Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Routex
Un Router Multitecnologia per Internet of Things

Relatore:
Chiar.mo Prof.
Luciano Bononi

Correlatore:
Dott.
Luca Bedogni

Presentata da:
Filippo Morselli

Sessione II
Anno Accademico 2015/2016

Indice

Capitolo 1

Introduzione

Nel presente documento si descrive la progettazione e l'implementazione di Routex, un sistema per lo sviluppo di applicazioni Internet of Things, il cui componente centrale consiste in un router multi-tecnologia realizzato utilizzando un Raspberry Pi.

Con il termine *Internet of Things* (IoT)[1] si fa riferimento a reti che permettono l'interazione fra le persone e dispositivi fisici o fra dispositivi stessi. I componenti fondamentali dei sistemi IoT sono sensori e attuatori, ossia i device che che fungono da interfaccia fra l'ambiente reale e quello virtuale, e un servizio internet che permette di connetterli in rete.
L'Internet of Things ha come proprio fulcro un processo di raccolta e di scambio di informazioni, che rappresentano lo stato del mondo fisico. Questi dati risultano accessibili tramite la rete e sono oggetto di comunicazioni dirette fra le macchine (M2M) che quindi non necessitano di una costante supervisione umana. In base a condizioni verificabili tramite un'apposita analisi é inoltre possibile effettuare azioni di conseguenza, definendo un comportamento complessivo del sistema.
I sistemi IoT hanno come scopo ultimo quello di fornire servizi che in passato

[1] Termine coniato da Kevin Ashton nel 1999 per indicare l'utilizzo tramite Internet di informazioni ottenute da RFID

non erano realizzabili ed oggi sono possibili grazie allo sviluppo di nuove tecnologie, alla riduzione dei prezzi dei componenti necessari e alla loro relativa facilitá di installazione ed uso.
L'Internet of Things é un ambito in costante sviluppo, ed é oggetto di interesse da parte di numerose aziende e della ricerca scientifica. Come conseguenza si sta ottenendo un aumento sia delle tipologie di oggetti che vengono nativamente realizzati con capacitá di comunicazione wireless, sia del numero di componenti complessivi effettivamente connessi in rete.

Le applicazioni di questo tipo hanno risvolti utili in numerosi ambiti. Fra le applicazioni emergenti si possono inserire:

- Ottimizzazione e controllo dei processi produttivi industriali, tramite raccolta delle informazioni sui prodotti e sul funzionamento dei macchinari
- Tracciabilitá e localizzazione, utilizzando sensori di posizione e tag RFID[2]
- Monitoraggio e sicurezza ambientale, con il fine di prevenzione di cambiamenti potenzialmente pericolosi, ad esempio per l'analisi in tempo reale di parametri legati all'inquinamento dell'aria o dell'acqua
- Domotica, ossia l'automazione applicata alle case e agli edifici, con lo scopo di assistere le persone nella gestione e di ridurre gli sprechi energetici

L'Internet of Things abilita alla realizzazione di un vasto insieme di applicazioni, ma di conseguenza richiede che sia possibile rendere interoperabili fra loro un elevato numero di dispositivi e di protocolli di comunicazione. Infatti piú é ampia la scelta piú risulta facile trovare la tecnologia che meglio si presta al proprio scopo, ma ció comporta anche una maggiore complessitá dell'architettura necessaria al funzionamento.

[2]Radio Frequency Identification

L'incompatibilitá non coinvolge solamente l'aspetto dei protocolli di comunicazione, ma anche il formato delle informazioni. Non si hanno infatti garanzie che il formato dei dati raccolti da sistemi diversi sia lo stesso. I servizi che necessitano di un'analisi complessiva di queste informazioni potrebbero dover effettuare procedure di conversione o di interpretazione, introducendo un ulteriore livello di complessitá.

Una problematica critica da tenere in considerazione nella descrizione dell'Internet of Things é quello relativo alla sicurezza e alla privacy. La progressiva diffusione di questi sistemi comporta infatti un notevole aumento del traffico di dati. Specialmente quando sono coinvolte informazioni private e sensibili é necessario implementare meccanismi di protezione sempre piú affidabili e complessi.

Riguardo i dispositivi, il supporto si deve estendere dalle macchine general purpose, fra cui ormai rientrano anche telefoni e tablet, fino ai sistemi embedded. Sono proprio questi ultimi che determinano le maggiori restrizioni in una rete IoT, dettate dalle loro limitate capacitá di elaborazione e dalla necessitá di ridurre al minimo i consumi energetici, in quanto spesso alimentati a batteria. Come diretta conseguenza si ha che essi non sono sempre in grado di implementare tutte le tecnologie (ad esempio il protocollo IP tramite WiFi) e che, per risparmiare energia, é possibile che si connettano in rete soltanto periodicamente.
Un'altro aspetto che si ritiene importante, specialmente per valutare l'usabilitá di un sistema, é la semplicitá di configurazione di un dispositivo, ossia quanto lavoro é necessario per integrarlo con successo nella rete.

Per quanto concerne le tecnologie wireless, uno degli aspetti principali da analizzare é il metodo tramite il quale si interagisce con tutti quei dispositivi che non implementano il protocollo IP e quindi non permettono comunicazio-

ni dirette tramite la rete internet. Non esistono infatti meccanismi standard per la conversione a protocolli non-IP.
Inoltre si deve valutare l'affidabilitá delle tecnologie utilizzate. Non é infatti realistico assumere che la rete sia sempre disponibile e performante al massimo delle capacitá, ma bisogna invece tenere in considerazioni eventuali ritardi, e predisporre meccanismi di ritrasmissione per essere maggiormente tolleranti a problemi dei canali di comunicazione.
Tutto ció assume rilevanza ancora maggiore quando si interagisce con sensori e attuatori, ossia oggetti del mondo fisico da cui non ci si aspetta tale livello di asincronicitá.

Si propone quindi Routex come soluzione.

L'obiettivo del progetto é quello di permettere la creazione di una rete composta da dispositivi che fanno uso di tecnologie wireless e di protocolli di comunicazione che a priori risultano non compatibili fra loro, e per questi dispositivi fornire logiche di funzionamento e di interazione reciproca al fine di realizzare servizi utili.
Si fa uso di un Raspberry Pi come router del sistema. Esso supporta tutte le tecnlogie previste, gestisce le differenze di indirizzamento e comunicazione fra gli altri componenti e si occupa della raccolta, dell'analisi e della condivisione dei dati.
Routex rende possibile lo sviluppo di applicazioni IoT senza introdurre restrizioni sul tipo di utilizzo. Si lascia infatti la massima libertá nella scelta della configurazione hardware e software degli elementi della rete.
Routex vuole fare della semplicitá di impostazione e di utilizzo una caratteristica distintiva. Per questo motivo oltre al software del router sono state implementate librerie per diverse piattaforme, utili per la programmazione dei dispositivi, e una applicazione e un sito web tramite le quali stabilire il comportamento complessivo del sistema e interagire con le varie funzionalitá messe a disposizione dai nodi.

L'utente interagisce con i dispositivi in modo omogeneo, ossia senza doversi preoccupare delle differenze nelle tecnologie sottostanti, che sono gestite dal router.
Le tecnologie wireless supportate sono WiFi, Bluetooth, ZigBee, nRF24L01 e moduli radio 433MHz, ma l'architettura é stata progettata per rendere il sistema predisposto a eventuali estensioni, in modo da potersi adattare a nuovi protocolli richiedendo modifiche di ridotta entitá.
Infine Routex permette l'interoperabilitá con altri sistemi IoT e con servizi cloud di raccolta dati, con il fine di ampliare ulteriormente le proprie capacitá e per facilitare la condivisione delle informazioni.

Struttura dell'elaborato

Nel secondo capitolo si presentano i lavori correlati al progetto e le soluzioni proposte dalla letteratura e dall'industria alle problematiche dell'Internet of Things. Si introducono inoltre le tecnologie wireless supportate da Routex.

Nel terzo capitolo si analizzano l'architettura e l'implementazione di Routex, illustrando in particolare i componenti e le interazioni fra essi, il meccanismo di routing, le logiche applicative messe a disposizione e i metodi per interagire con il sistema.

Nel quarto capitolo si presentano le conclusioni ed eventuali lavori futuri riguardanti il progetto.

Infine, nell'appendice A si mostra un esempio di utilizzo di Routex, descrivendo nel dettaglio le operazioni di configurazione necessarie.

Capitolo 2

Lavori Correlati

Una soluzione tipicamente utilizzata per far fronte al problema dell'incompatibilitá é la realizzazione di un dispositivo, detto gateway o router per Internet of Things, in grado di implementare tutti i protocolli previsti dal sistema, che funge quindi da interfaccia fra la rete internet e la rete dei dispositivi.
Il nucleo del progetto Routex é appunto l'implementazione di un particolare gateway IoT.
Nonostante risulti essere un potenziale *single point of failure*, introduce i seguenti vantaggi [1]:

- Abilita funzionalitá di routing fra dispositivi che utilizzano protocolli di comunicazione differenti fra loro
- Permette di mantenere piú semplice il codice degli end device, gestendo la logica complessiva in un unico punto
- Garantisce che il sistema continui il proprio regolare funzionamento anche in caso di connessione Internet non disponibile
- É il singolo punto in cui implementare funzioni di interazione con eventuali servizi online di raccolta ed elaborazione dati

Si forma in questo modo quella che viene definita Wireless Sensor Network (WSN), ossia una rete di sensori coordinati da un nodo centrale che é inter-

facciato anche alla rete Internet.

Di seguito si analizzano le principali soluzioni alle problematiche dell'Internet of Things proposte sia in ambito commerciale che in quello accademico e scientifico.

Al momento, uno degli ambiti in cui trovano grande riscontro le applicazioni IoT é la domotica, per cui esistono numerosi prodotti commerciali.
La struttura di questi sistemi é generalmente composta da un hub centrale che implementa la comunicazione con appositi dispositivi (tipicamente usando ZigBee) e con cui l'utente interagisce tramite una specifica applicazione per smartphone.
Sono caratterizzati dalla facilitá di utilizzo e di inserimento di nuovi dispositivi nella rete. Tuttavia le funzionalitá che mettono a disposizione sono limitate dal tipo di sensori e attuatori che vengono fabbricati per essi. Inoltre utilizzano protocolli proprietari e soltanto alcuni forniscono API per potersi interfacciare da un sistema diverso, risultando spesso incompatibili fra loro. Non va trascurato infine il costo di ciascun singolo componente.
Da essi Routex riprende la possibilitá di impostare il funzionamento in modo intuitivo da applicazione e da browser e, per quanto possibile, la facilitá di configurazione dei nuovi dispositivi.

OpenHAB[5] é un software di automazione domestica completamente open source che permette l'integrazione di differenti sistemi e tecnologie, ed é funzionante su un qualsiasi dispositivo su cui sia implementata una Java Virtual Machine.
É strutturato a moduli componibili che aggiungono il supporto per i vari sistemi con cui permette di interagire e ha un proprio engine per lo sviluppo di regole anche molto sofisticate, scritte con un apposito linguaggio di scripting. Richiede peró un processo di configurazione piuttosto complesso.

Dall'analisi di diversi articoli si sono determinate altre caratteristiche che si desiderano da un sistema IoT.
[6] espone i principali vantaggi di realizzare un hub IoT tramite Raspberry Pi su cui si esegue un sistema operativo basato sul kernel Linux, fra cui il poter utilizzare i numerosi pacchetti software messi a disposizione e il poter accedere al router in modo diretto anche da remoto tramite il protocollo SSH. Nel progetto in questione si implementano metodi di conversione per portare il protocollo IP fino agli end device, limitando peró le tecnologie supportabili a quelle contemplate da 6LoWPAN[1].

[7] propone un metodo di riconfigurazione dinamica delle tecnologie utilizzabili dal gateway, permettendo cambiamenti a runtime riguardanti i canali di comunicazione.
Anche il Raspberry Pi di Routex vuole essere facilmente espandibile per quanto riguarda le periferiche supportate, necessitando tuttavia di un riavvio del programma in seguito a tali modifiche.
La dinamicitá del sistema realizzato riguarda in modo maggiore il software e si riscontra nel permettere disparati utilizzi a seconda del tipo dei dispositivi con cui il router interagisce, non limitandolo a una prefissata applicazione.

In [9] viene descritta una possibile soluzione per il problema dell'eterogeneitá dei dati ottenuti da differenti sistemi IoT. L'articolo presenta infatti un'architettura in grado di integrare dati forniti da piattaforme open source online per la raccolta e l'analisi (fra cui ThingSpeak), utilizzabili poi per la realizzazione di nuovi servizi.
Dal lavoro appena citato si deduce l'importanza che ha l'interfacciamento con i servizi cloud al fine di garantire una maggiore interoperabilitá.

Nello sviluppo del progetto si vogliono infine rispettare i principi esposti da

[1] IPv6 over Low-Power Wireless Personal Area Networks, standard di incapsulamento e compressione di pacchetti IPv6 per canali IEEE 802.15.4

[9] riguardo a quelle che dovrebbero essere le responsabilitá di un tipico gateway IoT, ossia: traduzione fra protocolli, controllo dello stato del sistema, raccolta delle informazioni, indirizzamento degli end device e autenticazione. Inoltre suggerisce di rispettare i principi REST[2] per favorire interoperabilitá con eventuali altri sistemi.
Routex implementa ciascuno degli aspetti proposti, ad eccezione dell'autenticazione, che puó rientrare nei possibili lavori futuri.

2.1 Tecnologie Wireless

Uno dei componenti principali dell'Internet of Things sono le tecnologie wireless, che, generalmente utilizzando come mezzo di trasmissione le onde radio, permettono la comunicazione a distanza fra i dispositivi e di conseguenza la creazione di una rete.
Si distinugono fra loro per caratteristiche come l'affidabilitá, il raggio d'azione e la capacitá e la velocitá di trasmissione dati, senza trascurare anche i consumi energetici richiesti per il funzionamento, che specialmente in applicazioni IoT diventano un elemento fondamentale, in quanto ci si aspetta che la maggior parte dei componenti sia alimentata a batteria.
Le tecnologie supportate da Routex forniscono una buona varietá relativamente a questi parametri, al fine di poter soddisfare differenti necessitá. La scelta di quale si presta maggiormente a un determinato utilizzo verrá effettuata valutando ad esempio la distanza dell'end device dal router, il tipo di alimentazione disponibile, quanto si é tolleranti a errori nella trasmissione o la prevista frequenza delle comunicazioni.
Il router del sistema si occupa di gestire le differenze nell'indirizzamento e nello stabilire le connessioni, rendendo il tutto uniforme dal punto di vista dell'utente ma senza annullare vantaggi e svantaggi delle tecnologie stesse.

[2]REpresentational State Transfer, modello di architettura software che si basa principalmente sull'utilizzo di risorse in formato strutturato e autodescrittvo, accedute tramite i metodi del protocollo HTTP.

Le seguenti sono le tecnologie con cui Routex é in grado di interagire.

WiFi

Si basa sullo standard IEEE 802.11, opera sulla banda ISM a 2.4GHz, permette elevate velocitá di trasmissione dati (anche oltre 100MBps) e ha un range che puó arrivare fino a 50-100m.
É stato progettato con lo scopo principale di essere il livello di trasporto di reti che usano lo stack TCP/IP, e ció implica la necessitá di un software complesso per gestirlo.
La topologia tipica di una rete WiFi é a stella, con l'access point come nodo centrale.
É la tecnologia piú diffusa e fornisce accesso diretto ad internet a ciascun dispositivo connesso, ma senza accorgimenti é molto dispendiosa dal punto di vista del consumo energetico.

Bluetooth

Opera sulla banda ISM a 2.4GHz ed é tipicamente usato per connessioni a brevi distanze, cioé circa 10m, nonostante da specifiche possa raggiungere anche 100m.
Lo standard Bluetooth definisce un proprio stack di protocolli e richiede, per motivi di sicurezza, che fra due dispositivi avvenga un accoppiamento prima di poter stabilire una connessione.
La topologia tipica di rete é punto a punto o a stella.
La versione 4.0 introduce il Bluetooth Low Energy, che presenta le caratteristiche migliori per essere utilizzato in un ambiente IoT, in quanto ha un data rate di 1Mbps che peró comporta una riduzione dei consumi e annulla il limite massimo di 8 dispositivi connessi presente nelle precedenti versioni.

ZigBee

Suite di protocolli che si basano sullo standard IEEE 802.15.4 (che definisce caratteristiche del livello fisico e MAC). Opera a 2.4GHz, con un data rate fino a 250kbps e con un range massimo di 100m.
La topologia tipica é la rete mesh, e garantisce alta sicurezza e scalabilitá. I dispositivi in una rete ZigBee possono ricoprire uno dei seguenti tre ruoli:

- Coordinator: esattamente uno per ogni rete, é l'origine, mantiene le informazioni principali della rete;
- Router: puó inoltrare dati ad altri dispositivi, fungendo da intermediario;
- End Device: puó comunicare solamente con il proprio nodo genitore ed ha la possibilitá di entrare in modalitá sleep.

ZigBee fa dei bassi consumi e del basso costo dell'hardware i suoi punti di forza.

Moduli nRF24L01

Modulo ricetrasmettitore low-cost che opera a 2.4GHz. Ha un data rate di 1Mbps o 2Mbps e permette comunicazioni fino a una distanza di circa 100m.
É configurabile tramite interfaccia SPI, e ció lo rende facilmente utilizzabile con Raspberry Pi e Arduino.
Si basa su Enhanced Shockburst, ossia un protocollo a livello di trasporto basato su pacchetti che garantisce acknowledgement e ritrasmissione automatica in caso di errore. Hanno bassi consumi, ma la dimensione dei pacchetti é limitata a un massimo di 32B di payload.

Moduli Radio 433MHz

Semplici moduli di trasmissione e ricezione a 433MHz. Hanno una massima velocitá di trasmissione di 4800bps, con un range che varia fra i 10m e

i 100m, a seconda dell'alimentazione.
Hanno il vantaggio di essere acquistabili a prezzi molto bassi, ma a differenza delle tecnologie descritte in precedenza non forniscono indirizzamento, controllo degli errori e gestione delle collisoni, che, se richieste, dovranno essere implementate dall'utilizzatore.
Comportano consumi molto bassi, ma sono facilmente soggette a interferenze.

Tecnologia	**Banda**	**Range**	**Data Rate**	**Consumi**
WiFi	2.4 GHz	50-100 m	$\approx$ 50Mbps	Elevati
Bluetooth LE	2.4 GHz	10 m (100 m)	1Mbps	Bassi
ZigBee	2.4 GHz	100 m	250 Kbps	Molto bassi
nRF24L01	2.4 GHz	100 m	2 Mbps	Molto bassi
RF 433	433 MHz	10-100 m	4800 bps	Molto bassi

Tabella 2.1: Riassunto delle caratteristiche delle tecnologie [11][12]

Capitolo 3

Architettura

In un ambito come l'Internet of Things, caratterizzato dall'eterogeneitá di dispositivi e di tecnologie wireless e da sistemi spesso non interoperabili fra loro, Routex nasce con lo scopo di rendere possibili interazioni fra dispositivi che utilizzano standard di comunicazione differenti e di fornire un metodo uniforme per la gestione della rete che essi formano.
Il progetto vuole fare della semplicitá di configurazione e dell'immediatezza nell'utilizzo le proprie caratteristiche distintive, senza peró limitare i contesti in cui puó essere utilizzato e lasciando all'utente la libertá di definire le proprie applicazioni sfruttando l'infrastruttura che viene messa a disposizione.
Sono stati proprio questi i principi che hanno guidato la progettazione e l'implementazione di tutti gli aspetti della struttura di Routex.
Ció che si ottiene é un sistema realizzato di base con il supporto per le tecnologie di comunicazione e per i dispositivi piú utilizzati, e che implementa diverse logiche applicative, ma che risulta altrettanto predisposto all'espansione di ognuno di questi settori.

3.1 Componenti

Per comprendere nel dettaglio il funzionamento di Routex si analizzano di seguito i componenti logici che costituiscono l'intero sistema.

Router: É il componente cardine del sistema in quanto si occupa di tutti gli aspetti principali. É in grado di comunicare utilizzando ogni tecnologia prevista, e su ognuna di esse rimane in attesa di comandi o di nuovi device che si uniscono alla rete. Effettua il binding fra il nome di ciascun device e il suo indirizzo per le funzioni di routing e mantiene lo storico dei dati di ciascun servizio. Funge inoltre da interfaccia con la rete internet esterna e implementa il server web con cui il controller interagisce.

É realizzato utilizzando un Raspberry Pi collegato alle varie periferiche di comunicazione necessarie.

Device: Rappresenta uno dei dispositivi in comunicazione con il router. É identificato in modo univoco all'interno del sistema da un nome, ossia una stringa di testo definita dal device stesso. Dichiara i propri servizi ed é in grado di iniziare una connessione con il router per inviare nuovi dati o di reagire a espliciti comandi ricevuti.

Puó essere sempre attivo, oppure impostato in modo tale che sia per intervalli di tempo in modalitá sleep, e che si ricolleghi al sistema solo periodicamente.

Puó essere realizzato con componenti fisici come Arduino, ESP8266 o smartphone ma é possibile far funzionare come un device anche un processo in esecuzione sul Raspberry Pi o su un eventuale altro computer (vedi sezione 3.6.2 per i dettagli).

Servizio: Ció che mette a disposizione un dispositivo, ossia i suoi sensori e attuatori, ciascuno con il proprio insieme di comandi. Sono anch'essi identificati da un nome, che deve essere univoco all'interno del singolo device. Ad esempio sono considerati servizi un sensore di temperatura o un led collegati a un Arduino, o la percentuale di batteria residua comunicata da uno smartphone.

Controller: É il componente di Routex che permette di interfacciarsi con il sistema. Attraverso il controller l'utente é in grado di verificare lo

stato dei dispositivi, impartire comandi, analizzare i valori raccolti dai sensori e impostare logiche di funzionamento.
Fungono da controller l'applicazione per smartphone e il sito web.

3.2 Interazioni fra Componenti

Il sistema é stato realizzato ammettendo due tipi di interazione fra componenti: quelle Router-Device e quelle Router-Controller.
Ció significa quindi che l'utente, tramite il controller, deve interfacciarsi sempre con il router per qualsiasi richiesta e che se un dipositivo deve reagire in base al valore raccolto da un diverso sensore, richiederá quel dato al router.
Sarebbe possibile realizzare un collegamento diretto fra device che sfruttano la stessa tecnologia oppure permettere all'applicazione del controller di comunicare senza intermediari con i nodi di rete che usano WiFi e Bluetooth (dato che praticamente ogni smartphone ne sarebbe in grado), migliorando anche le prestazioni del sistema dato che si ridurrebbe il traffico al router.
Tuttavia si é scelto di non implementare questa funzionalitá per questioni di semplicitá e uniformitá dell'architettura, decidendo di lasciare i compiti di routing esclusivamente al Raspberry Pi, che é quindi anche in grado di tracciare ogni singolo scambio di messaggi nella rete.

3.2.1 Interazioni Router-Device

Ogni comunicazione Router-Device avviene usando un semplice protocollo, ideato appositamente per Routex con lo scopo di fornire un metodo uniforme utilizzabile in modo efficiente su tutte le tecnologie wireless supportate.
Il protocollo si basa su stringhe testuali intervallate da caratteri separatori ed ogni messaggio ha una dimensione massima di 28 byte (limitazione imposta dai moduli radio nRF24L01). Per fornire maggiore affidabilitá al sistema si utilizza un meccanismo di acknowledgement, per cui il mittente rimane in

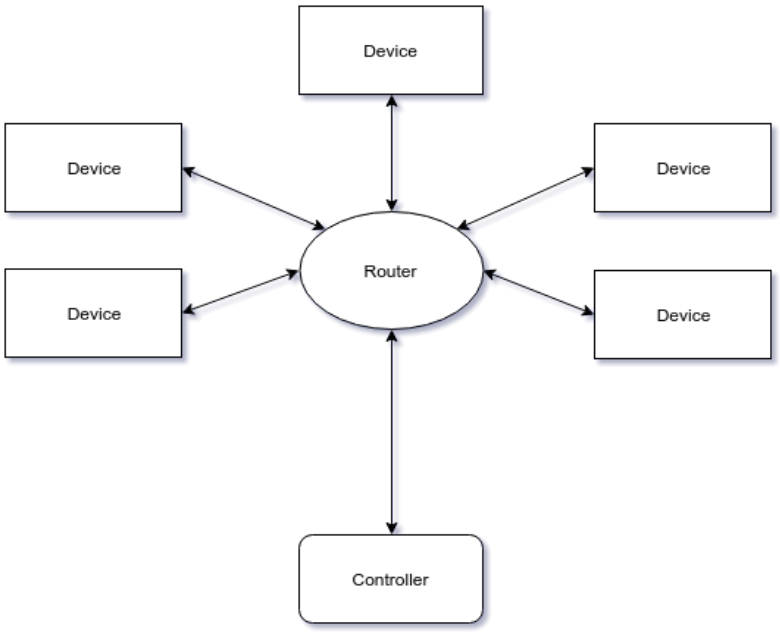


Figura 3.1: Struttura Generale di Routex

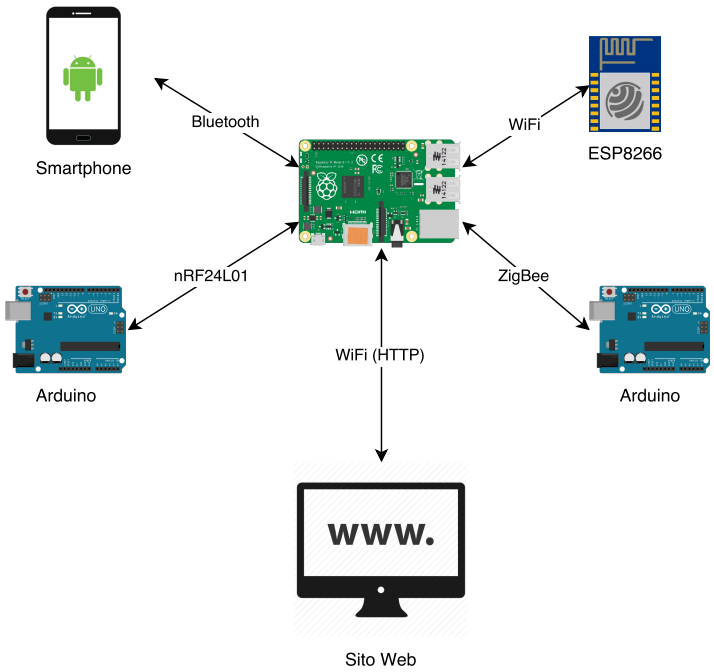


Figura 3.2: Un possibile esempio di rete

attesa di un ACK per un tempo massimo di 5 secondi dopo l'invio, scaduti i quali ritrasmette il messaggio originale.

Questa tipologia di interazione é alla base delle seguenti operazione fondamentali della rete:

Join di un device

I device sono programmati in modo tale che effettuino come prima operazione l'invio di un messaggio di join, dichiarando il prorio nome identificativo e la modalitá di funzionamento, ossia se rimane sempre attivo o se puó entrare in modalitá sleep. Il router, che é sempre in ascolto su tutti i canali di comunicazione ed ha per ciascuno un indirizzo noto ai device, effettua un binding fra il nome del dispositivo, la tecnologia utilizzata e l'indirizzo da cui riceve la richiesta. In seguito il dispositivo comunica i propri servizi, e per ciascuno di essi i comandi disponibili.

Si é scelto di non implementare nativamente in Routex un meccanismo automatico di gestione dell'uscita dalla rete, ma deve essere l'utente a richiederlo esplicitamente. Questo per evitare che momentanee ostruzioni del segnale causino erroneamente l'esclusione di un nodo. É tuttavia semplice realizzare un servizio che abbia unicamente lo scopo di verificare periodicamente la presenza e notificare in caso di mancata risposta.

Richiesta di esecuzione di un comando

É un tipo di messaggio che contiene il nome identificativo di un servizio e un comando, ed é inviato dal router a un dispositivo per richiedere che venga eseguita l'azione associata al comando specificato. Il comportamento di reazione al comando e il contenuto della risposta sono definiti dal codice del dispositivo stesso.
Questa richiesta puó essere originata dai seguenti eventi:

- Verificarsi delle condizioni di un trigger che ha come azione conseguente l'esecuzione di un comando un device

- Raggiungimento di un orario per cui era programmata l'esecuzione di un comando di un device

- Richiesta esplicita da parte dell'utente, e in questo caso la risposta viene inoltrata anche al controller con cui é stata effettuata la richiesta

Aggiornamento di valori di un servizio

Comunicazione che ha lo scopo di informare il router di un cambiamento di stato o della presenza di un nuovo dato significativo per un servizio. Il messaggio specifica il servizio coinvolto e il relativo valore.
Seppur portando lo stesso risultato ai fini del sistema, si distingue dal caso precendente in quanto é il device che inizia la connessione e non il router. É quindi ideale per tutti i sensori che vogliono implementare un meccanismo di notifica istantanea e per tutte le azioni che avvengono periodicamente ma non a intervalli regolari.

Interazioni relative a un device non sempre attivo

Un caso importante da tenere in considerazione é quello che riguarda i dispositivi che funzionano in modalitá non sempre attiva. Sono nodi che si collegano alla rete soltanto periodicamente, pertanto non é possibile gestire le interazioni con essi senza appositi accorgimenti.
Il router é a conoscenza di quali device possono entrare in modalitá sleep, dato che viene dichiarato durante la join.
Ogni comando (anche programmato, o generato da un trigger) che li coinvolge viene salvato su un file dal Raspberry Pi. Quando termina il periodo di sleep il device puó segnalare al router lo stato di attivitá con un apposito messaggio, e solo allora gli eventuali comandi salvati sul file saranno effettivamente inviati e gestiti come nei casi precedenti.

3.2.2 Interazioni Router-Controller

Il software del router implementa un server web, che si occupa della gestione di tutte le interazioni con il controller e quindi con l'utente.
In questa categoria rientrano ad esempio le richieste di inserimento di un nuovo trigger o l'esplicita invocazione di un comando, ma anche le comunicazioni che forniscono all'utente l'elenco dei dispositivi connessi o lo storico dei valori raccolti da un sensore.
Per un elenco dettagliato delle funzionalitá messe a disposizione dal controller si faccia riferimento alla sezione 3.5
Le richieste sono effettuate utilizzando il protocollo HTTP, con i dati strutturati in formato JSON. La scelta é dovuta alla semplicitá con cui si riesce a trattare questo formato di dati lato server, grazie alle librerie disponibili per Python. In piú si ha che l'unica configurazione necessaria per l'utente risulta quella di specificare nelle impostazioni del controller l'indirizzo IP del router e la porta su cui é in ascolto il server web.
Inoltre é possibile interagire con il sistema anche al di fuori della rete locale se il Raspberry Pi risulta visibile. Cosí facendo si rischierebbe peró di introdurre problemi relativi alla sicurezza, e per evitarlo senza rinunciare a controllare lo stato del sistema da remoto é stato introdotto il supporto a ThingSpeak (Sezione 3.4.1)

```
{
  "device_name": "Arduino-1",
  "service_name": "RedLed",
  "command" : "Turn On"
}
```

```
{
  "device_name": "Arduino-1",
  "service_name": "RedLed",
  "result": 1,
```

```
  "timestamp" : 1473370657094
}
```

Esempio 3.1: Contenuto di richiesta e risposta HTTP per l'esecuzione di un comando

3.3 Logiche

Rientrano nelle logiche tutte quelle funzionalitá messe a disposizione da Routex che non riguardano direttamente il routing multitecnologia, ma definiscono invece il comportamento dei nodi della rete.
Sono state inserite nel sistema con il fine di permettere la creazione di applicazioni che prevedono interazioni complesse fra i dispositivi senza dover constringere l'utente a scrivere il codice per farlo. Le regole che determinano queste interazioni sono infatti stabilite tramite il controller, riducendo al minimo la complessitá della programmazione dei device. Si richiede infatti che si stabilisca solamente in che modo reagire ai comandi che esso stesso mette a disposizione.
É il router che si occupa della logica, rendendo quindi non necessaria la costante presenza del controller in rete. Inoltre tutte le regole logiche sono gestite in modo dinamico, per ammettere l'inserimento e la cancellazione in qualsiasi momento e con effetto immediato.

3.3.1 Schedule

Il primo tipo di regole logiche ammesso da Routex sono quelle che consentono la programmazione di eventi. Esse permettono il verificarsi di un'azione in un istante temporale futuro.
Le azioni ammissibili sono tutti i comandi messi a disposizione dai dispositivi appartenenti alla rete e possono essere pianificati per essere eseguiti ad un preciso orario oppure in modo periodico, a partire dall'istante in cui si crea la regola e con intervallo di tempo arbitrario.

Gli elementi della schedule sono identificati da un numero progressivo, per ridurre la complessitá delle operazioni di modifica e rimozione e sono salvati dal Raspberry Pi su un file in formato JSON, per poterli inviare facilmente quando richiesti dal controller.

3.3.2 Trigger

La seconda categoria riguarda i trigger, ossia azioni scatenate solamente quando sono verificate determinate circostanze.
In questo caso le azioni possibili prevedono anche, oltre alla richiesta di esecuzione di comandi dei device, l'invio di mail e di notifiche al controller. L'utente é libero di specificare il proprio indirizzo mail e il contenuto dei messaggi che riceverá durante la creazione del trigger. Il meccanismo di gestione delle notifiche utilizza Google Cloud Messaging, ed é utilizzabile solamente quando il controller é l'applicazione Android.
L'esecuzione dell'evento é resa conseguenza della soddisfazione di una o piú condizioni e la singola condizione é il confronto dell'ultimo valore disponibile di un servizio con un valore fissato dall'utente (ad esempio se la temperatura rilevata da un Arduino supera i 25 gradi). Inseririe condizioni multiple equivale ad effettuare un AND logico fra esse, ossia si richiede che siano verifiate tutte. L'OR risulta comunque realizzabile inserendo piú trigger.
Il controllo sui requisiti avviene ogniqualvolta il router riceve un dato da uno dei dispositivi in rete e la verifica é effettuata su tutti i trigger che hanno fra i fattori scatenanti il servizio di cui si é aggiornato il valore.
Come per gli elementi della schedule, anche i trigger sono salvati su file JSON e univocamente identificati da un intero.

3.4 Gestione dei Dati

I dati utili generati dall'intero sistema sono conservati e gestiti dal router. I dati in questione possono essere di tipo numerico o stringhe di testo e sono i

valori raccolti e comunicati dai dispositivi in esecuzione. A ciascuno di essi é associato un timestamp che indica, in millisecondi dal 1/1/1970, l'istante in cui é stato originato, formando in questo modo uno storico di tutti i servizi. Sono strutturati in formato JSON e salvati su un file sul Raspberry Pi.

Il router accede ed effettua una analisi sui dati nei seguenti casi:

- Il controller richiede al router l'ultimo valore disponibile di un servizio, che viene determinato in base al timestamp, per farlo visualizzare all'utente.
- Un dispositivo invia l'aggiornamento del valore di un servizio. In questo caso il nuovo valore viene salvato e se il servizio coinvolto risulta facente parte di un trigger si accedono anche gli ultimi valori relativi alle altre condizioni in gioco per gli opportuni controlli.
- L'utente vuole visualizzare il grafico di un servizio relativo a un certo intervallo di tempo. Il router deve quindi fornire al controller tutti i valori che hanno un timestamp compreso fra l'istante di inizio e di fine specificato dall'utente.


```
[{"service_name": "Temperature",
  "device_name": "Arduino-1",
  "values": [
      {
       "timestamp": "1471940745603",
       "value": 22
      },
      {
       "timestamp": "1471940768929",
       "value": 22
      },
      {
```

```
        "timestamp": "1472146146185",
        "value": 21
       }
   ]
}]
```

Esempio 3.2: Esempio del formato di memorizzazione dati

3.4.1 ThingSpeak

ThingSpeak é una piattoforma per Internet of Things che permette di raccogliere e salvare sul cloud dati raccolti da sensori e favorisce lo sviluppo di applicazioni IoT [13]. É strutturato a canali, con cui si interagisce tramite richeste HTTP per la gestione dei dati, e possono essere resi pubblici o privati.
In Routex é stato inserito il supporto all'interazione con ThingSpeak principalmente per fornire funzioni di monitoraggio in tempo reale del sistema anche da remoto, senza quindi dovere rendere visibile il Raspberry Pi dalla rete esterna.
Ma questo non é l'unico utilizzo possibile. Puó risultare anche molto utile per fornire i dati ad altri servizi in grado di interfacciarsi a ThingSpeak, oppure come servizio di backup per dati rilevanti.
Per poter usufruire delle funzionalitá di ThingSpeak, un utente di Routex deve essere registrato al servizio e inserire la propria API key utente nel corrispondente campo del file di configurazione di Routex sul Raspberry Pi.
Con il precedente requisito soddisfatto, risulta sufficiente abilitare dal controller il salvataggio dei dati su ThingSpeak. É possibile attivare o disattivare l'interazione con il cloud in qualsiasi momento, e in modo separato per ogni servizio.
Routex crea in modo automatico il canale ThingSpeak associato a un servizio la prima volta che se ne attivano le funzionalitá. Dopodiché, fino a una eventuale disattivazione, tutti i valori ricevuti riguardanti quel servizio

saranno inoltrati al canale.

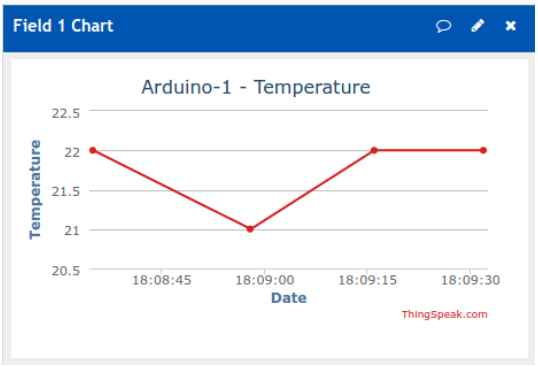


Figura 3.3: Grafico di ThingSpeak con i dati di un dispositivo di Routex

3.5 Front-End

La parte front-end del sistema é costituita dal controller. Esso é infatti l'unico componente tramite il quale l'utente interagisce con Routex e ne sfrutta le funzionalitá.
Il controller é stato realizzato in modo tale che rifletta nella propria interfaccia grafica la struttura interna del sistema stesso, e con lo scopo di fornire un metodo comodo e rapido per visualizzare lo stato del sistema e i valori piú recenti di ciascun servizio.
L'architettura e il modello delle interazioni di Routex permettono inoltre al controller di non essere in ogni istante connesso alla rete, rendendolo non necessario al corretto funzionamento del sistema una volta che si sono stabilite le regole logiche.
Inizialmente é stato implementato come applicazione Android e in seguito anche come sito web, per ampliare i metodi di utilizzo e rendendo non necessario il possesso di un dispositvo Android. In entrambi i casi l'interazione avviene con il server web gestito dal Raspberry Pi, tramite richieste HTTP e dati in formato JSON.

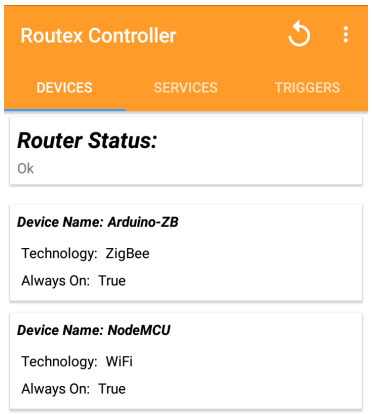
3.5.1 Interfaccia del Controller

Si analizza ora nel dettaglio l'interfaccia del controller e come usufruire di tutte le funzionalitá che mette a disposizione facendo riferimento all'applicazione. Si consideri che la struttura dell'applicazione rispecchia esattamente anche quella del sito.

La schermata principale utilizza un layout a tab per consentire di accedere rapidamente alle informazioni rienute principali, ossia dispositivi, servizi e trigger, e da essa inizia la navigazione.

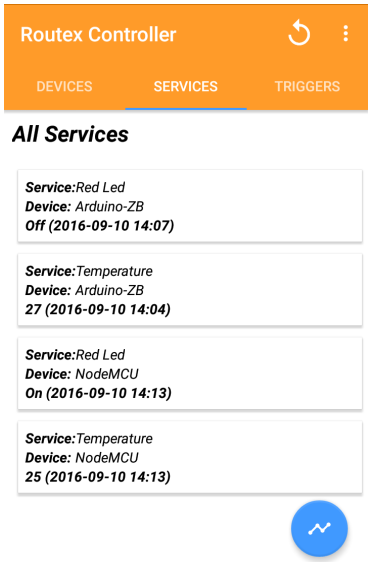
Lista dei Device

É la schermata che viene visualizzata all'avvio dell'applicazione.
Fornisce l'elenco dei dispositivi connessi al momento alla rete IoT e per ciascuno di essi mostra il nome, la tecnologia utilizzata e la modalitá di funzionamento.
Da qui l'utente puó vedere i servizi messi a disposizione da un dispositivo cliccando l'elemento di interesse, oppure richiedere l'uscita dalla rete tenendo premuto sul device che si vuole rimuovere.



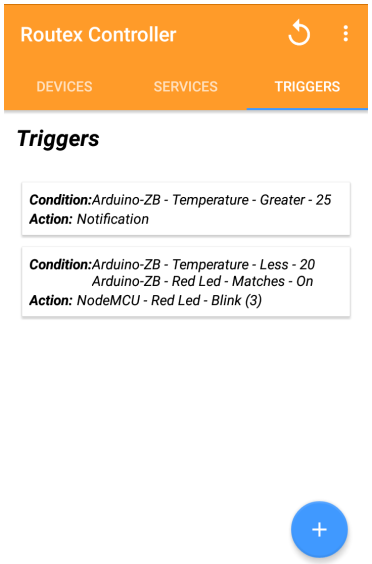
Lista dei Service

Permette di scorrere l'elenco di tutti i servizi messi a disposizione dai dispositivi in rete, specificando per ciascuno di essi l'ultimo valore rilevato (se presente) con relativo istante di tempo.
Il click su uno degli elementi della lista porta la navigazione a un nuovo layout a tab, che permette l'interazione con i comandi, la schedule e il grafico del servizio corrispondente.
É possibile inoltre visualizzare un grafico che mostri l'andamento di piú servizi contemporaneamente tramite l'apposito bottone.



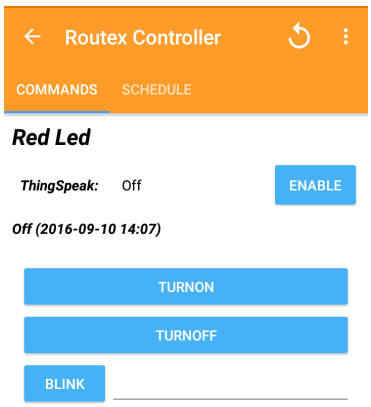
Trigger

É la sezione che mostra l'elenco di tutti i trigger del sistema, descrivendo per ciascuno di essi l'elenco delle condizioni da verificare e l'azione conseguente (notifica, mail o comando).
Tenendo premuto su uno dei trigger si effettua la cancellazione, mentre la creazione di nuovi avviene tramite un dialog che si apre tramite click sull'apposito bottone.



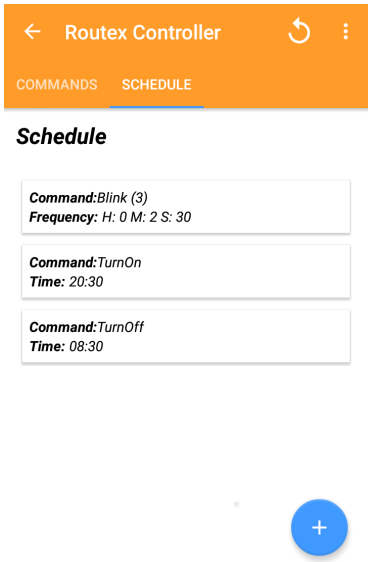
Comandi

Interfaccia composta da un elenco di bottoni corrispodnenti ai comandi del servizio, una label di testo che mostra l'ultimo valore disponibile e il bottone per la gestione dei servizi di ThingSpeak.
I comandi possono essere associati a una casella di testo se richiedono l'inserimento di un parametro. Il click su un bottone genera la richiesta di esecuzione del comando, che viene subito inviata al device tramite il router. Se il processo va a buon fine la label di testo viene aggiornata con il valore restituito dal dispositivo.



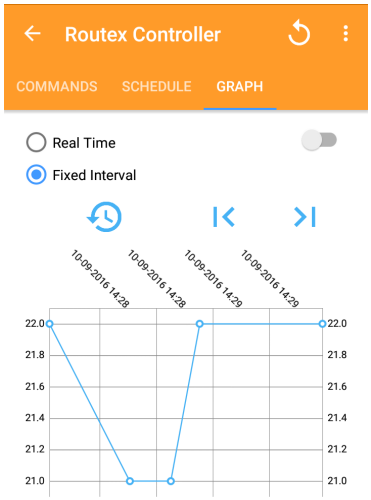
Schedule

Elenca tutti gli eventi programmati che riguardano il servizio selezionato, specificando per ciascuno il comando che verrá eseguito, un eventuale parametro associato al comando, la frequenza o l'orario.
Tramite click sull'apposito bottone verrá aperto il dialog per l'aggiunta di nuovi eventi alla schedule, e la cancellazione si effettua tendo premuto sull'elemento che si desidera rimuovere.



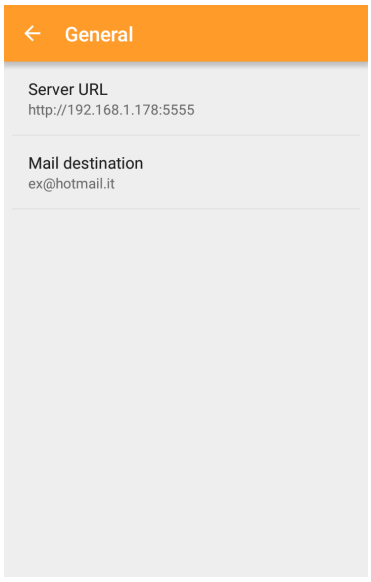
Grafico

Questa sezione consente di visualizzarne in un grafico l'andamento nel tempo dei valori relativi al servizio selezionato, ed é disponibile esclusivamente se i valori stessi sono di tipo numerico.
É possibile definire un intervallo temporale di interesse oppure lasciare che il grafico venga aggiornato in tempo reale. Quest'ultima modalitá é implementata effettuando in modo automatico richieste periodiche al router a cui segue una risposta contentente eventuali dati piú recenti.



Impostazioni

É la pagina in cui si impostano indirizzo e porta del server web del Raspberry Pi e l'indirizzo e-mail a cui saranno inviate eventuali comunicazioni in seguito al verificarsi delle condizioni di un trigger.
Le impostazioni possono essere accedute e modificate in qualunque momento.
I cambiamenti all'indirizzo del router hanno effetto immediato, mentre quelle all'indirizzo mail influenzano solo i trigger definiti successivamente, e non quelli giá esistenti.



3.6 Implementazione

3.6.1 Router

Per quanto riguarda l'hardware, si é scelto di realizzare il router usando un Raspberry Pi, a cui sono stati collegati i moduli necessari ad aggiungere il supporto alle tecnologie wireless.
Il Raspberry Pi garantisce, a un prezzo accessibile e con ridotti consumi energetici, prestazioni ampiamente sufficienti a svolgere tutti compiti che si richiedono al componente principale di Routex. Inoltre la grande diffusione e l'ampia community rappresentano altri importanti punti a favore del suo utilizzo.
Nel dettaglio, si é realizzato il router e testato il suo funzionamento con un Raspberry Pi modello B+ con sistema operativo Raspbian, insieme a dongle Wifi e Bluetooth, modulo XBee con relativo adattatore USB e moduli nRF24L01 e radio a 433MHz interfacciati tramite i GPIO.

La parte software é stata scritta usando Python 2.7 come unico linguaggio di programmazione, in quanto caratterizzato dalla semplicitá di realizzazione e di lettura del codice. Altri aspetti fondamentali che sono stati tenuti in considerazione sono i numerosi moduli che mette a disposizione per la gestione delle tecnologie wireless e dei GPIO del Raspberry Pi.
La struttura del software si compone dei seguenti tre moduli:

routex.py : É il modulo principale del software del router. In esso sono presenti le primitive per l'invio e la ricezione sui vari canali di comunicazione e le funzioni necessarie al routing. Contiene anche il codice responsabile della gestione delle logiche e dell'analisi dei file contenti i dati del sistema.

flaskServer.py : Implementa il server web utilizzando `flask`. Si occupa quindi dell'interazione con il controller, ma ottiene i dati tramite chiamate a funzioni del modulo `routex.py`

thingSpeak.py : Implementa le funzionalitá che richiedono interazione con il servizio ThingSpeak. In particolare mette a disposizione procedure per la creazione di canali e per l'aggiunta di dati a uno di essi.

Il software permette anche all'utente l'utilizzo di un sottoinsieme delle tecnologie supportate, specificabili tramite argomenti a riga di comando, per evitare al router di rimanere sempre in attesa su canali di comunicazione che a priori si é consapevoli di non voler usare.

Tutti i dati relativi al sistema sono salvati sul Raspberry Pi su file in formato JSON.
A ciascun file é associato un lock di Python, per garantire che vengano acceduti in mutua esclusione, evitando problemi di concorrenza, dato che il software del router utilizza molteplici thread.
Il file `routex_config.json` funge da file di configurazione del sistema e serve per la personalizzazione di alcuni parametri di Routex, fra cui l'account per l'invio delle mail, l'UUID del servizio Bluetooth e l'API key utente di ThingSpeak.

3.6.2 Device

A differenza del caso del router, le caratteristiche desiderabili da un end device dipendono molto dal tipo di applicazione che si vuole costruire basandosi sulla rete IoT e per questo motivo Routex fornisce software destinato a differenti piattaforme.
Ció che viene messo a diposizione non sono programmi che definiscono il comportamento complessivo di un dispositivo, ma librerie di codice che implementano le procedure di interazione con il router. Questo avviene per lasciare massima libertá all'utente nella programmazione dei propri nodi di rete, senza dover entrare nei dettagli del protocollo di comunicazione usato in Routex.

Nel codice di un device é quindi necessario inserire le chiamate per effettuare il join al sistema, specificando il nome, i servizi messi a disposizione e i comandi per ciascuno di essi. In seguito é possibile definire una logica che il dispositivo deve seguire (indicato per applicazioni complesse), oppure soltanto come deve reagire a ogni singolo comando che ha dichiarato e in questo secondo caso il comportamento sará dettato dalle regole logiche impartite tramite il controller.
Se si desidera inoltre estendere il supporto a un tipo di dispositivo che non rientra fra quelli contemplati da queste librerie, quello che risulta necessario fare é implementare il protocollo per le interazioni fra i dispositivi e il router (Sezione 3.2.1).
Di seguito si descrivono le caratteristiche specifiche per ciascun tipo di dispositivo supportato.

Arduino e ESP8266

Arduino e ESP8266 sono fra i componenti hardware piú comuni per la realizzazione di sensori e attuatori per Internet of Things, perció si é deciso di implementare una libreria per favorire la programmazione di questi dispositivi.
Essa permette l'utilizzo di moduli XBee, nRF24L01, radio 433MHz, e WiFi nella versione per ESP8266, in modo uniforme, ossia senza distinzioni nell'interfaccia dopo aver dichiarato quale si vuole usare ed eventuali informazioni correlate (ad esempio SSID e password della rete WiFi).
Forniscono inoltre le chiamate per entrare e uscire dalla modalitá sleep.

Client Android

Applicazione Android che consente di collegare smartphone e tablet alla rete IoT utilizzando il Bluetooth.
Grazie alle socket Bluetooth di Android effettua in modo automatico una scansione dei dispositivi nel proprio raggio di azione, e tramite Service Di-

scovery Protocol (SDP) identifica il Raspberry Pi, che fa advertising di un servizio Bluetooth di cui é noto l'UUID. Se questa operazione avviene con successo si inizia la procedura di join.
L'applicazione permette di base la verifica della percentuale di batteria e il controllo della torcia, dichiarandoli come servizi Routex, ma sono facilmente espandibili modificando una opportuna sezione del codice.

Modulo Python

Viene importato come modulo in un programma Python e serve per creare un processo che simuli un dispositivo collegato alla rete.
Effettua le stesse interazioni con il router di un dispositivo fisico e utilizza lo stack TCP/IP per la comunicazione.
Il processo puó essere in esecuzione sullo stesso Raspberry Pi, per fornire all'utente informazioni riguardanti il suo stato, oppure su una qualsiasi altra macchina.
Puó essere sfruttato anche come metodo per interfacciarsi ad un altro sistema, associando all'esecuzione di un comando relativo a un servizio di Routex l'invocazione di funzioni di API di terze parti.

3.6.3 Estensione delle Tecnologie

Come giá specificato in precedenza, la struttura hardware e software del sistema é stata realizzata in modo tale che si presti facilmente ad una estensione delle tecnologie supportate. E' una caratteristica di fondamentale importanza per un router IoT, data la rapiditá con cui nuovi standard di comunicazione possono diffondersi e affermarsi.
Le operazioni da compiere per aggiungere un nuovo protocollo a quelli giá gestiti si riducono a collegare eventuali moduli fisici necessari al router ed effettuare semplici modifiche al software, dove risulta sufficiente estendere le primitive di invio e di ricezione dei messaggi.

Ovviamente lo stesso procedimento va effettuato anche sul codice dell'end device che si vuole aggiungere alla rete. In entrambi i casi comunque non é richiesta alcuna variazione alla struttura dei messaggi che il router e i dispositivi si scambiano per la sincronizzazione.

Capitolo 4

Conclusioni

In conclusione, si puó affermare che il sistema descritto rappresenta una soluzione al problema di rendere possibile la comunicazione fra sottoreti che utilizzano standard differenti.
La mancanza di una tecnologia comune é ovviata dalla presenza di un router che implementa tutte le tecnologie e funge da intermediario per ogni interazione fra i nodi della rete, occupandosi della gestione delle differenze. Grazie a questa struttura si garantisce inoltre la trasparenza dal punto di vista dell'utente, che si interfaccia in modo uniforme con i servizi messi a disposizione dai dispositivi.
Il formato dei messaggi che si scambiano router e dispositivi segue le norme stabilite da un un semplice procotollo definito appositamente per Routex, implementabile su tutte le tecnologie previste. Il protocollo ammette diverse modalitá di comunicazione, fra cui anche quelle necessarie per gestire il caso in cui un dipositivo si connetta solamente periodicamente per il risparmio di energia.
Il sistema fornisce quindi una infrastruttura che permette la creazione di una rete IoT su cui si possono implementare svariati servizi, grazie alle logiche applicative configurabili dinamicamente dall'utente.
La generalitá di utilizzo e la libertá di realizzare differenti applicazioni con Routex é ottenuta lasciando ai device il compito di dichiarare in fase di join

le proprie funzioni, permettendo quindi al router di gestire tutti i dispositivi in modo omogeneo e indipendente da quali attuatori e sensori mettono a disposizione.
Al fine di permettere questa generalitá é tuttavia necessario programmare appositamente i propri dispositivi, in modo che implementino il protocollo utilizzato per le comunicazioni con il router. Per ridurre le competenze richieste e facilitare lo sviluppo sono quindi fornite librerie che nasconodo la complessitá della configurazione.
Si é cercato di limitare al minimo la programmazione richiesta anche per la definizione di come deve agire un nodo della rete, ma per la realizzazione di logiche complesse é attualmente necessario specificare il comportamento nel codice del dispositvo stesso.
La pubblicazione di dati di interesse raccolti dai dispositivi su canali ThingSpeak permette di monitorare lo stato del sistema anche da reti esterne, senza dover esporre il sistema a problemi legati ad aspetti di sicurezza. Tuttavia non risulta possibile l'esplicita richiesta di esecuzione di comandi da remoto, se non precedentemente impostata come azione programmata o come reazione di un trigger.
Infine l'interoperabilitá con sistemi IoT giá presenti avviene tramite lo sviluppo di programmi che fungono da adattatori. Essi interagiscono con il router come un device ed associano ai propri comandi le operazioni messe a disposizione dalle API del sistema con cui si vuole interagire. Questi processi possono essere in esecuzione sul Raspberry Pi o su altri computer della rete.

4.1 Lavori Futuri

Si elencano di seguito possibili estensioni del progetto realizzato:

- Introdurre maggiori misure di sicurezza, come ad esempio implementare meccanismi di gestione di vari profili utente, ciascuno autorizzato a effettuare determinate operazioni e con autenticazione richiesta all'inizio dell'interazione con il sistema.

- Aumentare le tecnologie wireless supportate e i dispositivi per cui sono fornite le librerie. Nonostante l'estensione sia un processo relativamente semplice, ampliare il supporto in modo nativo renderebbe ancora piú immediato e completo il sistema.

- Migliorare il protocollo di comunicazione fra router e dispositivi. Le modifiche in questo ambito potrebbero riguardare sia l'ottimizzazione dell'implementazione che l'introduzione di nuove funzionalitá. Ad esempio permettere ai dispositivi di dichiarare i valori possibili dell'argomento di un comando, in modo tale da poterli mostrare in modo intuitivo all'utente tramite il controller.

- Estendere le logiche applicative del sistema, ampliando le condizioni che possono generare trigger, ad esempio analizzando piú valori di uno stesso servizio e non solamente l'ultimo. Inoltre per migliorare l'usabilitá si potrebbe introdurre l'OR logico fra gli elementi della lista delle condizioni, che allo stato attuale é realizzabile solo inserendo molteplici regole che fanno uso dell'AND.

Appendice A

Esempio di Configurazione e Utilizzo

In questo capitolo si mostra la sequenza di operazioni da eseguire per la configurazione del router e dei dispositivi necessari alla realizzazione di un'applicazione IoT utilizzando Routex.
L'applicazione che si vuole realizzare permetterá di rilevare periodicamente la temperatura e l'umiditá dell'ambiente tramite un dispositivo, pubblicando i dati raccolti su ThingSpeak e segnalando a un diverso dispositivo il superamento di un certo valore fissato.
É un esempio semplice, che prevede l'utilizzo di soli due end device, ma risulta utile per capire meglio il funzionamento e le potenzialitá del sistema.

A.1 Componenti

Si elencano di seguito i componenti utilizzati:

- Raspberry Pi modello B+, dongle WiFi e modulo XBee con relativo USB explorer per il router.
- NodeMCU e DHT11 per il dispositivo che si occuperá di misurare temperatura e umiditá.

- Arduino Uno con modulo XBee (e relativo shield) e un led per l'altro dispositivo.

A.2 Prerequisiti

Si assumono veri i seguenti requisiti, la cui configurazione non sará trattata in questo documento.

- Il Raspberry Pi utilizza il dongle WiFi per realizzare una rete in cui funge da Access Point ed accede alla rete Internet tramite la porta Ethernet (necessario solamente per l'interazione con ThingSpeak).
- Il modulo XBee del Raspberry Pi é configurato come Controller, mentre quello che verrá usato dall'Arduino puó essere Router o End Device, entrambi in API mode.
- Si é in possesso di un account ThingSpeak e della relativa API key utente.

A.3 Configurazione del Router

- Effettuare il download del software di Routex dal repository Github (`https://github.com/exmorse/Routex`).
- Installare tutte le dipendenze richieste, ossia i moduli Python specificati in `README.md`
- Modificare con i dati opportuni il file `routex_config.json`. Per questo esempio é sufficiente inserire la API key utente di ThingSpeak.

A.4 Configurazione dei Device

Si descrivono ora la configurazione dei dispositivi e il codice utilizzato per la realizzazione dell'esempio.

I programmi fanno uso delle librerie presenti nei file `ArduinoRoutex.zip` e `ESPRoutex.zip`, disponibili anch'esse dal repository Github.
In entrambi i casi lo scopo del codice é quello di dichiarare le capacitá dei dispositivi, senza definire una logica applicativa specifica, che sará definita tramite il controller in un secondo momento.
La struttura degli sketch é la seguente:

- Nel costruttore della classe `Routex` si specificano la tecnologia usata ed eventuali parametri relativi ad essa
- In `setup()` si dichiara il nome del dispositivo, i servizi messi a disposizione e i comandi per ciascuno di essi e si comunicano queste informazioni al router
- In `loop()` si rimane in ascolto per eventuali comandi e si definisce il relativo comportamento e il valore da restituire

NodeMCU

```
#include <ESPRoutex.h>
#include "DHT.h"

Routex R(Routex::WIFI, "TestNetwork", "TestPassword", true);
#define DHT11Pin D1
DHT dht(DHT11Pin, DHT11);

void setup() {
  Serial.begin(9600);
  Serial.println("Start Connecting...");
  if (!R.init(5100)) Serial.println("Init Failed");
  else Serial.println("Init Successful");
  if (R.registerDevice("NodeMCU") != -1)
      Serial.println("Registered Successfully");
  if (R.registerService("Temperature", Routex::NUMBER))
      Serial.println("Service Registered Successfully");
  R.doneService();
  if (R.registerService("Humidity", Routex::NUMBER))
      Serial.println("Service Registered Successfully");
```

```
20   R.doneService();
21 }
22
23 void loop() {
24   if (R.checkServiceRequest() == 1) {
25     String service = R.getRequestedServiceName();
26     String command = R.getRequestedServiceCommand();
27     if (service.equals("Temperature")) {
28       if (command.equals("Get")) {
29         int celsius = dht.readTemperature();
30         R.serviceResponse(String(celsius));
31       }
32     }
33     if (service.equals("Humidity")) {
34       if (command.equals("Get")) {
35         int humidity = dht.readHumidity();
36         R.serviceResponse(String(humidity));
37       }
38     }
39   }
40 }
```

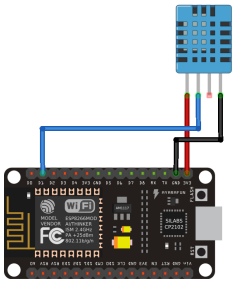


Figura A.1: Configurazione del NodeMCU

Arduino

```
1 #include <Routex.h>
2
3 int redPin = 13;
4 Routex R(Routex::ZIGBEE, 0, true);
5
```

```
6 void setup()
7 {
8   Serial.begin(9600);
9   pinMode(redPin, OUTPUT);
10  digitalWrite(redPin, LOW);
11  if (!R.init()) Serial.println("Init Failed");
12  else Serial.println("Init Successful");
13  if (R.registerDevice("Arduino-ZB") != -1)
14      Serial.println("Registered Successfully");
15  if (R.registerService("Led", Routex::STATUS))
16      Serial.println("Service Registered Successfully");
17  R.addCommandToService("TurnOn", Routex::NO_PARAM);
18  R.addCommandToService("TurnOff", Routex::NO_PARAM);
19  R.doneService();
20 }
21
22 void loop()
23 {
24  if (R.checkServiceRequest() == 1) {
25    String service = R.getRequestedServiceName();
26    String command = R.getRequestedServiceCommand();
27    if (service.equals("Led")) {
28      if (command.equals("TurnOn")) {
29        digitalWrite(redPin, HIGH);
30        R.serviceResponse("On");
31      }
32      if (command.equals("TurnOff")) {
33        digitalWrite(redPin, LOW);
34        R.serviceResponse("Off");
35      }
36    }
37  }
38 }
```

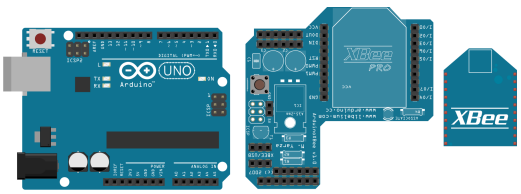


Figura A.2: Componenti del secondo dispositivo utilizzato

A.5 Esecuzione

- Sul Raspberry Pi, con i permessi di root, lanciare il comando `./routex.py wifi zigbee`
- Dal controller inserire l'indirizzo IP del Raspberry Pi. Come controller é possibile utilizzare l'applicazione Android (disponibile in `RoutexControllerApp.tar.gz`) o il sito web (`RoutexControllerSite`).
- Alimentando i dispositivi, essi effettuano il join alla rete. Se le operazioni sono state eseguite correttamente il controller mostra i dettagli dei device e i servizi che implementano.

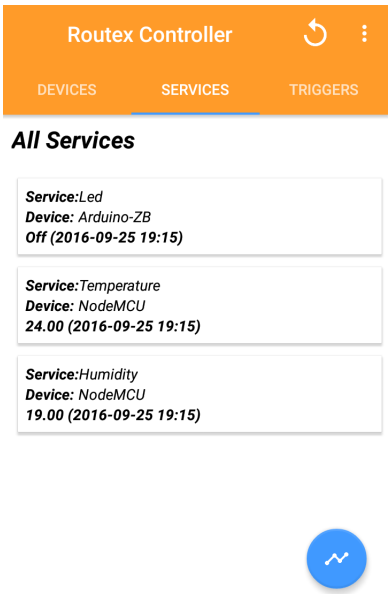


Figura A.3: Il controller mostra la lista dei servizi

- Per interagire in modo diretto con i device é sufficiente fare click su uno dei servizi elencati dal controller e poi richiedere l'esplicita esecuzione di un comando. Il dispositivo coinvolto agisce di conseguenza e restituisce il valore prodotto.

- Per impostare la lettura periodica della temperatura bisogna aprire il dialog di creazione di un evento programmato dalla pagina del servizio, selezionare il comando `Get` (che é l'unico che il servizio relativo alla temperatura mette a disposizione) e specificare l'intervallo di tempo desiderato.

- Per la creazione di un trigger si richiede di aprire il dialog di creazione del tab dei trigger. Il primo parametro da selezionare é la reazione, quindi in questo caso l'accensione del led dell'Arduino. In seguito verrá richiesta la condizione, ad esempio il superamento di un certo valore di temperatura. Quando la condizione risulta verificata si puó notare l'accensione del led.

- Infine per la pubblicazione dei valori raccolti su un canale ThingSpeak é sufficiente premere il bottone di attivazione nella pagina del servizio. In questo modo si effettua la creazione del canale su cui tutti i valori raccolti saranno pubblicati.

Bibliografia

[1] Claire Rowland, Elizabeth Goodman, Martin Charlier, Ann Light, Alfred Lui, *Designing Connected Products: UX for the Consumer Internet of Things*, O'Reilly Media Inc, 2015

[2] Adrian Mcewen, Hakim Cassimally, *Designing the Internet of Things*, John Wiley & Sons, 2014

[3] Luigi Atzori, Antonio Iera, Giacomo Morabito, *The Internet of Things: A survey*, Computer Networks, 2010

[4] Roy Want, Bill N. Schilit, Scott Jenson, *Enabling the Internet of Things*, Computer, 2015

[5] http://www.openhab.org/

[6] C.P. Kruger, A.M. Abu-Mahfouz, G.P. Hancke, *Rapid Prototyping of a Wireless Sensor Network Gateway for the Internet of Things Using off-the-shelf Components*, Industrial Technology (ICIT), IEEE International Conference on, 2015

[7] Kevin I-Kai Wang, Diwakar Somu, Tejas Parnerkar, Zoran Salcic, *Intelligent reconfigurable gateway for heterogeneous wireless sensor and actuator networks*, UIC-ATC-ScalCom-CBDCom-IoP, 2015

[8] Trio Adiono, R. V. W. Putra, Maulana Yusuf Fathany, Waskita Adijarto, *Smart Home Platform Based on Optimized Wireless Sensor Network*

Protocol and Scalable Architecture, Telecommunication Systems Services and Applications (TSSA), 9th International Conference on, 2015

[9] F. Montori, L. Bedogni, L. Bononi, *On the Integration of Heterogeneous Data Sources for the Collaborative Internet of Things*, Proceedings of the 2nd IEEE Internal Forum on Research and Technologies for Society and Industry, Technologies for smarter societies (IEEE RTSI), 2016

[10] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, Weijun Qin, *IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things*, Embedded and Ubiquitous Computing (EUC), IEEE/IFIP 8th International Conference on, 2010

[11] V.Abinayaa, Anagha Jayan, *Case Study on Comparison of Wireless Technologies in Industrial Applications*, International Journal of Scientific and Research Publications, 2014

[12] http://www.digikey.com/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies

[13] https://it.mathworks.com/help/thingspeak/