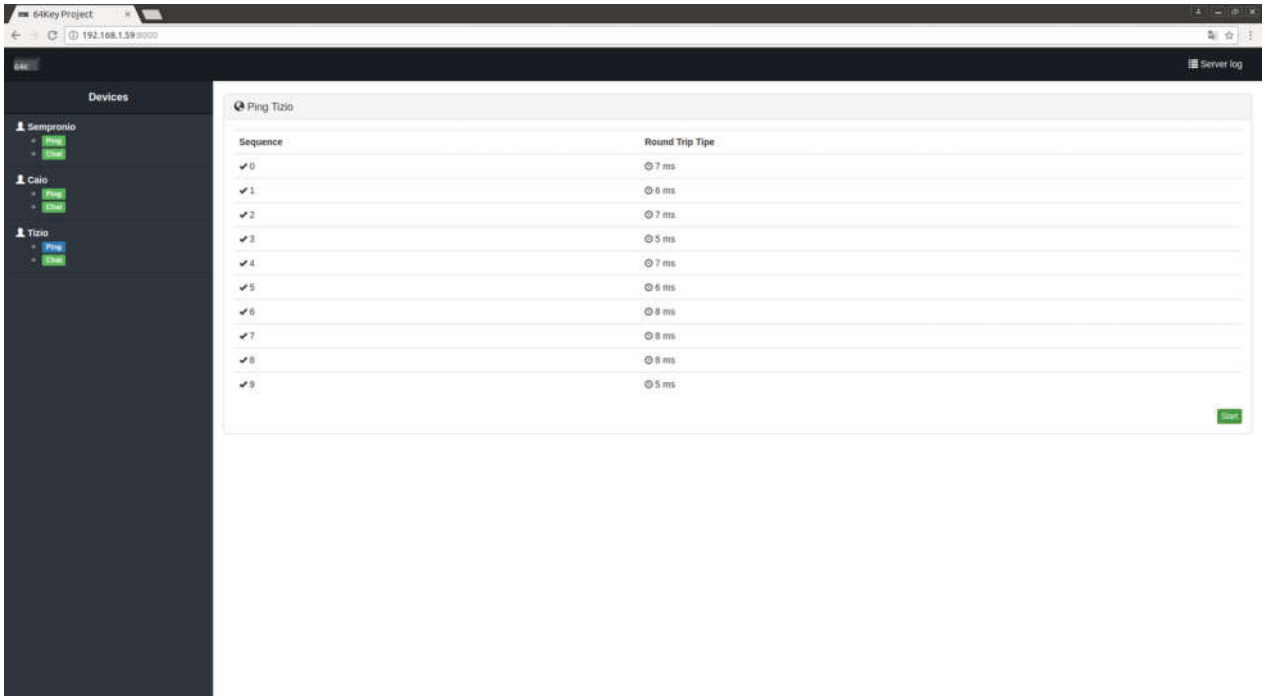


I Servizi

Nell'ottica di astratte il più possibile la componente applicativa, abbiamo deciso di sviluppare nella GUI Web un sistema a Servizi.
Per servizio si intende uno script, realizzato in JavaScript, CSS, HTML e conforme ad una specifica interfaccia, che permette di interagire con altri device nella rete.
L'unico limite per un nuovo servizio è l'immaginazione (..e i 5M su 10 di spazio residui).
Nel nostro caso abbiamo pensato ad una Chat e a un applicativo Ping.



Servizio Chat attivo con Tizio



Ping con Tizio

Tuttavia un complesso gioco multiplayer 3D, magari sviluppato con la potente libreria Three.js, non andrebbe incontro a nessuna difficoltà ad essere implementato, se non l'esperienza del programmatore.
Il cuore pulsante è il Controller; esso riceve i dati dal Server sotto forma di frame JSon, li interpreta, e quindi li inoltra al servizio corretto.

Dunque **se volessimo creare un nuovo applicativo** dovremmo:

1. Creare un nuovo servizio ed estendere la classe base Service.js.
2. Fare overriding dei tre metodi principali (Vedi la documentazione).
3. Istanziare, nel costruttore del controller, il nuovo servizio
4. Sempre nel controller, aggiungere lo switch case **frame.service** per il nuovo servizio nel metodo **parseConnectionFrame()** (Basarsi sugli altri).
5. Implementare il servizio appena creato con la sua logica:
   Dal momento che viene costruito con uno spazio HTML (DOM Object) in cui operare, è libero di prenderne il controllo ed iniettare qualsiasi cosa.
   Potete quindi inserire un elemento con un id particolare e registrarvi come observer per ogni evento su di esso oppure modificare in modo dinamico l'HTML, aggiungere effetti grafici ecc.
6. Registrare il servizio sul Server modificando un file di configurazione (64Key/settings.py) che indica quali servizi sono implementati.

```
//64Key/settings.py

services = {
    'chat': {
        'name': 'Chat',
        'options': {}
    },
    'ping': {
```

```
        'name': 'Ping',
        'options': {}
    }
}
```

Da questo momento in poi quando il vostro servizio invierà qualsiasi dato verso un altro host, questo verrà inoltrato automaticamente al device corretto che, a sua volta, lo inoltrerà alla pagina web su cui risiede lo stesso servizio, questa volta in attesa di dati.

Tornando all'idea del gioco, potreste dover mandare la posizione del vostro giocatore a intervalli regolari a tutti gli altri host nella rete che hanno il gioco in funzione (Servizio caricato). In questo caso basterà creare un metodo broadcast che manderà un frame specifico a tutti.

Tuttavia non è stato ancora affrontato un aspetto chiave: come seleziono l'host (o gli hosts) destinatari ? Dove trovo questa informazione ?

E' semplice.
Nel momento in cui la pagina web viene aperta, come già affrontato, viene aperta la comunicazione col server (Il dispositivo).
Il controller riceve immediatamente un frame di configurazione e, successivamente, non appena un nuovo host entra a far parte della rete, anche le informazioni relative a quest'ultimo come il suo UID (Unique IDentifier).
Viene quindi aggiornata automaticamente l'istanza di Connection.js che possiede una cache con le coppie Nome:UID.

Quindi per parlare con Tizio sarà sufficiente chiamare il metodo di invio frame nella vostra nuova implementazione del servizio con l'UID a lui corrispondente.
C'è un metodo apposito in Connection.js per ottenerlo.
Notare inoltre che l'oggetto connection entra come argomento nel costruttore del servizio generico ed è quindi sempre visibile

```
Service : sendData(to, data); //to is the target UID
Connection : hostNameForUid(uid);
```

Nonostante gli sforzi per cercare di rendere l'implementazione il più semplice possibile mi rendo conto che sviluppare da zero un servizio potrebbe essere apparentemente difficile.
Consiglio in questo caso di guardare l'implementazione degli altri servizi già esistenti e in caso di contattarmi.