



**UNIVERSITÀ DEGLI STUDI DI PARMA**  
**DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE**  
**CORSO DI LAUREA IN INGEGNERIA INFORMATICA**  
**CLASSE DELLE LAUREE IN INGEGNERIA DELL'INFORMAZIONE**

**PROGETTAZIONE E REALIZZAZIONE DI UN SISTEMA DI  
AMBIENT INTELLIGENCE BASATO SU ARDUINO E ANDROID**

**DESIGN AND IMPLEMENTATION OF AN AMBIENT  
INTELLIGENCE SYSTEM BASED ON ARDUINO AND ANDROID**

**Relatore:**

**Chiar.mo Prof. GIANNI CONTE**

**Correlatori:**

**Dott. Ing. MARCO PICONE**

**Dott. Ing. MICHELE AMORETTI**

**Tesi di Laurea Triennale di:  
MIRKO MANCIN**

**ANNO ACCADEMICO 2011-2012**

*A chi mi ha supportato e sopportato!*

# Indice

|  |           |
|--|-----------|
| <b>Introduzione</b>  | <b>1</b>  |
| <b>1 Base scientifica e stato dell'arte</b>                    | <b>5</b>  |
| 1.1 Introduzione . . . . .                                     | 5         |
| 1.2 Open hardware . . . . .                                    | 5         |
| 1.2.1 Arduino . . . . .  | 7         |
| 1.2.2 OpenPicus . . . . .                                      | 13        |
| 1.2.3 Libelium Wasp mote . . . . .                             | 15        |
| 1.2.4 Raspberry PI . . . . .                                   | 17        |
| 1.2.5 Sviluppi e progetti . . . . .                            | 18        |
| 1.3 Ambient intelligence . . . . .                             | 19        |
| 1.4 Wireless sensor network WSN . . . . .                      | 24        |
| 1.5 M2M . . . . .  | 28        |
| 1.6 Android . . . . .  | 33        |
| <b>2 Architettura e implementazione</b>                        | <b>38</b> |
| 2.1 Introduzione . . . . .                                     | 38        |
| 2.2 Architettura del sistema di Ambient Intelligence . . . . . | 39        |
| 2.2.1 Monitoraggio ambientale . . . . .                        | 40        |
| 2.2.2 Scheda di riconoscimento . . . . .                       | 46        |

|                            |   |            |
|----------------------------|---|------------|
| 2.2.3                      | Mega ADK e Attuatori . . . . .                          | 54         |
| 2.3                        | Networking . . . . .                                    | 59         |
| 2.3.1                      | Ethernet . . . . .                                      | 60         |
| 2.3.2                      | Wireless . . . . .                                      | 64         |
| 2.3.3                      | WiFi - IEEE 802.11 . . . . .                            | 65         |
| 2.3.4                      | ZigBee - IEEE 802.15.4 . . . . .                        | 71         |
| 2.4                        | Server . . . . .  | 75         |
| 2.4.1                      | Web Client . . . . .                                    | 80         |
| 2.5                        | Applicazione mobile . . . . .                           | 83         |
| 2.6                        | Data Mining . . . . .                                   | 86         |
| 2.6.1                      | Reti bayesiane . . . . .                                | 92         |
| 2.7                        | Attuatori . . . . .                                     | 100        |
| <b>3</b>                   | <b>Testing e risultati sperimental</b> i                | <b>108</b> |
| 3.1                        | Introduzione . . . . .                                  | 108        |
| 3.2                        | Testing . . . . .                                       | 108        |
| 3.2.1                      | Test sulle schede Arduino con il modulo XBEE . . . . .  | 109        |
| 3.2.2                      | Test sulle schede Arduino con il modulo WiFly . . . . . | 111        |
| 3.2.3                      | Problemi di memoria . . . . .                           | 113        |
| 3.2.4                      | Test dell'applicazione Android . . . . .                | 114        |
| 3.2.5                      | Test delle reti di Bayes . . . . .                      | 114        |
| <b>Conclusioni</b>         |   | <b>119</b> |
| <b>Indice delle Figure</b> |   | <b>123</b> |
| <b>Bibliografia</b>        |   | <b>124</b> |

# Introduzione

Negli ultimi anni molta ricerca è stata fatta al fine di trovare nuove soluzioni per migliorare il comfort negli ambienti di lavoro e di vita quotidiana. Si è così scoperta la presenza di una forte corrispondenza tra gli ambienti monitorati ed il rendimento cognitivo e il livello di attenzione delle persone occupanti i luoghi in esame.

Il concetto di intelligenza ambientale (sigla AmI) riguarda gli scenari nei quali le persone vivono circondati da tecnologia informatica e telematica, cioè da dispositivi con capacità computazionali e di connessione in rete, che si mettono a loro disposizione in modo non invadente. Questo concetto è stato sviluppato da ISTAG, Information Society Technologies Advisory Group, gruppo di consulenza della direzione generale della Società dell'Informazione e Mezzi di Comunicazione della Commissione Europea. L'intelligenza ambientale pone in rilievo le caratteristiche di user-friendliness, il supporto a servizi efficienti e distribuiti, il potenziamento degli utenti e il sostegno alle interazioni umane. Per realizzare questa visione è necessario l'incremento dall'attuale disponibilità di personal computer e di dispositivi di vario genere, inseriti in modo discreto nei nostri ambienti e accessibili attraverso interfacce intelligenti. L'AmI si basa su diverse soluzioni tecnologiche:

- Hardware non invadente (miniaturizzazione, nanotecnologia, dispositivi intelligenti, sensori ecc.);
- Infrastruttura di comunicazione integrata con il Web con componenti fisse e mobili priva di smagliature (dotata di interoperabilità, reti cablate e wireless ecc.);

- Dispositivi di rete largamente distribuiti;
- Interfacce con le quali si possa interagire in modo naturale (agenti intelligenti, interfacce multimodali, modelli di consapevolezza del contesto nel quale ci si trova, ecc.);
- Affidabilità e sicurezza (software autoverificante e autoriconfigurante, tecnologia che assicuri la riservatezza ecc.).

In sintesi sono necessari sistemi e tecnologie altamente reattivi, interconnessi, context-aware, trasparenti e intelligenti.

I modelli di sviluppo software open source hanno creato alcuni degli strumenti più innovativi e le imprese del settore hanno modificato il loro modo di creare e sviluppare il business aziendale. Uno degli scopi di questa tesi è quello di analizzare l'hardware open source nel suo stato attuale e il suo impatto potenziale a diversi livelli del suo sviluppo. Con il termine hardware open source (OSHW) ci si riferisce ad hardware elettronici e computer che sono stati progettati con la stessa politica del software libero ed open source (FOSS, Free and open source software). L'hardware libero è parte della cultura dell'open source, che espande quest'ideologia al di fuori dell'ambito del software. Il termine viene principalmente usato per esprimere la libera divulgazione di informazioni riguardanti il progetto stesso. L'open source hardware comprende gli schemi, la lista dei materiali, il layout dei dati del circuito stampato unito al FOSS per far funzionare l'hardware. OSHW e FOSS possono essere usati, studiati e modificati senza restrizioni, possono essere copiati e ridistribuiti in qualsiasi forma: l'unica restrizione consiste nel rispetto dei criteri sopra citati per le copie modificate successivamente. Con l'aumento di dispositivi logici programmabili riconfigurabili, la condivisione degli schemi logici, cioè il linguaggio di descrizione hardware (VHDL), è stata una prima forma di hardware open source. Descrizioni VHDL sono comunemente usate per impostare il sistema-on-a-chip sia nel campo delle FPGA (Field Programmable Gate Array - Microcontrollori per uso industriale) o direttamente in applicazioni specifiche di

circuiti integrati. Moduli HDL, se distribuiti, sono chiamati core di proprietà intellettuale a semiconduttore, o core IP.

Oggi invece abbiamo a disposizione diverse tecnologie open hardware che permettono, con un spesa irrisoria e una conoscenza minima di programmazione, di ottenere dispositivi in grado di emulare sistemi ad alte prestazioni. Arduino ne è un esempio. Questa piattaforma in pochi anni ha sviluppato una considerevole comunità online in cui si possono consultare innumerevoli blog amatoriali, siti Internet e forum in cui le persone, secondo la filosofia FOSS, possono condividere i loro progetti in modo tale da poterli migliorare a loro volta. Con l'open hardware è inoltre possibile progettare sistemi di ambient intelligence a basso costo.

L'M2M è un concetto indipendente che può essere sviluppato anche grazie all'Open Hardware. Con il termine M2M, acronimo di Machine-to-Machine, ci si riferisce a tecnologie ed applicazioni di telemetria e telematica che utilizzano le reti sia cablate che wireless. Machine-to-Machine indica anche un insieme di software ed applicazioni che migliorano l'efficienza e la qualità dei processi domestici e soprattutto industriali: dalla spostamento e stoccaggio merci utilizzando tag RFID, alla gestione delle relazioni con il cliente finale tramite sms. Il funzionamento di una comunicazione M2M è semplice: un dispositivo (ad esempio un sensore) rileva un dato (come temperatura, quantità di gas nell'aria, livello di liquidi, ecc.) che è inviato, attraverso la rete, ad un'applicazione (spesso un server) che traduce il dato ricevuto in un'informazione utile al sistema. Al giorno d'oggi è possibile utilizzare queste tecnologie sui vari dispositivi offerti dal mercato, ad esempio i dispositivi mobile.

L'obiettivo di questa Tesi è la progettazione e lo sviluppo di un sistema di monitoraggio di ambienti indoor attraverso i principi dell'ambient intelligence e del Machine-to-Machine. La particolarità di questo progetto è la determinazione delle diverse situazioni domestiche in base ad alcune grandezze fisiche tipiche del monitoraggio ambientale dell'aria e del rilevamento ambientale. Infatti, si è cercato di raggiungere lo scopo raccogliendo semplicemente

dati di temperatura, umidità, luce e CO<sub>2</sub> e monitorando sensori di supervisione. Un sistema che riesca a determinare la presenza di persone o meno, riuscirebbe in questo caso a dare delle informazioni importanti ai sistemi di comfort installati ( impianti di areazione , tapparelle automatizzate, ecc). Queste rilevazioni sono state effettuate grazie ad Arduino e Android, appunto per mantenere i principi di Open Hardware e Open Source citati precedentemente. Con queste due piattaforme è possibile creare dispositivi smart e autonomi in grado di interagire con l'uomo ma anche indipendenti. Con Arduino è stata creata la parte hardware di raccolta dati, di invio dei dati ad un server e infine la parte d'interazione con degli attuatori. Mentre con Android è stata creata un'applicazione ad-hoc con la quale è possibile monitorare i dati ricevuti sul server e controllare in remoto gli attuatori.

## Struttura della tesi

Questa Tesi si compone di tre sezioni: nel Capitolo 1 vengono analizzate le basi scientifiche e lo stato dell'arte sulle piattaforme aperte, dove si descrive come sono nate, le tecnologie che utilizzano questa architettura e progetti principali già esistenti. Nel Capitolo 2 si illustrano l'architettura e l'implementazione di un sistema di ambient intelligence. Nel Capitolo 3 si descrivono i risultati ottenuti e si analizzano i dati raccolti. Alla fine della tesi 3.2.5 vengono presentati alcuni possibili sviluppi futuri.

# **Capitolo 1**

## **Base scientifica e stato dell'arte**

### **1.1 Introduzione**

In questo capitolo vengono presentate le tecnologie open hardware, i principi dell'ambient intelligence, del M2M e delle WSN; infine viene introdotto l'ambiente Android per il monitoraggio mobile. Nella Sezione 1.2 si parla delle tecnologie presenti attualmente sul mercato; nella Sezione 1.3 si introduce il concetto di ambient intelligence per ambienti domestici e non. Nella Sezione 1.5 si presentano le tecnologie M2M con riferimenti ad alcuni progetti presenti sul mercato, mentre nella Sezione 1.4 si introducono le reti WSN. Per finire, nella Sezione 1.6 è presente una breve introduzione al sistema operativo mobile Android.

### **1.2 Open hardware**

Con il termine Hardware Commodity [1] si intende un hardware semplice, conveniente e facilmente reperibile. Si dice che un dispositivo, o un apparato, utilizza hardware commodity se fa uso di componenti già esistenti o già progettati, cioè non sviluppati appositamente per quella periferica. Ad esempio, la console di casa Microsoft, XBOX, utilizza l'hardware commodity perché fa uso di componenti comuni a PC desktop ordinari. L'hardware com-

modity sta trainando l'innovazione nel settore dell'Information Technology. Tutto quanto di nuovo si sta vedendo è creato con il software su hardware standard. Le startup nascono e si sviluppano grazie al fatto che è molto più economico sviluppare idee nuove su hardware standard e componenti software open. Ora però lo sviluppo tecnologico sta vivendo una nuova fase, dove i dispositivi della nuova generazione diventano sempre più intelligenti (smart), connessi e autonomi; l'hardware standard, inoltre, non è più all'altezza, anche solo per dimensioni fisiche. Molte delle novità che stanno arrivando sul mercato cambiano radicalmente le necessità degli sviluppatori e il loro modo di operare. Tutti i dispositivi della prossima generazione saranno sempre più connessi e dovranno essere capaci di parlare fra loro per scambiarsi informazioni, ad esempio attraverso l'utilizzo di API dedicate. Per questo motivo è importante sviluppare progetti basati su criteri Open e hardware standard [2].

L'Open Hardware è una filosofia relativamente recente che sta prendendo rapidamente piede. Tra i vari esempi di Open Hardware spiccano sicuramente Arduino, Libelium e Raspberry PI, tre piattaforme diverse per struttura e obiettivi, ma ugualmente interessanti. L'obiettivo di questi progetti, in quanto rivolti a un pubblico relativamente eterogeneo, è quello di mettere a disposizione piattaforme a bassissimo costo ma molto elastiche e relativamente potenti per sviluppare progetti in cui serve una certa capacità di elaborazione e connettività.

Gli esempi di Open Hardware sopra citati sono semplici, facili da espandere, implementare e modificare con strumenti di sviluppo software alla portata della maggior parte degli sviluppatori. Essi sono una categoria di computer particolarmente efficaci per creare prototipi adatti ad una realizzazione su piccola scala; sono adatti anche alla miniaturizzazione in vista di una produzione industriale. In figura 1.1 un esempio di una stampante 3D open source. Si abbatte anche in questo caso ogni barriera all'innovazione e le nuove idee sono sviluppabili con investimenti economici contenuti.

Questi dispositivi usano principalmente hardware ARM-based (le CPU presenti su te-



Figura 1.1: RepRap - Stampante 3D open source a basso costo

lefonini e tablet): questa tecnologia è già pronta per applicazioni in campo mobile ed è caratterizzata da un basso consumo di energia, caratteristica indispensabile per dispositivi portatili. L'adozione di questi nuovi Open HW non è un vantaggio solo per i laboratori di ricerca delle Università o per i progetti nelle aziende ma anche, e forse soprattutto, per tutte quelle persone che hanno un'idea e hanno l'opportunità di svilupparla senza ricorrere ad esperti del settore o a soluzioni professionali [3].

### 1.2.1 Arduino

Arduino (Figura 1.2) è un sistema Hardware Open Source a basso costo e molto versatile, programmabile su tutti i sistemi operativi (Mac, Linux e Windows), adatto per controllare luci, motori e attuatori di ogni genere, integrare dei sensori e realizzare in maniera agevole sperimentazioni in campo robotico.

Il team di Arduino [4] è composto da Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis. Il progetto prese avvio in Italia a Ivrea nel 2005, con lo scopo di rendere disponibile per progetti di Interaction design realizzati da studenti, un dispositivo per il controllo che fosse più economico rispetto ai sistemi di prototipazione allora disponibili. I progettisti riuscirono a creare una piattaforma di semplice utilizzo ma che, al tempo stesso, permetteva un'efficace riduzione dei costi rispetto ad altri prodotti

disponibili sul mercato. A ottobre 2008 in tutto il mondo erano già stati venduti più di cinquantamila esemplari di Arduino.

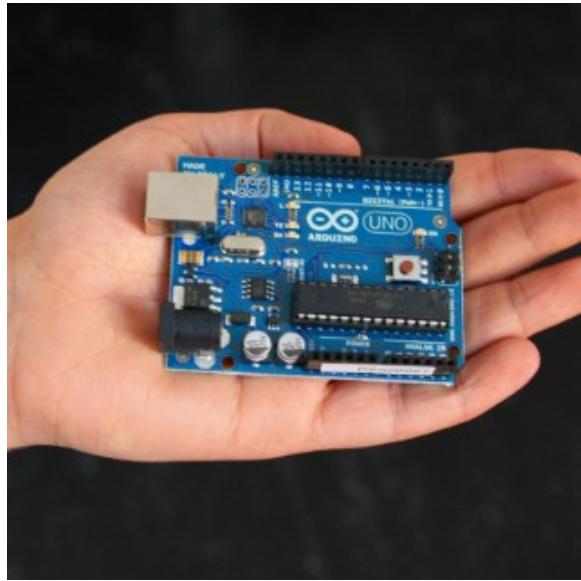


Figura 1.2: Arduino Uno

Arduino è una piattaforma basata su una scheda a microcontrollore ed un semplice ambiente di sviluppo che implementa il linguaggio Processing/Wiring. Le sue dimensioni sono paragonabili ad una carta di credito ed il suo costo è molto contenuto.

La principale caratteristica di Arduino sta nel fatto che il suo sistema è “aperto” sia a livello software che hardware: gli schemi elettrici e tutte le altre informazioni per costruire un clone di Arduino sono reperibili online gratuitamente (esempio Freakduino, Freeduino, Luigino, ecc) [5].

Quindi Arduino è un sistema con software aperto, hardware aperto e documentazione facilmente reperibile in rete e utilizzabile spiegando l’elettronica in un linguaggio più semplice possibile. Le applicazioni e i progetti già sviluppati con questa piattaforma sono diversi: strumenti musicali, installazioni interattive per i musei, prodotti tecnologici come i lettori mp3 e microPC portatili, riproduzione di strumenti da laboratorio dai costi molto alti (e per questo non accessibili a tutti). In molte Università e scuole Arduino è utilizz-

zato a scopo didattico. Oltre alla scheda elettronica, Arduino include anche un ambiente di sviluppo (IDE in inglese)(Figura 1.3), cioè un programma da installare sul computer per la scrittura e il caricamento del software sulla scheda Arduino. L'IDE è un'applicazione multipiattaforma scritta in Java, ed è derivata dal software creato per il linguaggio di programmazione Processing e per il progetto Wiring. Essa è concepita e progettata per permettere la programmazione ad utenti come artisti ed altri neofiti che non sono pratici di sviluppo software. Per permettere la stesura del codice sorgente, l'IDE include un editor di testo dotato inoltre di alcune particolarità, come il syntax highlighting, il controllo delle parentesi, e l'indentazione automatica. L'editor è inoltre in grado di compilare e lanciare il programma eseguibile con un unico click. In genere non vi è bisogno di creare dei Makefile o far girare programmi dalla riga di comando.

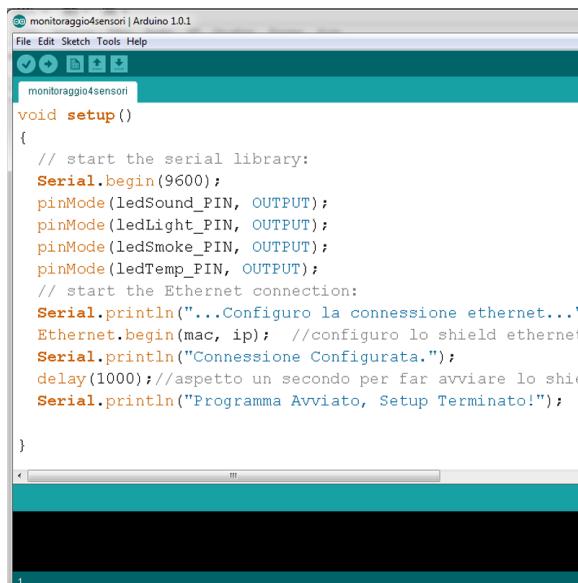


Figura 1.3: Sketch d'esempio con l'Arduino IDE

Grazie alla base software comune, ideata dai creatori del progetto, per la comunità Arduino è stato possibile sviluppare programmi per connettere a questo hardware praticamente oggetto elettronico, computer, sensore, display o attuatore. Dopo anni di sperimentazione è oggi possibile fruire di un vastissimo database online d'informazioni.

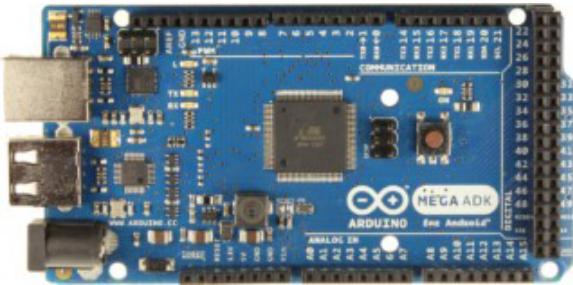


Figura 1.4: Arduino MEGA ADK dall'alto

Una scheda Arduino tipica consiste di un microcontrollore a 8-bit AVR prodotto dall'Atmel, con l'aggiunta di componenti complementari per facilitarne l'incorporazione in altri circuiti. In queste schede sono usati chip della serie megaAVR - nello specifico i modelli ATmega8, ATmega168, ATmega328, ATmega1280 e ATmega2560.

Molte schede includono un regolatore lineare di tensione a 5 V e un oscillatore a cristallo a 16 MHz, sebbene alcune implementazioni, come ad esempio la piccola Lillypad, abbiano un clock di 8 MHz e facciano a meno dello stabilizzatore di tensione.

Fino a oggi sono state commercializzate diverse versioni dell'hardware Arduino [6]:

- Serial Arduino, programmata con una porta seriale DB9. Fa uso del microcontrollore ATmega8;
- LillyPad Arduino, un progetto minimalista (scheda circolare dal diametro di 50mm, per circa 8mm di spessore), per applicazione su indumenti, con lo stesso ATmega168 in versione SMD;
- Arduino Duemilanove, facente uso del chip Atmega168 (o Atmega328 nelle versioni più recenti) e alimentata in corrente continua tramite USB, con commutazione automatica tra le sorgenti di alimentazione;
- Arduino Mega, che fa uso di un ATmega1280 a montaggio superficiale per I/O e memoria addizionale;

- Arduino Uno, evoluzione della Duemilanove, con un differente chip, programmabile e più economico, dedicato alla conversione USB-seriale;
- Arduino Mega2560, che fa uso di un ATmega2560 ed è un'evoluzione dell'Arduino Mega;
- Arduino Leonardo, che fa uso di un ATmega32u4 di nuova generazione e più pin per la programmazione I/O rispetto Arduino UNO;
- Arduino Due, di prossima uscita.

Il microcontroller della scheda, in alcuni casi, è pre-programmato con un bootloader che semplifica il caricamento dei programmi sulla memoria flash incorporata nel chip [7].

A livello concettuale, tutte le schede sono programmate attraverso una porta seriale RS-232, ma il modo in cui questa funzionalità è implementata nell'hardware varia da versione a versione. Le schede seriali Arduino contengono un semplice circuito inverter che permette la conversione tra il livello della RS-232 e il livello dei segnali TTL.

Le versioni attuali di Arduino sono gestite via USB: la versione Uno utilizza un microcontrollore Atmega8U2 programmato come convertitore USB-seriale mentre le precedenti versioni (come la Diecimila e Duemilanove) usavano chip adattatori USB-seriale, come gli FT232 di FTDI. Alcune varianti, come la Arduino Mini e la versione non ufficiale Boarduino, usano una scheda o un cavo adattatore USB-seriale staccabile [8][9].

Per implementare il comportamento interattivo, Arduino è fornita di funzionalità di input/output (I/O). Grazie ai canali d'input riceve i segnali raccolti da sensori esterni. Secondo tali valori, il comportamento della scheda è gestito dal microcontroller, in base alle decisioni determinate dal particolare programma in esecuzione in quel momento sulla scheda. L'interazione con l'esterno avviene attraverso attuatori pilotati dal programma attraverso i canali di output in dotazione.

A tale scopo, Arduino è dotata di molti dei connettori di input/output per microcontroller in uso su altri circuiti. Tutti i pin di I/O sono collocati sulla parte superiore della

scheda, mediante connettori femmina da 0,1; sono disponibili commercialmente molte schede applicative plug-in, note come shields, che possono essere montate direttamente sopra ad Arduino [10].

La scheda Arduino Uno, ad esempio, offre 14 connettori per l'I/O digitale (numerati da 0 a 13). La direzione di funzionamento, input o output, è decisa dallo sketch programmato sull'IDE.

Sei dei canali I/O possono produrre segnali Pulse-width modulation (PWM). Attraverso di essi è possibile, ad esempio, regolare l'intensità di luminosità di un LED o la velocità di rotazione di un motorino elettrico.

Sono presenti altri 6 connettori appositamente dedicati a ingressi di segnali analogici (collegati quindi ad una ADC), cioè livelli di tensione letti da sensori esterni i cui valori, al massimo di 5 Volt, sono convertiti in 1024 livelli discreti (da 0 a 1023). Questi connettori possono essere riprogrammati (sempre dal codice dello sketch sull'IDE) per funzionare come normali entrate/uscite digitali.

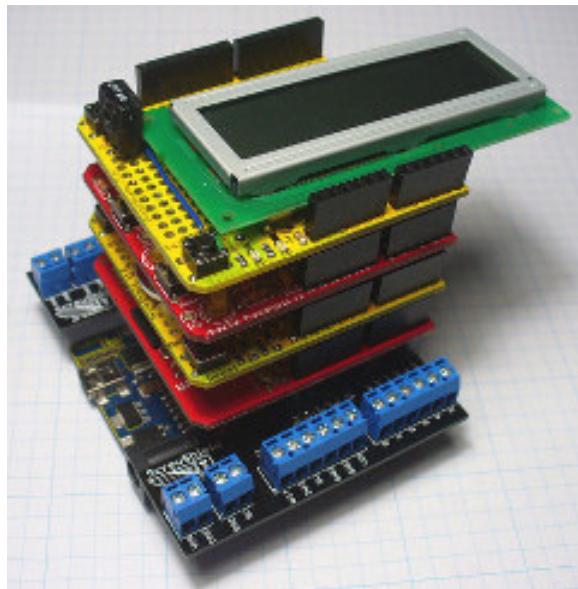


Figura 1.5: Esempio di Arduino Extreme Shielding

Un altro aspetto interessante è la modularità di questi dispositivi: grazie all'hardware

messo a disposizione da Arduino è possibile ampliare la scheda di base con ulteriori moduli (shield) semplicemente impilandoli uno sopra l'altro (grazie allo stesso PCB e allo stesso pin out). Ciò permettere ad Arduino, per esempio, di ottenere connettività TCP/IP attraverso un'interfaccia Ethernet o WiFi, Bluetooth, possibilità di collegare motori in corrente alternata, ecc. Un esempio è mostrato in Figura 1.5.

### 1.2.2 OpenPicus

OpenPicus [11] è una azienda italiana che produce piattaforme wireless embedded (Bluetooth+Wi-Fi) con alla base un PIC24F (Figura 1.6). I moduli sono sviluppati secondo le specifiche di Free and Open Source OS (FreeRTOS) e sono forniti di un software stack TCP/IP dedicato utile per far funzionare il modulo stesso come web server. Il modulo è dedicato per la sensoristica wireless e la domotica, e sul PIC gira sia lo stack Wi-Fi che Bluetooth.



Figura 1.6: scheda Flyport prodotta da OpenPicus

L'azienda produce principalmente tre tipi di prodotti:

- Flyport, un sistema basato sul processore PIC della Microchip con due differenti tipi di connettività: WiFi ed Ethernet. I moduli Flyport sono usati per connettere e controllare diversi sistemi in Internet attraverso un embedded-server personalizzato o

servizi standard TCP/IP. Il microcontrollore esegue l'applicazione e non sono necessari ulteriori processori. Il pinout è modificabile da software.

- Schede di espansione (Nest) che sono create per specifiche applicazioni e sono compatibili con ogni Flyport.
- Un software IDE gratuito di sviluppo che fornisce gli strumenti per creare, compilare e scaricare le applicazioni (firmware) sui moduli.

Su una scheda Flyport possono essere eseguiti diversi servizi contemporaneamente:

- Web server personalizzabile
- TCP/UDP client/server
- FTP Client
- Email Client

Flyport ha anche la possibilità di aggiornare il firmware su Internet: il file .hex può essere salvato in un server FTP il quale si occuperà di scaricare nella memoria flash del modulo il programma aggiornato (funziona solo con il modulo Flyport Ethernet). Le librerie esterne possono essere utilizzate per l'hardware esterno (come sensori/attuatori) o per mandare/ricevere dati in Cloud [12].

### **Hardware**

- CPU Microchip PIC 24F 16 bit 44 pins QFP,
- 64K Flash 8K Ram,
- Wireless (Bluetooth / Wi-Fi),
- Alimentazione 5V o 3,3V,
- Connessioni: UART, Input digitali, Output digitali, Input Analogici, PWMs, SPI display, I2C,

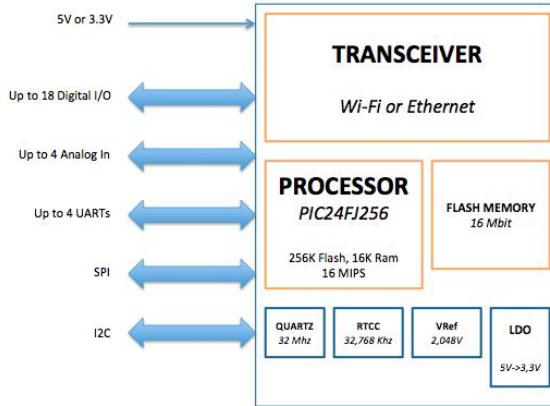


Figura 1.7: Architettura di una scheda FlyPort

- Connettore: 26 Vie IDC passo 2,54mm (maschio)

## Software

- Flyport dispone di un bootloader integrato per essere programmato tramite seriale.
- Profili Bluetooth : SPP, OBEX, Headset profiles.
- Funzioni Wi-Fi : webserver integrato, TCP socket, embedded FTP ed email client.

### 1.2.3 Libelium WaspMote

Libelium[13] è un'azienda spagnola che produce schede open source a basso costo. Il suo nuovo prodotto commercializzato nel 2012 è la scheda WaspMote ed è un dispositivo che gira su un microcontrollore Atmel ATmega1281, con clock a 8MHz, 8KB SRAM, 4KB EEPROM e 128KB flash. La mainboard è disponibile anche con una Flash card da 2GB opzionale. Essa vanta consumi ridotti in standby (circa 0.7uA), fornisce frequenze radio a 2.4GHz, 900MHz, e 868MHz con i moduli ZigBee, offre add-ons per GPS, GPRS e SD e monta una sensor board integrata (novità rispetto alle precedenti schede SquidBee). Differentemente dal Libelium's x86-based o dal Meshlibum multi-protocol mesh router o la sua versione outdoor N-Vio WiFi

routers, sul Wasp mote non gira Linux, anche se, API per programmatore C sono disponibili open source per le piattaforme Linux, Mac e Windows.



Figura 1.8: Wasp mote Sensorboards

Il Wasp mote [14] è equipaggiato con un I/O digitale (7 analogico, 8 digitale), più una porta mini-USB, interfacce per PWM, I2C e due UART. I controlli onboard includono un accelerometro a 3 assi per misurare ranges che vanno da  $\pm 2g$  a  $\pm 6g$ , un sensore di temperatura con un range che va da -40 a 85 gradi Celsius, con una precisione di 0.25 C di scarto. I consumatori potranno scegliere tra sette XBee radios, equipaggiati con antenne esterne. Queste includono due 802.15.4 (ZigBee) standard e due ZigBee-Pro a 2.4GHz di frequenza. Essi sono offerti in una varietà di configurazioni di potenza: i radios vanno da 500 mt a 7 km di range. Gli altri tre radios sono modelli RF standard a 868MHz e 900MHz che offrono ranges che vanno dai 10Km ai 40Km. In aggiunta alle XBee radios, gli utenti possono comprare un modulo opzionale GSM/GPRS e GPS, così come una radio gateway Wasp mote ZigBee che si inserisce in un PC tramite interfaccia USB. Inoltre Libelium dispone di tre sensori integrati:

- Gas - rileva CO, CO<sub>2</sub>, CH<sub>4</sub>, SH<sub>2</sub>, NH<sub>3</sub>, e altro. Questo sensore è orientato nello studio contro l'inquinamento, controllo di emissioni di fabbrica, controllo dei processi chimici e incendi boschivi.
- Eventi - rileva la luminosità, vibrazioni, livello dei liquidi, e altro ancora. E' stato pensato la sicurezza e controllo logistico.
- Prototipi - disegnato per testare nuove funzioni e nuovi sensori, incluso l' ADC, pad area e amplificazione di ambienti.

### 1.2.4 Raspberry PI

Il Raspberry Pi[15] è un single-board computer (un calcolatore implementato su una sola scheda elettronica) sviluppato nel Regno Unito dalla Raspberry Pi Foundation all'inizio del 2012 (Figura 1.9).

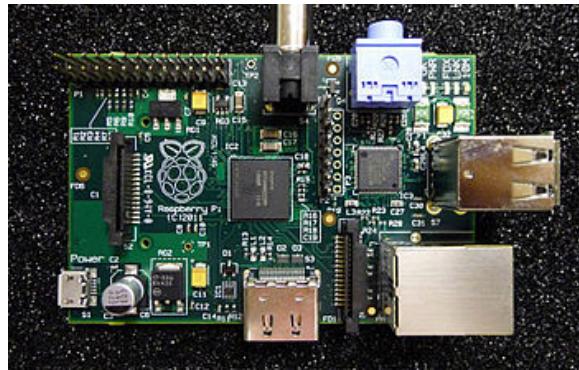


Figura 1.9: Scheda Raspberry PI

L'idea di base è la realizzazione di un dispositivo economico, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole. Il progetto ruota attorno a un System-on-a-chip (SoC) Broadcom BCM2835 che incorpora un processore ARM1176JZF-S a 700 MHz, una GPU VideoCore IV e 256 megabyte (MiB) di memoria. Il progetto non prevede né hard disk né una unità a stato solido, affidandosi invece a una scheda SD per il boot e per la memoria non volatile. La scheda è stata progettata per ospitare sistemi operativi basati su un kernel Linux o RISC OS.

La fondazione mette a disposizione due modelli. Il Model A (inizialmente previsto con una dotazione RAM dimezzata) ha una porta USB ed è privo di controller Ethernet, a un costo di 25 US\$, mentre il Model B è equipaggiato con due porte USB e un controller Ethernet 10/100 e ha un costo di 35 US\$. Sebbene il Modello A non abbia una porta Ethernet RJ45, può comunque accedere a una rete attraverso la porta USB, facendo uso di adattatori Ethernet o Wi-Fi con alimentazione autonoma. In maniera analoga ai moderni computer, Raspberry PI è compatibile con tastiere e mouse generici collegabili tramite porta

USB. Raspberry PI userà il sistema operativo Linux. È previsto in futuro che Raspberry sia distribuito in bundle con Debian GNU/Linux, Iceweasel, Calligra Suite e Python.

Raspberry PI non è fornito di un real-time clock, così un sistema operativo deve usare un Network Time Server, o chiedere l'ora all'utente al bootstrap per avere accesso a data e ora per la marca temporale. Tuttavia è facile aggiungere un real time clock (come il DS1307) con batteria tampone, attraverso l'interfaccia I<sup>2</sup>C.

### 1.2.5 Sviluppi e progetti

L'Open Hardware ha dato il via ad un nuovo modo di progettare e reinventare dispositivi. Sono sorti diversi siti internet, blog e magazine nei quali è possibile condividere le proprie idee e aiutare altre persone che necessitano di un aiuto. Do-It-Yourself (DIY) è un termine usato per descrivere la progettazione, modifica o riparazione di qualcosa senza l'ausilio di esperti o professionisti. La frase *fai da te* è entrata in uso comune nel 1950 come riferimento ai progetti di miglioramento domestico che le persone potrebbero autonomamente fare indipendentemente. Negli ultimi anni, il termine DIY ha assunto un significato più ampio che copre una vasta gamma di abilità [16]. Grazie alla concezione di fai da te sono nate vere e proprie comunità di nicchia nelle quali è possibile discutere, commentare, migliorare progetti con dispositivi differenti (come nel caso di Arduino).

Per questo motivo, non sono più necessari mesi di studio per sviluppare progetti basati su questi dispositivi: bastano, infatti, poche settimane per fare progetti veramente interessanti. Per esempio Sebastian Alegria, un ragazzo di quattordici anni cileno, ha creato un sistema di rilevamento per i terremoti e li pubblica su twitter ed ha anticipato un progetto governativo di un anno [17].

Gli ambiti di progettazione e sviluppi sono i più disparati:

- Giochi: Quadricottero (prima solo per scopi militari) [18] Robot che giocano a calcio (progettati da ragazzi che hanno 11 anni) [19];

- Strumenti musicali: Arpa Laser [20];
- Ambito medico: Dispositivo che analizza e rileva il DNA [21];
- Ambito assistenziale: Guanti che rilevano il linguaggio dei segni e li trasformano in suoni [22] Ragazzi con gravi problemi di mobilità che non possono giocare con console per videogiochi e con un dispositivo apposito riescono a giocare a baseball, golf, e altri giochi comunque;
- Ambito artistico: Txtbomber, è un dispositivo con il quale, inserendo un messaggio di testo come su un cellulare, è possibile poi aerografarlo dove si vuole con un semplice spruzzo di bomboletta [23];
- Ambito naturalistico: Dispositivo che rileva lo stato del benessere di una pianta e che interagisce con un account twitter [24];
- Ambito industriale: Google si è interessata a questi dispositivi creando un Kit di sviluppo accessori (ADK) con il quale è possibile interagire con uno smartphone o tablet Android direttamente con una scheda Arduino [25].

Anche il design è fondamentale per il merchandising e l'Italia eccelle in questo ambito. Su questa filosofia è sorta l'azienda italiana Habits [26] che crea design open source. Cioè mette a disposizione progetti dall'inizio alla fine per permettere chiunque di poter replicare in proprio i progetti che vengono pubblicati.

### 1.3 Ambient intelligence

La domotica è una sintesi di diverse discipline tecnico-scientifiche applicate nel contesto dell'automazione degli edifici. Essa mira a renderli più confortevoli, economici nella gestione, e fornisce, inoltre, nuovi servizi nell'ambito della sicurezza, dell'abitabilità, della comunicazione e della diagnosi dei guasti. In altre parole, la domotica si pone l'obiettivo di

migliorare le funzionalità tradizionali di un edificio e di fornirne di nuove, basate su concetti di automazione, elettronica e informatica.

L'Ambient Intelligence, abbreviato in AmI, rappresenta la naturale evoluzione della domotica; questa nuova disciplina prevede la presenza di piccoli sensori interconnessi tra loro, sparsi all'interno di un ambiente e utilizzati per interpretare e riconoscere, tramite tecniche di Intelligenza Artificiale, situazioni significative degli ambienti e delle persone che li frequentano. Alcuni esempi possono essere bisogni, situazioni di pericolo, richieste che possono essere anticipate: lo scopo è quello di risolvere automaticamente tali problemi mediante opportune strategie, possibilmente ancor prima che una richiesta specifica venga rivolta al sistema da un essere umano; si dice che tali sistemi mirano ad esibire un comportamento proattivo. Possiamo definire l'AmI come un nuovo paradigma che mira a migliorare la qualità di vita creando l'atmosfera e le funzionalità desiderate dall'utente mediante sistemi e servizi intelligenti, personalizzati ed interconnessi. Per sistemi distribuiti s'intende un modello d'interazione uomo-macchina nel quale l'elaborazione delle informazioni è interamente integrata nei dispositivi utilizzati; in questo modo essi possono compiere le attività quotidiane senza che l'utente che li utilizza si renda conto delle operazioni e dei calcoli che stanno eseguendo. Ciò è ben diverso dal contesto attuale, nel quale l'utente decide consciamente di azionare un dispositivo con uno scopo specifico. Il termine ambient si riferisce all'ambiente riflettendo le esigenze di distribuzione, pertanto i sistemi di AmI hanno un'architettura distribuita, in modo che il sistema sia presente ovunque sfruttando le nuove frontiere dell'embedding, e trasparente, in modo che il sistema sia invisibile e discreto. Il termine intelligence significa che l'ambiente digitale deve esibire specifiche forme di interazione sociale, riconoscendo le persone che ci vivono in modo da adattarsi a loro, imparare i loro comportamenti e possibilmente manifestare emozioni. Per raffinare la definizione di Ambient Intelligence sono state introdotte cinque caratteristiche chiave che devono avere i sistemi futuri:

- **embedded:** i sistemi AmI devono essere composti da molti dispositivi collegati ed

integrati naturalmente all'interno dell'ambiente;

- **context aware:** il sistema deve riconoscere quello che sta succedendo nell'ambiente;
- **personalizzati:** il sistema deve modificarsi a misura dei bisogni dell'utente;
- **adattativi:** il sistema deve cambiare in risposta alle abitudini dell'utente;
- **anticipatori:** il sistema deve anticipare i bisogni dell'utente senza interventi diretti dello stesso.

Le prime due caratteristiche richiedono all'integrazione dei dispositivi hardware nell'ambiente [27], mentre le altre tre si riferiscono alla necessità di adattamento dei sistemi elettronici in risposta agli utenti.

Un esempio di AmI è una Smart Home [28], una casa equipaggiata in modo da offrire servizi avanzati ai suoi utenti. Naturalmente, quanto intelligente deve essere una casa per essere definita Smart Home è una questione soggettiva. Ad esempio, una stanza può avere un sensore per decidere quando i suoi occupanti sono dentro o fuori oppure per decidere se tenere accese le luci. Se, tuttavia, si ha a disposizione un sensore di movimento, una persona che sta leggendo in una posizione di riposo può confondere il sistema, che potrebbe quindi lasciare la luce spenta, portando il sistema ad una valutazione errata dell'ambiente. Si rende quindi necessaria un'analisi del contesto più accurata.

In figura 1.10 possiamo vedere le caratteristiche base di una casa che può operare intelligentemente [29][30]. Essa deve offrire agli occupanti un normale ambiente familiare, e non deve assolutamente ricordare un laboratorio. Inoltre molti oggetti normalmente presenti in una casa possono essere arricchiti con sensori per raccogliere informazioni sul loro utilizzo ed in alcuni casi anche per agire indipendentemente, senza l'intervento umano.

Benefici che possiamo aspettarci da queste tecnologie possono essere:



Figura 1.10: esempio di smart home

- **maggior sicurezza**, ad esempio monitorando lo stile di vita o le ultime attività e offrendo assistenza qualora si stia per presentare una situazione potenzialmente nociva;
- **comfort**, ad esempio regolando automaticamente la temperatura;
- **risparmio economico**, controllando l'utilizzo delle luci.

Esistono altre applicazioni fattibili e rilevanti per l'ambient intelligence, ad esempio:

- **ospedali e ambulatori:** gli ospedali possono incrementare l'efficienza dei loro servizi monitorando la salute ed i progressi dei pazienti eseguendo analisi automatiche delle attività nelle loro stanze; possono incrementare la sicurezza e ridurre il rischio di

infezioni permettendo solo al personale autorizzato ed ai pazienti di ottenere l'accesso a specifiche aree e dispositivi;

- **settore del trasporto pubblico:** il flusso del traffico può avere dei benefici da tecnologie quali il sat-nav (GPS-based spatial location estimation), per rendere il trasporto più fluido, efficiente e sicuro;
- **servizi per l'educazione:** le università e gli istituti di educazione superiore usano smart card per permettere l'accesso ai laboratori, alle biblioteche, ai parcheggi, alle mense ed alle sale conferenze;
- **servizi per le emergenze:** ambulanze e autopompe dei pompieri possono migliorare il loro tempo di reazione ad un incidente tramite la localizzazione GPS accurata del luogo dove esso è capitato, velocizzando il percorso per raggiungere il dato luogo automatizzando i semafori in loro favore; inoltre, la polizia può rapidamente localizzare il luogo dove sta avvenendo un crimine in modo da gestire in modo ottimale l'accesso alla zona sia per il personale che per i civili;
- **luoghi orientati alla produzione:** le industrie possono auto-organizzarsi in accordo al rapporto tra produzione e domanda dei beni prodotti; questo può richiedere una accurata analisi della correlazione tra la collezione di dati ottenuti tramite sensori all'interno dei differenti settori della linea di produzione ed l'insieme di domande, ottenibile tramite un sistema diagnostico in grado di consigliare le persone su cosa fare, ad un livello di decision-making;
- **sorveglianza pubblica:** l'ampia diffusione di telecamere a circuito chiuso offre la possibilità di monitorare luoghi pubblici potenzialmente pericolosi come centri città, stazioni della metropolitana e mezzi di trasporto pubblico, incrementandone la sorveglianza.

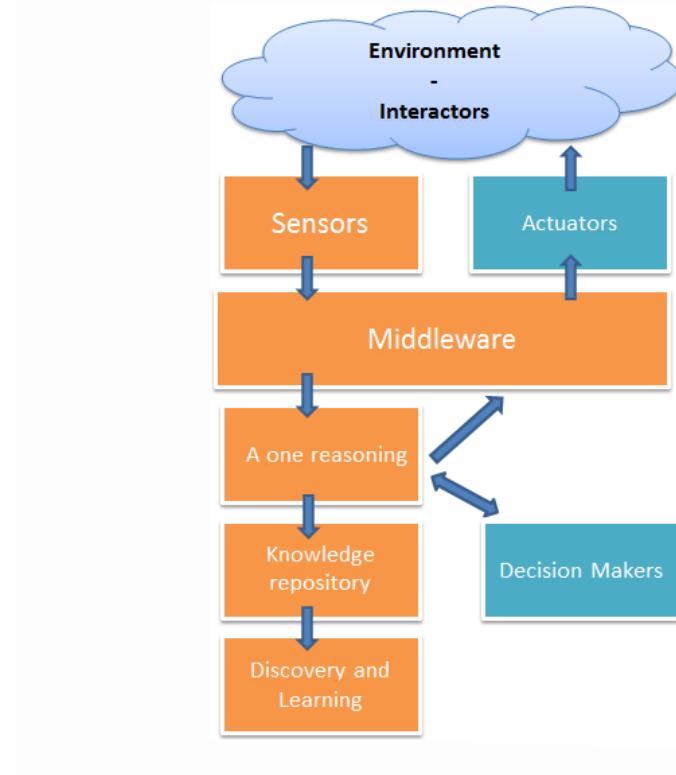


Figura 1.11: Flusso delle informazioni e schema generale di un sistema AmI

## 1.4 Wireless sensor network WSN

Una WSN (Wireless Sensor Network) è definita come un insieme di nodi wireless interconnessi, aventi poca RAM e una CPU a basse prestazioni. La struttura di una Wireless Sensor Network prevede solitamente diversi nodi wireless sparsi in un'area che inviano periodicamente dati rilevati tramite sensori a un punto di raccolta, detto base station o gateway; esso gestisce la rete, raccoglie i dati dei nodi e li inoltra a un altro sistema remoto per successive elaborazioni. I componenti basilari di una rete di questo tipo sono:

- Un insieme di sensori distribuiti
- Una rete d'interconnessione solitamente wireless
- Un punto di raccolta dati

- Un insieme di risorse computazionali nel punto di arrivo dei dati, al fine di eseguire data mining, correlazione dati, monitoraggio dello stato, ecc.

Questo tipo di reti ha il compito di interconnettere apparati di misura o attuatori, al fine di svolgere compiti specifici, come rilevare temperatura, umidità, luminosità, oscillazioni, ecc. Ogni nodo appartenente alla rete ha la capacità di eseguire calcoli, di catturare dati e di comunicare e cooperare con gli altri nodi della rete. Nel panorama delle Personal Area Network sono diffuse numerose tecnologie wireless, prima tra tutte lo standard Bluetooth, anche noto come IEEE 802.15.1. Bluetooth permette l'interconnessione di piccoli dispositivi per trasmissioni di breve durata e prevede due diversi tipi di dispositivi: i nodi master, aventi funzioni di coordinamento, e i nodi finali, detti slave. Lo standard IEEE 802.15.4 è invece specifico per le WSN. Questo tipo particolare di protocollo è stato progettato per offrire, diversamente dalla famiglia IEEE 802.15, la possibilità di poter realizzare reti costituite da un gran numero di nodi e quindi con una densità spaziale anche molto elevata. Inoltre nelle reti di sensori, i limiti in termini di consumo energetico sono fondamentali e la quantità di banda richiesta è piuttosto limitata; non a caso il protocollo 802.15.4, specifico per le Wireless Sensor Network, è un protocollo low-power e low-data-rate. Un esempio di dispositivi che sfrutta tale tecnologia è ZigBee con i suoi moduli XBee. Essi hanno un protocollo wireless compatibile con lo standard ZigBee/IEEE 802.15.4 che soddisfa i requisiti necessari per realizzare un sistema di trasmissione bidirezionale a basso costo ed a basso consumo, orientato soprattutto per l'impiego con sensori. I moduli sono semplici da utilizzare, di limitato ingombro, richiedono pochissima energia e costituiscono una soluzione efficace ed affidabile per la trasmissione di dati a breve raggio.

Una rete di sensori può essere realizzata in tre differenti topologie; con il termine topologia si indica la modalità in cui i diversi dispositivi della rete vengono disposti e le modalità di collegamento fisico e logico.

Nelle reti wireless esistono tre principali strutture: stella (star), mesh e albero (tree), figura 1.12.

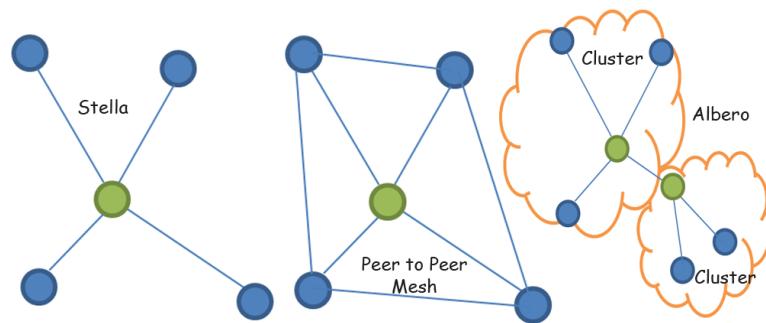


Figura 1.12: Topologie di reti WSN

- La topologia di rete a stella prevede che un unico nodo si comporti da coordinatore e gli altri nodi siano connessi direttamente al coordinatore tramite un link di comunicazione uno-ad-uno.
- La struttura a mesh è formata da nodi-sensori interconnessi tra loro attraverso uno o più link. Questa rete può essere chiamata anche a maglia. In questa struttura è necessario definire chiare regole di routing per lo scambio di messaggi tra i vari nodi. Può esistere anche un coordinatore posto al di fuori della struttura mesh. In questo caso sarà collegato ad uno solo dei nodi ed avrà funzioni di controllo limitate. Se non esiste un coordinatore, tutti i nodi avranno funzioni di coordinamento e raccolta dati in modo paritetico.
- La struttura ad albero è una topologia di rete che combina la struttura a stella e quella a mesh. Più nodi, che fanno capo ad un unico nodo, formano un gruppo chiamato cluster. I nodi-coordinatori possono a loro volta far riferimento ad un nodo coordinatore di livello superiore e costruire così vari livelli di gerarchia.

Le reti di sensori possono essere utilizzate in svariati campi. Viene di seguito presentato un elenco dei settori dove vengono tipicamente impiegate le Wireless Sensor Network.

- **Monitoraggio ambientale** (Enviromental and habitat monitoring) In campo ambientale le applicazioni possono essere molteplici: è possibile monitorare movimenti di animali oppure studiare particolari habitat, come il mare, il terreno o l'aria impiegando, ad esempio, sistemi di monitoraggio di agenti inquinanti. Appartengono a questo settore anche lo studio di eventi naturali catastrofici quali incendi, tornado, terremoti ed eruzioni vulcaniche.
- **Monitoraggio di strutture** ( Structural Health Monitoring) Le reti di sensori posizionate sulle strutture rilevano lo stato di salute di edifici, ponti, case sottoposte a sollecitazioni esterne; possono essere inoltre utilizzate per misurare i difetti strutturali delle componenti.
- **Sorveglianza militare** ( Militar control) Inizialmente, le reti di sensori erano utilizzate proprio per questo scopo. In questo campo, le loro applicazioni spaziano dal monitoraggio dello stato o della posizione delle forze sul campo di battaglia alla sorveglianza di luoghi strategici. Possono inoltre essere utilizzati in luoghi ostili al fine di raccogliere informazioni sugli spostamenti del nemico.
- **Monitoraggio di apparecchiature industriali** ( Industrial sensing) I sensori wireless possono essere applicati a macchinari industriali per analizzarne il comportamento dei componenti sottoposti a stress meccanico, migliorarne le performance e prevenire rotture e guasti.
- **Monitoraggio Agricolo** ( Agriculture sensing) Nel settore agricolo le reti WSN vengono utilizzate per il monitoraggio di particolari situazioni ambientali; in particolare la cosiddetta “agricoltura di precisione” utilizza le reti di sensori per rilevare in tempo reale il livello di pesticidi nell’acqua o nei terreni agricoli.
- **Applicazioni Personal** ( Personal sensoring) Nel settore della domotica, le reti di sensori riescono a fornire servizi all’utente all’interno della propria abitazione, ad

esempio informandolo tempestivamente di eventuali guasti. I sensori possono essere introdotti negli apparecchi elettrici che, interagendo tra loro e con una rete esterna o Internet stesso, permettono all'utente di comandare facilmente gli elettrodomestici a distanza.

## 1.5 M2M

Con l'acronimo M2M [31] ci si riferisce alla convergenza di tre famiglie (oggetti intelligenti, reti di comunicazione e sistemi di computer) appartenenti a tecnologie sia emergenti che già sperimentate che permettono ai diversi componenti di comunicare sia wireless che via cavo. Lo sviluppo di queste tecnologie nasce dall'incremento del numero di oggetti e dispositivi che l'uomo possiede: ogni giorno gli oggetti acquisiscono sempre più valore, essendo collegati in rete o intelligenti (equipaggiati per esempio con sensori real-time). Lo scopo principale del M2M si basa sulla comunicazione degli eventi, attraverso un trasferimento di dati acquisiti da dispositivi a basso livello (sensori), ad un alto livello di dati di calcolo o ad un livello di presentazione. I risultati si possono combinare in una formula unica o semplicemente essere forniti come input per software di data mining; informazioni significative possono essere estratte dalla lettura della temperatura, luminosità e da molte altre fonti. I vantaggi di utilizzo delle tecnologie M2M sono evidenti nelle imprese più grandi e tecnologicamente avanzate: secondo Juniper e IDC, il mercato globale M2M prevede un valore di \$18.9B nel 2014 [32]. Orange segue a ruota e offre competenze e tecnologie per i suoi clienti, presentando loro le opportunità offerte dal M2M, in termini di benefici aziendali reali.

L'idea di M2M si basa su quattro principi fondamentali:

- **Accessibilità in tempo reale:** una visione in tempo reale potrebbe aiutare a prendere decisioni rapide;
- **Conformità:** regolamenti, privacy, ecc;

- **Riduzione dei costi:** il monitoraggio remoto elimina inutili trasferimenti di cose e personale;
- **Differenziazione:** consentire il lancio di nuovi modelli di business o di nuovi servizi, in anticipo rispetto ad altri;

I benefici non sono solo per le imprese: M2M potrebbe migliorare anche le norme della società, portando nuovi mezzi per i servizi assistenziali, servizi di monitoraggio ambientale, auto-monitoraggio e auto-sperimentazione (l'attività per registrare il vostro lavoro, sonno, esercizi vari, la dieta, l'umore, ecc.). Questi servizi saranno supportati da dispositivi elettronici sempre più all'avanguardia, per esempio contapassi Fitbit, sistemi di gestione intelligente del traffico stradale , ecc.

La crescente disponibilità di reti wireless a banda larga in tutto il mondo contribuirà in modo significativo alla diffusione M2M e della Internet of Things. In questo scenario tutto sarà in rete e in grado di comunicare tra loro in modo intelligente, senza l'intervento dell'uomo.

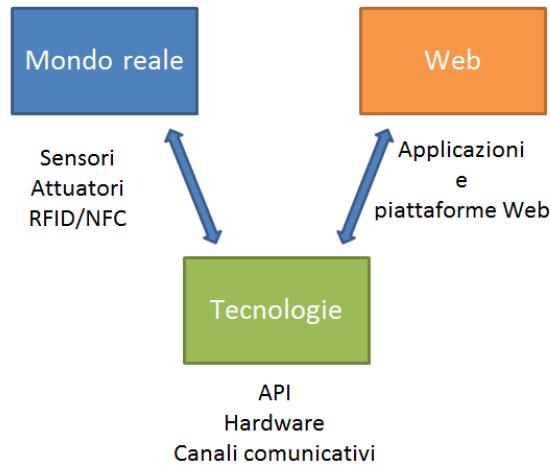


Figura 1.13: Intereazione tra mondo fisico e il mondo web

Internet delle cose (o Internet degli oggetti o IoT, acronimo dell'inglese Internet of Things) è un neologismo riferito all'estensione di Internet al mondo degli oggetti e dei luoghi

concreti. Il suo primo utilizzo ebbe luogo probabilmente nel 1999 presso l’Auto-ID Center, un consorzio di ricerca con sede al MIT. Il concetto fu in seguito sviluppato dall’agenzia di ricerca Gartner.

Internet of things [33] è vista come una possibile evoluzione dell’uso della Rete. Gli oggetti si rendono riconoscibili e acquisiscono intelligenza: infatti possono comunicare dati su se stessi e accedere ad informazioni aggregate da parte di altri. Le sveglie suonano prima in caso di traffico, le piante comunicano all’innaffiatoio quando è il momento di essere innaffiate, le scarpe da ginnastica trasmettono tempi, velocità e distanza per gareggiare in tempo reale con persone dall’altra parte del globo, i contenitori dei medicinali avvisano i familiari se si dimenticano di prendere il farmaco. Tutti gli oggetti possono acquisire un ruolo attivo grazie al collegamento alla Rete. L’obiettivo dell’Internet of Things è di far sì che il mondo elettronico tracci una mappa di quello reale, dando un’identità elettronica alle cose e ai luoghi che lo compongono. Gli oggetti e i luoghi muniti di Etichette Identificazione a Radio Frequenza (RFID) o Codici QR comunicano informazioni in rete o a dispositivi mobili come i telefoni cellulari. I campi di applicabilità sono molteplici: dalle applicazioni industriali (processi produttivi), alla logistica e all’infomobilità, fino all’efficienza energetica, all’assistenza remota e alla tutela ambientale.

Un esempio di azienda italiana che sviluppa M2M è Movia [34]. Essa sviluppa soluzioni per il controllo automatico di dispositivi remoti che trasmettono i loro dati (stato di funzionamento, rilevazioni, allarmi) a un Server di monitoraggio e gestione centralizzata. I dispositivi prodotti da Movia sono degli apparati multi-funzione particolarmente compatti, programmabili e configurabili a distanza. Essi sono in grado di interagire con il “mondo esterno” grazie a una vasta gamma di sensori, attuatori e interfacce di comunicazione. Il server di monitoraggio è una piattaforma robusta e scalabile che offre agli utenti, via web, il pieno controllo dei dispositivi remoti e una serie di funzionalità come: autenticazione, registrazione, ricerca, visualizzazione ed esportazione dei dati raccolti (video, audio, allarmi, storico, diagnostica, posizione, ecc.). Il server e i dispositivi periferici comunicano utilizz-

zando le reti wireless (GPRS, UMTS, WiFi, ecc.) di volta in volta più opportune in base a criteri quali: copertura radio, costi, prestazioni.

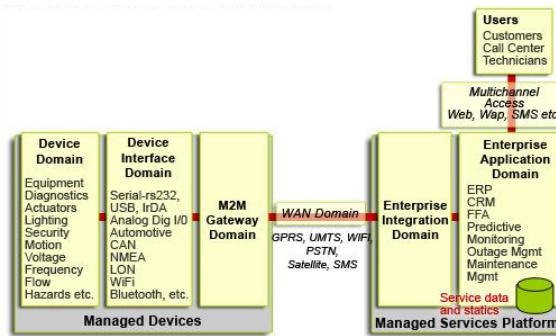


Figura 1.14: Schema di funzionamento di dispositivi Movia

I sistemi M2M offrono benefici in termini di risparmio e di efficienza gestionale in diversi settori di applicazione, quali:

- **Sorveglianza e sicurezza:** dispositivi tascabili per il monitoraggio audio e video, con registrazione su memoria locale e trasmissione a un centro di controllo. Localizzazione tramite GPS e centralizzazione di allarmi dalle periferiche. I benefici per i privati e per i professionisti della sicurezza sono: un controllo puntuale e discreto di beni, luoghi, persone impegnate in attività pericolose; Tutto ciò grazie a dispositivi miniaturizzati, semplici da installare e indossare.
- **Automazione degli impianti e degli edifici:** lettura automatica dei contatori, controllo remoto d'impianti elettrici e termici, diagnostica centralizzata di macchinari per la produzione e la movimentazione (robot industriali, trasportatori, elevatori, ecc), applicazioni domotiche. I benefici per i privati e le aziende di produzione, installazione e manutenzione sono: conoscere in ogni istante lo stato di funzionamento e i consumi riducendo i costi di gestione e manutenzione, e aumentando la qualità del servizio reso dagli impianti.

- **Logistica e Retail:** consentire l'inventario automatico delle merci, il tracciamento di beni e mezzi di trasporto (es. tramite tecnologie barcode, RFID, GPS), monitoraggio di vending machines. I dati raccolti sono integrabili immediatamente nei sistemi informativi del cliente. Forte riduzione dei costi per l'azienda, grazie ad un capillare controllo delle risorse e dell'efficienza lungo tutta la filiera di produzione, trasporto, distribuzione e vendita.

Altre tecnologie M2M sono quelle dell'azienda Guglielmo [35] di Reggio Emilia con il progetto “Guglielmo M2M”: essa è la piattaforma per la gestione di una rete «federata» di sensori che permette lo scambio e l'elaborazione di dati raccolti dagli hot spot Guglielmo. I dati locali sono elaborati dal datacenter carrier grade di LUMEN e ridistribuiti alle varie applicazioni su PC e Smartphone per gestire in tempo reale comandi di telecontrollo, smart metering, automazione, rilevamenti ecc.

Ogni nodo è composto da una scheda a microprocessore che può essere interfacciata a più di cinquanta tipi di sensori per la rilevazione e la misurazione di grandezze fisiche.

Tramite la funzione hibernate a basso consumo, è possibile aumentare la durata delle batterie, necessarie per applicazioni energeticamente autonome da uno a tre anni.

- Temperatura, umidità, flusso idrodinamico
- Sensori di presenza e prossimità
- Torsione, piegamento, rottura, impatto e vibrazione
- Qualità dell'aria, polveri sottili, gas nocivi e vapori di solventi
- Accensione/spegnimenti apparati elettrici
- Consumo di corrente e livello di tensione
- Reti di sensori di parcheggio libero/occupato
- Sensori per fenomeni metereologici

## 1.6 Android

Android è una piattaforma open-source per telefoni cellulari, basata sul sistema operativo Linux e sviluppata dall'Open Handset Alliance. La piattaforma ha a disposizione delle librerie dedicate, come il database SQLite o SGL e OpenGL per la grafica, un application framework, la Dalvik virtual machine (una Java virtual machine modificata) come ambiente di runtime e una serie di applicazioni preinstallate come un browser, una rubrica e un calendario. Considerando le caratteristiche di questo sistema operativo possiamo notare come sia compatibile con molte delle tecnologie attualmente presenti:

- **Adattabilità:** La piattaforma è adattabile a tutte le maggiori librerie grafiche VGA in 2D e le librerie grafiche in 3D basate su specifiche OpenGL ES 1.0.
- **Storage:** Il software per database SQLite è usato per lo storage dei dati.
- **Connettività:** Android supporta una vasta gamma di tecnologie per la connettività, come GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, e Wi-Fi.
- **Messaggistica:** Android supporta una vasta gamma di tecnologie per la messaggistica, come SMS, MMS, ed XMPP (basato su Smack di JiveSoftware).
- **Supporto Multimediale:** Android supporta tutti i maggiori formati di file audio/video/immagini, come: MPEG-4, H.264, MP3, AAC, AMR, JPEG, PNG e GIF.
- **Web Browser:** Il browser integrato in Android è basato sul framework open source WebKit (lo stesso di Safari, per intenderci).
- **Supporto hardware aggiuntivo:** Android permette di utilizzare videocamere, fotocamere, touchscreen, GPS, accelerometri ed altri componenti.
- **Sviluppo:** Android include un emulatore di dispositivi, strumenti per debugging, memoria, performance e un plug-in per l'IDE Eclipse.

L'architettura di Android si presenta suddivisa in diversi livelli, come mostrato in Figura 1.15.

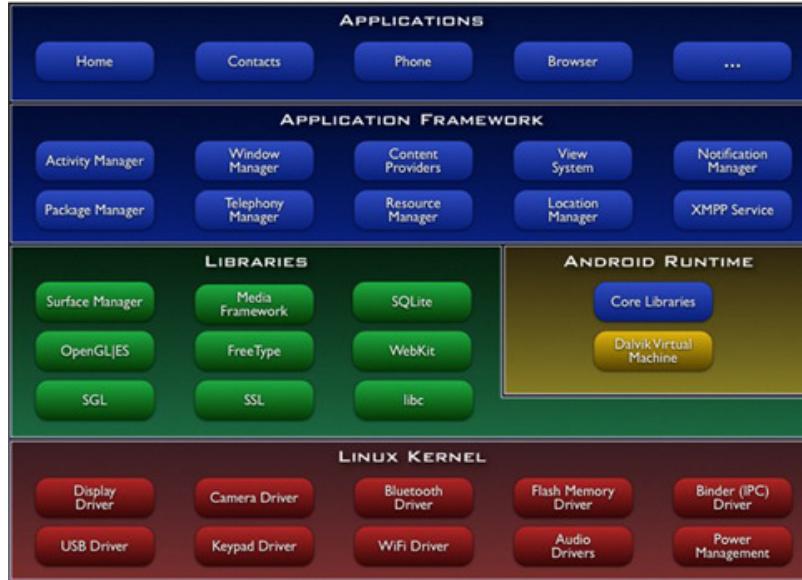


Figura 1.15: Architettura Android

Alla base dello stack Android troviamo un kernel Linux nella versione 2.6. La scelta di una simile configurazione è nata dalla necessità di disporre di un vero e proprio sistema operativo che fornisse gli strumenti di basso livello per la virtualizzazione dell'hardware sottostante attraverso l'utilizzo di diversi driver. A differenza di un kernel Linux standard per Android sono stati aggiunti ulteriori moduli come:

- **Binder IPC Driver**, un driver dedicato che permette a processi di fornire servizio ad altri processi attraverso un insieme di API di più alto livello rispetto a quelle presenti su un kernel linux standard, ciò permette la comunicazione tra processi con un costo computazionale minore e un relativo minore consumo di batteria;
- **Low Memory Killer**, un sistema che si occupa di terminare i processi liberando così spazio nella memoria centrale per soddisfare le richieste di un altro processo. La terminazione dei processi avviene secondo un sistema di ranking che assegna dei

punteggi ai processi; i processi che vengono terminati sono quelli con punteggio più alto. Ad esempio, un processo che controlla la UI (User Interface) di un'applicazione visibile sarà sicuramente più basso di quello relativo ad un'applicazione non visibile sullo schermo. Il processo init non può essere terminato;

- **Ashmem**, sistema di memoria condiviso anonimo (Anonymous Shared Memory) che definisce interfacce che consentono ai processi di condividere zone di memoria attraverso un nome. Il vantaggio di Ashmem rispetto ai sistemi Linux che fanno uso della memoria condivisa, è che viene fornito al kernel uno strumento per recuperare dei blocchi di memoria non utilizzati;
- **RAM Console e Log devices**, per agevolare il debugging, Android fornisce la capacità di memorizzare i messaggi di log generati dal kernel in un buffer RAM. È disponibile inoltre un modulo separato che permette ai processi utente di leggere e scrivere messaggi di log;
- **Android Debug Bridge**, uno strumento che permette di gestire in maniera versatile un'istanza dell'emulatore o eventualmente di un dispositivo reale;
- **Power Management**, sezione progettata per permettere alla CPU di adattare il proprio funzionamento al fine di non consumare energia se nessuna applicazione o servizio ne fa richiesta.

Esistono poi delle librerie native, realizzate in C-C++, che si trovano sopra al livello kernel e che rappresentano il vero e proprio core del sistema:

- **Surface Manager**: il modulo che gestisce le View, cioè i componenti di una interfaccia grafica. Funziona praticamente come uno scheduler per la gestione della visualizzazione delle interfacce grafiche e previene eventuali problemi di accavallamento scoordinato sul display. E' un componente di vitale importanza per un terminale che fa delle interfacce grafiche un punto di forza del suo funzionamento;

- **OpenGL SE**: è una libreria grafica, versione light della libreria OpenGL per terminali mobili, che permette di utilizzare grafica 3D;
- **SGL (Scalable Graphics Library)**: libreria scritta in C++ che permette di gestire la grafica 2D;
- **Media Framework**: componente in grado di gestire i contenuti multimediali (Codec per l'acquisizione e riproduzione di contenuti audio e video);
- **FreeType**: componente che gestisce i font;
- **SQLite**: Database relazionale piccolo ed efficiente messo a disposizione dello sviluppatore per la persistenza dei dati nelle varie applicazioni sviluppate;
- **WebKit**: browser engine open source (utilizzato anche per i più conosciuti browser web Safari e Chrome). Da sottolineare che non si tratta di un browser web, quindi deve essere integrato nelle diverse applicazioni;
- **SSL**: si tratta della famosa libreria per la gestione del Secure Socket Layer.

Le applicazioni Android vengono sviluppate in Java. Serve pertanto una Macchina Virtuale in grado di eseguire le applicazioni. Ne è presente infatti una ottimizzata dalla Oracle (la Dalvik Virtual Machine) per l'esecuzione di software su sistemi con risorse limitate, quali possono essere ad esempio i dispositivi portatili come gli smartphone. Tale macchina virtuale è in grado di eseguire codice contenuto in file con estensione .dex, generati a partire dal bytecode Java. I file .dex, hanno un tipo di compressione che permette di dimezzare lo spazio utilizzato rispetto ai file .jar non compressi. Anche la Dalvik Virtual Machine, similmente per quanto accade per la JVM, implementa un Garbage Collector, liberando pertanto lo sviluppatore dall'onere della gestione della memoria.

Questo sistema operativo è adatto per essere installato su smartphone, tablet o altri dispositivi mobile che possono essere integrati con sistemi automatizzati:

NerdyDog è un'azienda italiana che ha prodotto un'applicazione per Android con la quale è possibile gestire e controllare in remoto la propria abitazione interagendo con una scheda Arduino. Basta collegarsi sul sito, registrarsi e creare il codice da integrare sul microcontrollore in modo molto semplice ed intuitivo. Poi scaricare l'applicazione e collegarsi con il proprio account e gestire in modo semplice e veloce ciò che si è creato [36].

Simen Svale Skogsrud [37] ha creato un'applicazione con la quale è possibile controllare attraverso un pc una macchina a controllo numerico industriale [38].

Anche nell'ambito del gaming è possibile utilizzare questi dispositivi. Con questa applicazione è possibile simulare fino a quattro joypad della playstation 3 in modo che in mancanza di controller fisici, è possibile utilizzare il proprio smartphone per sopperire a tale assenza [39].

# **Capitolo 2**

## **Architettura e implementazione**

### **2.1 Introduzione**

In questo capitolo vengono descritti e analizzati i componenti e i dispositivi utilizzati per la realizzazione del sistema di Ambient Intelligence per il monitoraggio dello stato e dei componenti all'interno di un ufficio. In particolare nella sezione 2.2 verranno presentate le schede Arduino progettate. Nella sezione 2.3 saranno presentati i modelli di rete utilizzati per la comunicazione tra i diversi componenti coinvolti nell'architettura, tra cui il server centrale, spiegato nella sezione 2.4. Nella sezione 2.5 viene presentato il client Android con il quale monitorare lo stato dell'ambiente in remoto. Nella sezione 2.6 viene presentato il modello d'intelligenza utilizzato per classificare lo stato dell'ambiente. Infine nella sezione 2.7 si farà riferimento all'integrazione con gli attuatori installati all'interno della stanza presa in considerazione.

Le architetture potenziali per sviluppare una rete WSN in un sistema di Ambient Intelligence sono molteplici. Per le nostre necessità sono stati utilizzati dei sensori per il monitoraggio ambientale, per tener traccia e visualizzare dati come temperatura, umidità, gas e luce. È stata realizzata una scheda di riconoscimento sulla quale sono presenti un lettore RFID (Radio Frequency IDentification), un sensore PIR (Passive Infrared sensor) di

movimento ed un microfono ad alta sensibilità. L'ultima scheda progettata è invece dedicata all'integrazione con gli attuatori per simulare dispositivi reali. Queste schede formano un architettura generica, modulare, evolubile, che può essere potenzialmente inserita e adattata a qualsiasi contesto. Nel nostro caso abbiamo cercato di riprodurre in un laboratorio della palazzina un ambiente d'ufficio nel quale abbiamo inserito i dispositivi spiegati nelle sezioni successive.

## 2.2 Architettura del sistema di Ambient Intelligence

Il sistema illustrato in figura 2.1 è formato da quattro tipologie di schede realizzate tramite la piattaforma Arduino (tre schede Arduino UNO e una Mega ADK). Le schede Arduino sono state utilizzate per creare i seguenti dispositivi:

- una scheda Arduino UNO di monitoraggio ambientale, con la quale possiamo raccogliere i dati provenienti da un sensore di temperatura e di umidità, un sensore di fumo ed un sensore di luce ad alta precisione. I dati raccolti con questa scheda vengono poi inviati al server, grazie ad un modulo WiFi, in formato JSON tramite HTTP;
- una scheda Arduino UNO di riconoscimento con un lettore RFID per la lettura di tag RFID e NFC, un sensore di suono ad alta sensibilità e un sensore PIR di movimento. I dati raccolti vengono inviati attraverso un modulo Xbee Serie 1 sotto forma di stringa alfanumerica;
- una scheda Arduino UNO per la comunicazione ZigBee (sempre attraverso un modulo Xbee) che raccoglie i dati della scheda descritta precedentemente, ne ricava una stringa JSON e la invia, con una richiesta HTTP, al server attraverso un Ethernet Shield.
- L'Arduino Mega ADK invece è stato utilizzato per montare la scheda per gli attuatori e per l'USB Host. Anch'essa comunica con il server con un modulo Ethernet.

Nei seguenti paragrafi vengono spiegate in dettaglio tutte le schede sopra presentate.

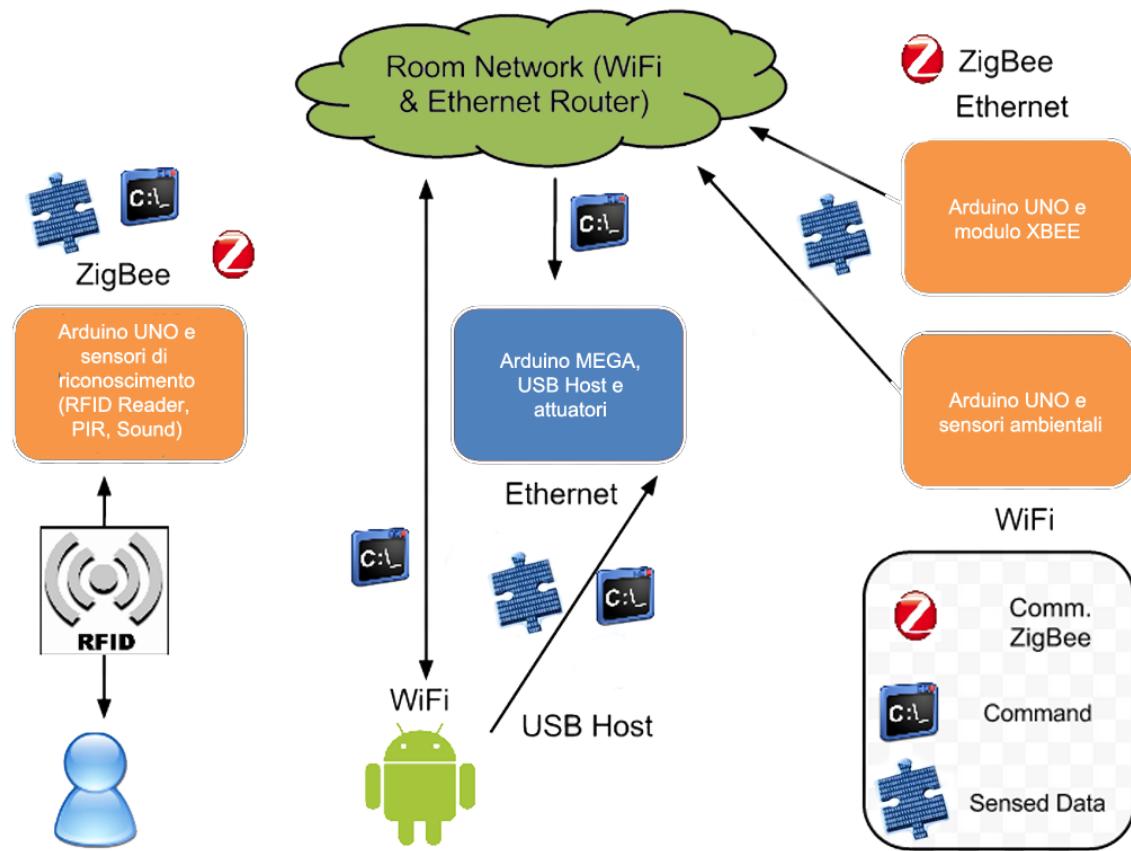


Figura 2.1: Architettura generale del sistema considerato

### 2.2.1 Monitoraggio ambientale

L'idea si basa sulla costruzione di una scheda 2.7 che raccolga a intervalli regolari i dati provenienti dai sensori installati sulla scheda Arduino, dislocata in un ambiente lavorativo, in grado di interfacciarsi con un server e che raccolga le segnalazioni a intervalli regolari.

Il funzionamento della scheda si può suddividere in tre fasi:

- Determinazione della tipologia di dati e sensori
- Raccolta dati
- Trasmissione dati

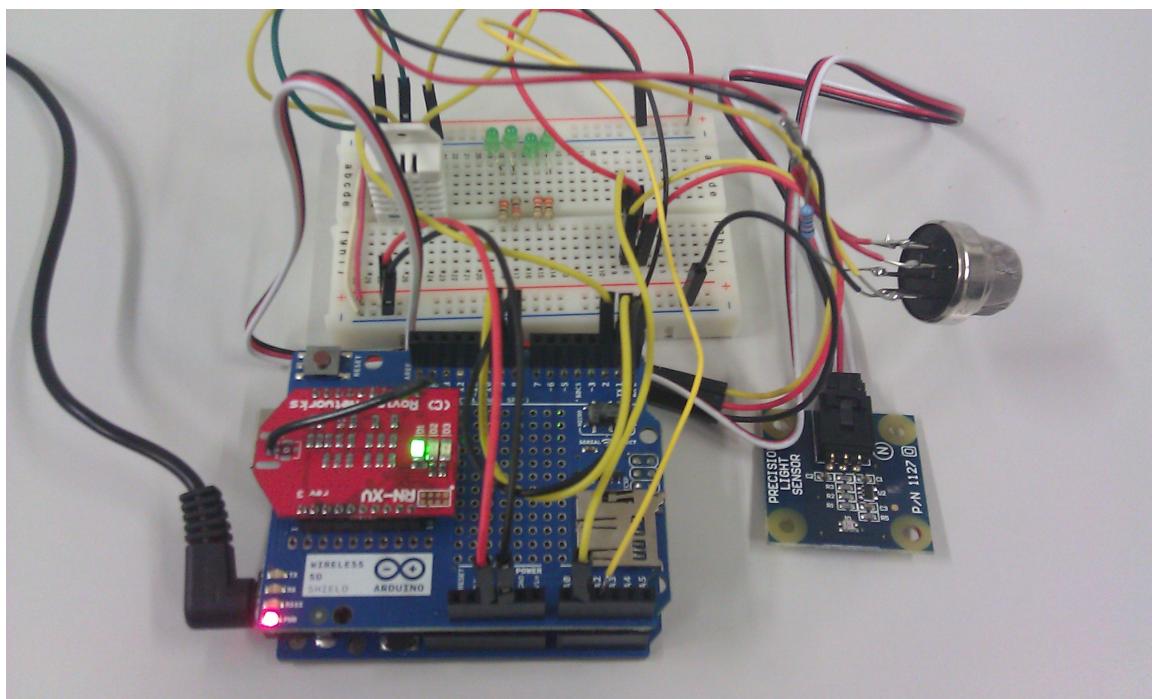


Figura 2.2: Scheda per il rilevamento ambientale realizzata in laboratorio

Per un primo approccio si è deciso di utilizzare alcuni semplici sensori per rilevare i dati più significativi all'interno di un ufficio:

- Temperatura
- Umidità
- Luce
- Fumo

La raccolta dati viene effettuata da una scheda Arduino Uno, sulla quale è inserito uno sketch per la rilevazione dei seguenti sensori:

Il **DHT22**(Fig. 2.3) è un sensore a basso costo che misura temperatura e umidità con un'interfaccia digitale ad un filo. Il sensore è già calibrato e non richiede ulteriori componenti per funzionare.

Caratteristiche: 3.3-6V Input, 1-1.5mA measuring current, 40-50 uA standby current, Humidity from 0-100% RH, -40 - 80 degrees, temperature range, +-2% RH accuracy, +-0.5 degrees C.

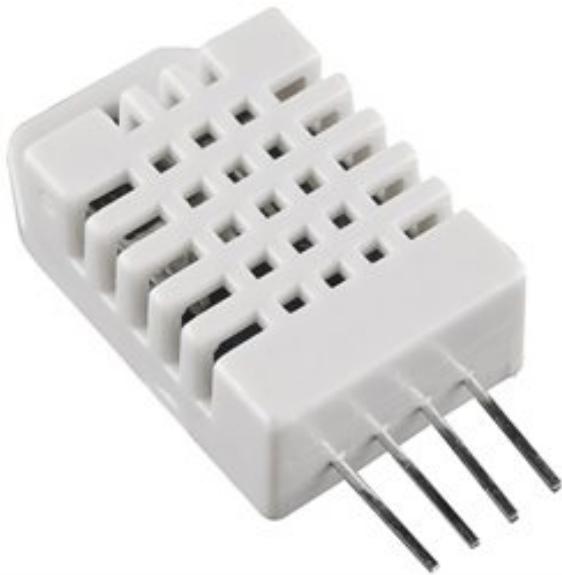


Figura 2.3: Sensore DHT22

Con Arduino la lettura di questo sensore è molto semplice. Questo dispositivo è fornito di una libreria direttamente scaricabile dal playground di Arduino [40]. Come possiamo vedere nel codice seguente, bisogna salvare in due variabili float i valori della temperatura e dell'umidità. Questo sensore ha un incovento: ogni lettura deve avvenire come minimo ogni 2 secondo perché è un sensore molto lento, altrimenti si rischia di ottenere un risultato incorretto.

```
void readDHTSensor() {
    float t = dht.readTemperature();
    float h = dht.readHumidity();

    // check if returns are valid, if they are NaN (not a number) then something
    // went wrong!
    if ((isnan(t))||(isnan(h))) {
```

```

    Serial.println("Failed to read from DHT");
} else {
    temp_accum=t;
    hum_accum=h;
}
}

```

Il **Precision Light Sensor** (Fig. 2.4) è un sensore che misura il livello della luce visibile (Unità di misura: Lux). Il range di misurazione parte da 1lux (luce della luna) fino a 1000Lux (illuminazione di uno studio televisivo).

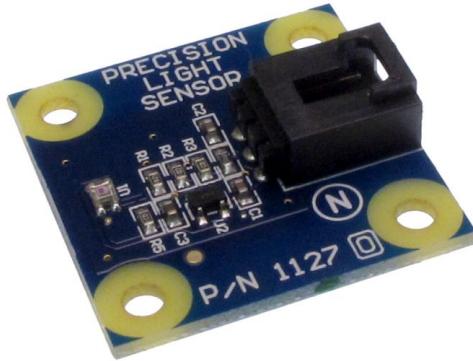


Figura 2.4: Sensore di precisione di luminosità

Il sensore si connette ad un ingresso analogico dell'Arduino che utilizza un convertitore analogico-digitale a 10bit. C'è quindi la necessità di convertire il valore letto attraverso la formula mostrata nel codice seguente:

```

int readLightSensor() {
    int luce = analogRead(Light_PIN);
    luce = (5.0 * luce * 200.0) / 1024.0;
    return luce;
}

```

Lo **Smoke Sensor MQ-2** (Fig. 2.5) è un sensore di gas con un'elevata sensibilità e tempi rapidi di risposta. Questo tipo di sensore viene utilizzato in apparecchiature per il

rilevamento di perdite di gas ed è adatto alla rilevazione di perdite di metano, GPL, isobutano, propano, alcool etilico, idrogeno e fumo. Il sensore varia la resistenza in dipendenza della concentrazione di gas nell'aria. E' necessario fornire una tensione di alimentazione all'elemento riscaldante interno, lasciar riscaldare il sensore e leggere la resistenza. Il funzionamento è identico a quello degli altri sensori della serie MQ, con la differenza che questo tipo è adatto a rilevare tutti i gas in maniera specifica. In questa tesi, il sensore MQ-2 è stato utilizzato per misurare la concentrazione di anidride carbonica nell'aria.

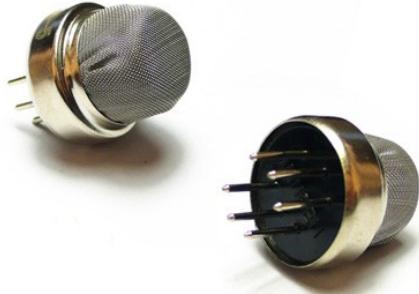


Figura 2.5: Sensore di fumo

Come per il sensore precedente, l'MQ-2 si connette ad un ingresso analogico. Esso necessita di una resistenza sul pin per limitare l'ingresso di Arduino e per filtrare il risultato. Nel nostro lavoro abbiamo voluto misurare la concentrazione di CO<sub>2</sub> all'interno dell'ambiente.

```
float readSmokeSensor() {  
    int val = analogRead(Smoke_PIN);  
    float Ro = 12000.0; //Resistenza da 12kohm  
  
    float Vrl = val * ( 5.00 / 1024.0 ); // V  
    float Rs = 20000 * ( 5.00 - Vrl ) / Vrl; // Ohm  
    float ratio = Rs/Ro;  
  
    //CONCENTRAZIONE DI CO2
```

```
float ppm = 0.0;  
ppm = 37143 * pow (ratio, -3.178);  
return ppm;  
}
```



Figura 2.6: Schema di comunicazione della scheda

Gli errori a livello del sensore possono verificarsi in due occasioni:

- Durante la misurazione, nella circuiteria interna
- Durante la trasmissione del dato ad Arduino

Possiamo trascurare il secondo punto perché nella comunicazione seriale il sensore inserisce un byte di CRC per il controllo degli errori. Per il primo punto, tuttavia, non possiamo avere delle certezze e ci affideremo solamente alla taratura effettuata dal produttore.

In figura 2.7 possiamo vedere lo schema completo:

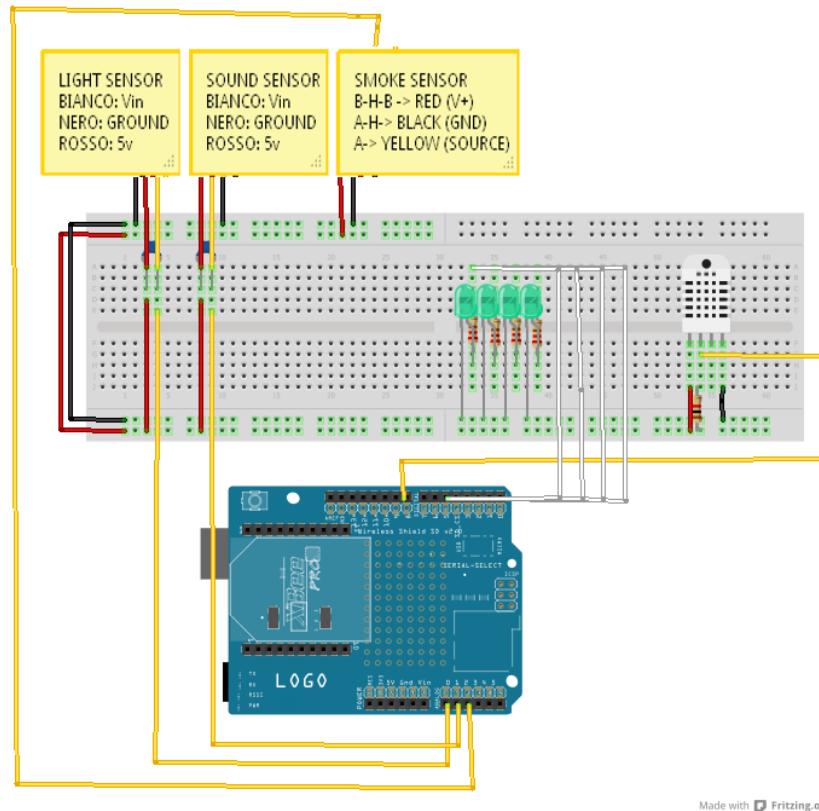


Figura 2.7: Schema circuitale della scheda

I condensatori ai sensori sono da 10uF, le resistenze ai Led da 330ohm, la resistenza al DHT è da 1Kohm.

### 2.2.2 Scheda di riconoscimento

Questa scheda è stata progettata per un monitoraggio ambientale: l'obiettivo è determinare la presenza di utenti all'interno di una stanza. Essa è in grado di rilevare il passaggio, all'interno della stanza, di utenti che sono identificati da un TAG univoco e di rilevare l'ingresso di utenze non identificate, segnalandole attraverso una rilevazione dei sensori di movimento e di suono.

I dati significativi rilevati nell'ufficio sono:

- Presenza fisica all'interno dei locali

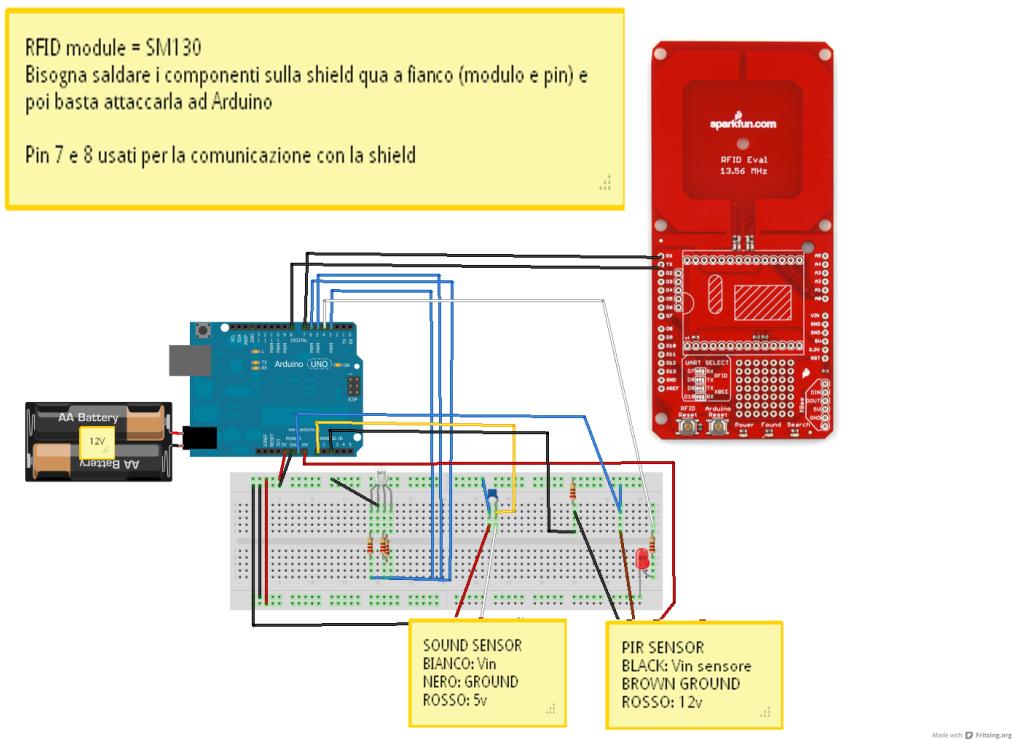


Figura 2.8: Schema circuitale della scheda

- Rumore
- Identificazione attraverso un sistema di riconoscimento univoco

La raccolta dati viene effettuata da una scheda Arduino Uno, sulla quale è inserito uno sketch per la rilevazione delle varie grandezze. Le letture del sensore del suono sono eseguite come per la scheda descritta in precedenza. I sensori PIR e l'RFID sono sempre in “attesa” di un evento e non appena questo viene rilevato, viene subito inviato al server un dato che viene successivamente elaborato.

I sensori utilizzati sono:

#### **PIR motion sensor(Fig. 2.11)**

Il sensore PIR permette di catturare un movimento, utile appunto per determinare la presenza di una persona all'interno di un ambiente. Si tratta di un dispositivo di piccole

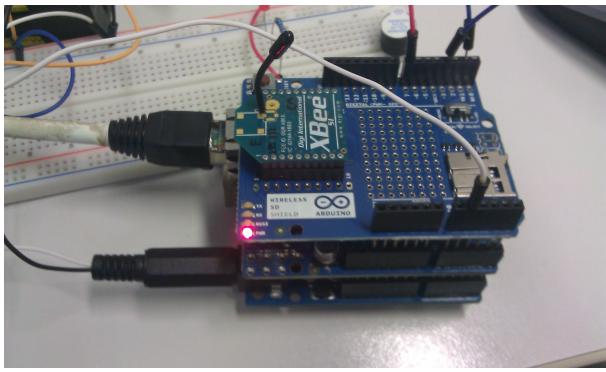


Figura 2.9: Scheda con Xbee ricevente e invia su Ethernet realizzata.

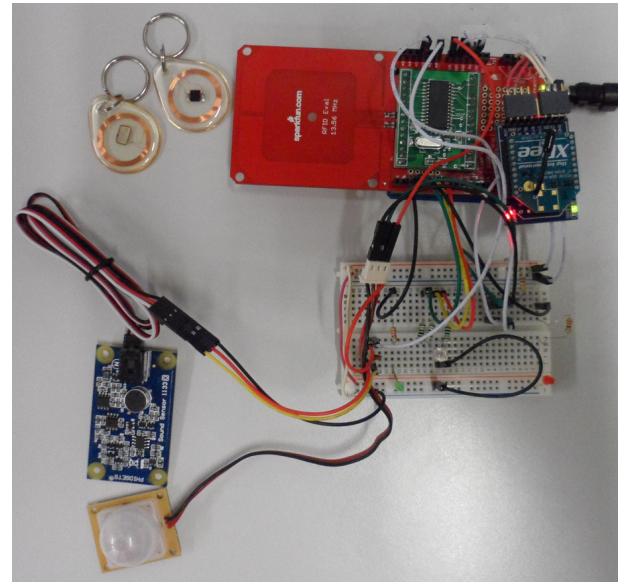


Figura 2.10: Scheda di riconoscimento con Xbee mittente realizzata.

dimensioni, economico, a basso consumo e semplice da utilizzare. Per questa ragione è spesso utilizzato in applicazioni domotiche o industriali. Il sensore PIR è costituito da un sensore piroelettrico capace di catturare diversi livelli di radiazioni infrarosse. Il rilevatore di movimento è formato da due parti collegate in modo che una prevalga sull'altra in base alle radiazioni IR rilevate. Sulla base di ciò, l'uscita è alta o bassa.

Il sensore piroelettrico ha un gruppo di circuiti di supporto, formato da resistenze e condensatori. I sensori più piccoli utilizzano il chip BISS0001 (Micro Potenza di movimento PIR IC), poco costoso. Questo chip utilizza l'uscita analogica del sensore ed esegue alcune operazioni secondarie su di essa per emettere un impulso digitale.

Per molti progetti di base o prodotti che necessitano di rilevare quando una persona ha lasciato oppure sta entrando nell'area di rilevazione, sono necessari sensori PIR più grandi. Si noti che con il sensore PIR non si riesce a sapere quante persone ci sono nell'ambiente o quanto sono vicini al sensore. Il sensore che è stato utilizzato è un semplice sensore di movimento PIR. Basta alimentarlo ed attendere 1-2 secondi affinchè il sensore esegua

uno *snapshot* dell'ambiente in cui si trova. Se vengono rilevati movimenti all'interno della stanza, il pin contrassegnato con *alarm* va al livello logico basso. Esso necessita di una tensione di alimentazione da 7 a 12V. Siccome l'allarme ha un'uscita a collettore aperto, è necessario inserire una resistenza di pull-up su tale filo.



Figura 2.11: PIR motion sensor

Il seguente codice mostra come in Arduino è possibile rilevare un movimento attraverso la lettura di un dato analogico proveniente dal sensore sopra citato. Il PIR utilizzato necessita di un periodo di settaggio tra una lettura e un'altra. Per evitare falsi positivi si utilizza un contatore di movimenti che, se supera una certa soglia, rileva correttamente il movimento.

```
if(time>readyPIR+timerPIR){  
    alarmValue = analogRead(alarmPin);  
  
    if (alarmValue < 100) count++; //Se l'uscita e' bassa ha rilevato un  
    movimento  
    else count=0;  
  
    if (count>5){  
        Serial.println("MOVIMENTO!!");  
        sendXbee(101,id_PIR);  
    }  
}
```

```
    blinky(ledPir);  
    count=0;  
}  
readyPIR=millis();  
}
```

### RFID shield(Fig. 2.12)

RFID significa Radio Frequency IDentification ed è un sistema di identificazione attraverso appositi TAG che possono essere assegnati a oggetti, animali o persone. Questo sistema richiede un lettore ed uno o più TAG (o transponder) che non sono altro che dispositivi montati sull'oggetto da riconoscere. Quando il TAG è in prossimità del lettore, invia un identificativo univoco (solitamente in formato esadecimale). La portata di questo sistema è di pochi centimetri, ma, grazie ad antenne, è possibile anche arrivare a distanza di qualche metro: tale caratteristica rende questa tecnologia particolarmente interessante. Facciamo un paio di esempi per semplificare il concetto. Un possibile utilizzo è in un ambiente di lavoro: all'ingresso dell'ufficio viene collocato un lettore RFID e viene assegnato a ciascun dipendente un badge con un TAG RFID d'identificazione. Quando il dipendente entra o esce, passa il badge sul lettore che registra l'ingresso e l'uscita. Un altro esempio possiamo trovarlo nelle industrie. In una catena di produzione, viene posto su ogni oggetto da catalogare un TAG che ne permette la registrazione. Tutte le volte che tale pezzo esce dal magazzino, passa sotto un lettore RFID che identifica l'uscita di quel particolare pezzo, aggiornando un database. Nel nostro caso, al passaggio di un TAG, il dispositivo legge e invia il dato attraverso la rete Xbee.

Nel seguente pezzo di codice è mostrato come attendere una lettura di un TAG RFID. Ogni TAG inizia con il codice FF (in esadecimale equivale a 255), mentre i successivi 11 byte saranno univoci e salvati in un buffer.

```
void parse() {
    while(rfid.available()) {
        if(rfid.read() == 255) {
            for(int i=1;i<11;i++) {
                Str1[i]= rfid.read();
            }
        }
    }
}
```

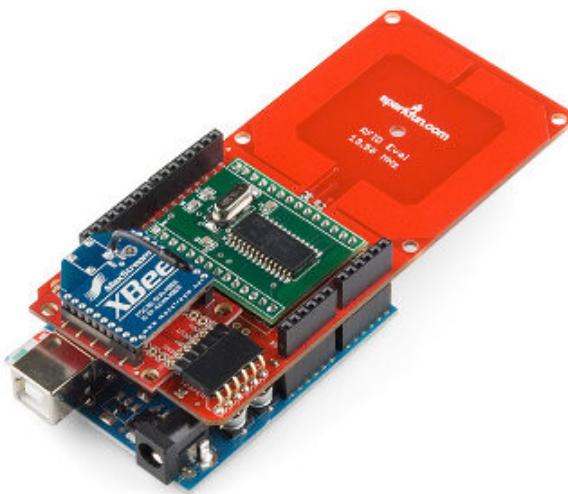


Figura 2.12: RFID shield già montata sopra la scheda Arduino

#### Precision sound sensor (Fig. 2.13)

Il sensore di suoni è un sensore analogico che fornisce un valore proporzionale al suono rilevato. Il sensore rileva suoni fino a circa 100 dB. È un sensore a largo spettro di sensibilità che lavora dai 100Hz ai 8kHz con un errore di circa 3 dB: la parte bassa dello spettro (5-%) rileva i rumori di sottofondo, dal 5% al 10% i rumori di una persona in lontananza che parla, dal 10% al 30% il rumore per una normale conversazione tenuta vicino al sensore e dal 30% al 100% i rumori per persone che urlano o musica ad alto volume.

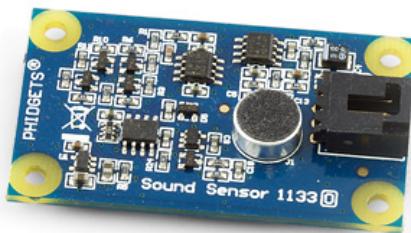


Figura 2.13: Precision sound sensor

Siccome il sensore è letto molto frequentemente, è stato necessario campionare la sua lettura. Ogni volta che scade il timeout per il periodo di lettura (circa ogni secondo), viene letto il valore del sensore e salvato. Una volta richiesta la lettura del sensore dall'esterno, viene inviata la media dei valori precedentemente letti. Per il calcolo del valore in dB (deciBel) è necessaria la formula riportata nello sketch.

```
if(time > ReadTime + Periodo_Lettura_Sensore){  
    ReadTime = millis();  
    sound_accum += readSoundSensor();  
  
    n_camp++;  
    Serial.print("Campione : ");  
    Serial.print(n_camp);  
    Serial.print(" - ");  
    Serial.println(time);  
}  
  
//RESTITUISCO IL VALORE DEL SENSORE IN BASE AI CAMPIONAMENTI PRECEDENTI  
int readSoundSensor(){  
    int value = analogRead(Sound_PIN);  
    int suono = 16.801 * log(value) + 9.872; //calcolo per il corretto  
    valore in dB  
    return suono;  
}
```

Il sensore PIR necessita di un'alimentazione obbligatoria a 12V (quindi non è utilizzabile l'alimentazione USB) altrimenti il piroelettrico all'interno del sensore riceve impulsi scorretti e quindi rilevazioni sbagliate. Gli altri sensori sono precisi e non necessitano di una configurazione.

### 2.2.3 Mega ADK e Attuatori

Questa scheda 2.16 è dedicata sia al riconoscimento di un dispositivo Android attraverso il “plug-in” sulla scheda grazie all’USB Host e all’azionamento degli attuatori. Tramite l’utilizzo di questa scheda si è voluto simulare un classico ambiente dove sono presenti impianto d’illuminazione, impianto d’aerazione, automazione degli infissi, ecc.

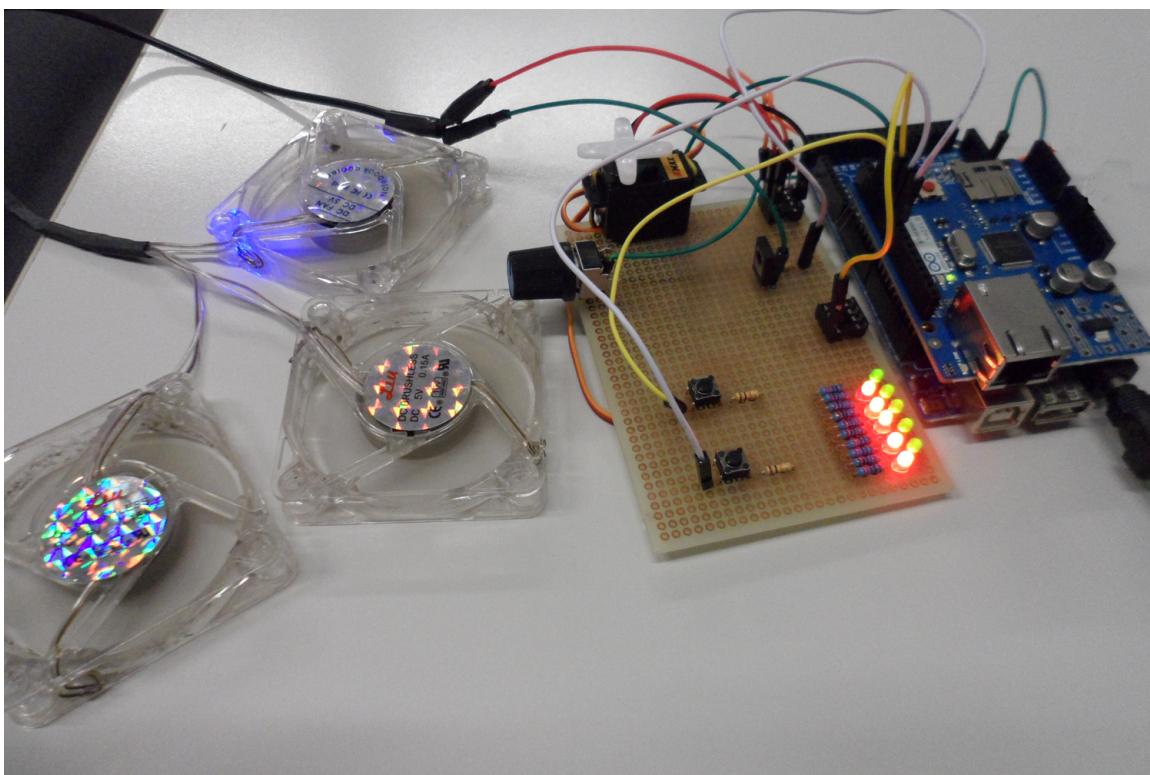


Figura 2.14: Scheda con prototipi di attuatori realizzata in laboratorio

I Componenti utilizzati sono: Arduino MEGA ADK, un transistor NPN, bottoni, resistori da 1k e da 10k, un led, ventole 5v, servo motore, potenziometro, cavi.

Arduino fornisce solamente pochi mA in uscita mentre invece è stato necessario alimentare le ventole che assorbono 500mA. Ciascuno dei Pin di Arduino può essere usato per alimentare dispositivi che usano fino a 40mA: si tratta di una quantità di corrente molto piccola, sufficiente appena per illuminare un Led. Se si prova a gestire qualcosa come

un motore, il Pin smette di funzionare e potrebbe anche bruciare l'intero processore. Per gestire carichi maggiori, come motori o lampade a incandescenza, dobbiamo usare un componente esterno che sia in grado di accendere e spegnere questo genere di componenti e che sia alimentato da un Pin di Arduino. I dispositivi di questo genere possono essere sia Relè di potenza ma anche transistor MOSFET. Questi ultimi sono interruttori elettronici che possono essere alimentati applicando un voltaggio su uno dei loro tre Pin, ognuno dei quali è detto gate. Concettualmente il funzionamento è simile al classico interruttore domestico della luce, sostituito in questa situazione da un pin sulla scheda Arduino, il quale invia il voltaggio opportuno al gate del MOSFET.

Nello schema 2.15 è possibile vedere l'utilizzo di un MOSFET come l'IRF520 per accendere e spegnere un piccolo motore collegato a un ventilatore. Si noti anche che il motore viene alimentato dal connettore Vin/9V sulla scheda Arduino. Questo è un altro vantaggio offerto dai MOSFET: permettono infatti di gestire dispositivi la cui alimentazione è diversa da quella usata da Arduino. Dato che il MOSFET è collegato al Pin 9, per controllare la velocità del motore via PWM possiamo anche usare analogWrite().

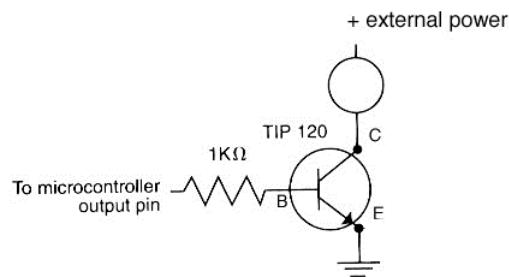


Figura 2.15: Come collegare il transistor per funzionare con Arduino

Quando che il MOSFET è collegato, basta trattare il dispositivo come se fosse un LED. È poi il transistore a fare il resto.

Per identificare un dispositivo connesso all'USB Host abbiamo fatto uso della libreria fornita direttamente da Android in collaborazione con Arduino, *UsbHost library for Ardui-*

no (che include la classe AndroidAccessory), con la quale è possibile identificare in modo univoco un dispositivo connesso. Tuttavia, i dispositivi che si collegano all'USB Host devono disporre dell'accesory mode (alcuni dispositivi che hanno questa funzionalità sono il Nexus S, il Nexus One e i Samsung Galaxy Tab). Il codice sviluppato per questa parte è il seguente:

```
...
//CONTROLLI PER VERIFICARE CHE UN DISPOSITIVO SIA STATO CONNESSO
if(usb_android.isConnected() && !conn) {
    conn = true;
    if (!usbcheck()) Serial.println("USB connection test failed.");
}

if(!usb_android.isConnected()) {
    conn = false;
}
...

/* Test USB connectivity */
bool usbcheck() {
    bool exit=false;
    byte rcode;
    byte usbstate;
    Max.powerOn();
    delay( 200 );
    Serial.println("\r\nUSB Connectivity test. Waiting for device connection...
");
    while(!exit) {
        delay( 200 );
        Max.Task();
        Usb.Task();
        usbstate = Usb.getUsbTaskState();
        //--DEBUG Serial.println(usbstate);
    }
}
```

```
switch( usbstate ) {  
    case( USB_ATTACHED_SUBSTATE_RESET_DEVICE ):  
        Serial.println("\r\nDevice connected. Resetting");  
        break;  
    case( USB_ATTACHED_SUBSTATE_WAIT_SOF ):  
        Serial.println("\r\nReset complete. Waiting for the first SOF...");  
        break;  
    case( USB_ATTACHED_SUBSTATE_GET_DEVICE_DESCRIPTOR_SIZE ):  
        Serial.println("\r\nSOF generation started. Enumerating device.");  
        break;  
    case( USB_STATE_ADDRESSING ):  
        Serial.println("\r\nSetting device address");  
        break;  
    case( USB_STATE_RUNNING ):  
        Serial.println("\r\nGetting device descriptor");  
        rcode = getdevdescr( 1 );  
        exit = true;  
        if( rcode ) {  
            Serial.print("\r\nError reading device descriptor. Error code ");  
            //print_hex( rcode, 8 );  
            Serial.print( rcode, OCT );  
        }  
        else {  
            Serial.println("\r\nAll tests passed.");  
            Usb.setUsbTaskState(0xF0);  
        }  
        break;  
    case( USB_STATE_ERROR ):  
        Serial.println("\r\nUSB state machine reached error state");  
        break;  
    default:  
    {  
        break;  
    }
```

```
    }
} //switch
} //while(1)
}
```

Con la seguente funzione viene letto il descrittore del dispositivo appena collegato:

```
byte getdevdescr( byte addr ){
    USB_DEVICE_DESCRIPTOR buf;
    byte rcode;
    rcode = Usb.getDevDescr( addr, 0, 0x12, ( char *)&buf );
    if( rcode ) {
        return( rcode );
    }
    Serial.print("\r\nDevice descriptor: ");
    Serial.print("\r\nDescriptor Length:\t");
    //print_hex( buf.bLength, 8 );
    Serial.print( buf.bLength, OCT );
    Serial.print("\r\nUSB version:\t");
    //print_hex( buf.bcdUSB, 16 );
    Serial.print( buf.bcdUSB, HEX );
    Serial.print("\r\nVendor ID:\t");
    Serial.print( buf.idVendor, HEX );
    //print_hex( buf.idVendor, 16 );
    Serial.print("\r\nProduct ID:\t");
    Serial.print( buf.idProduct, HEX );
    //print_hex( buf.idProduct, 16 );
    Serial.print("\r\nRevision ID:\t");
    Serial.print( buf.bcdDevice, HEX );
    //print_hex( buf.bcdDevice, 16 );
    sendDataToEthernet( buf.idVendor, buf.idProduct, buf.bcdDevice );
    return( 0 );
}
```

Con questa funzione mando i dati letti dall'USB Host al server e li tratto come se fossero

dei TAG:

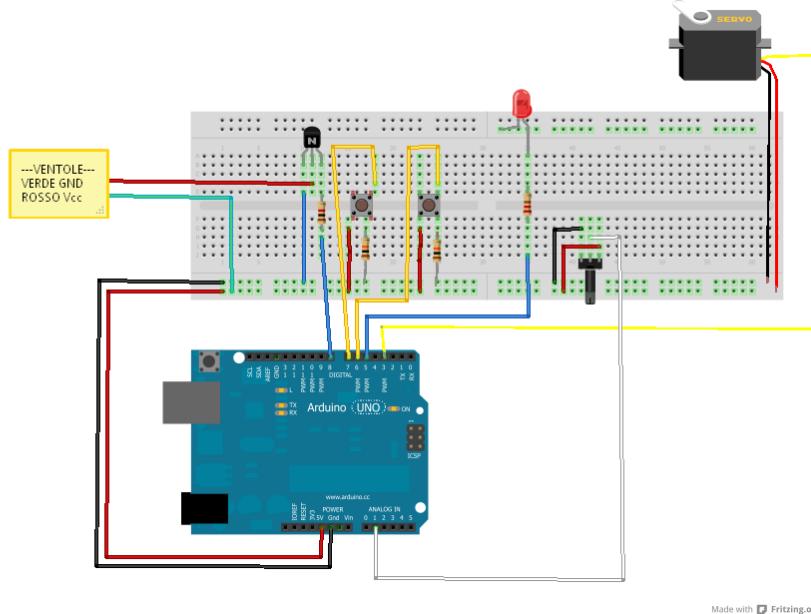
```
void sendDataToEthernet(int idVendor, int idProduct, int bcdDevice){  
    sprintf(timestamp, "%d", random(1, 32000));  
  
    sprintf(jsonMsg, "{\"timestamp\": \"%s\", \"checksum\": \"%s\", \"mac\": \"%s\", \"  
        usbhost\": %d.%d.%d}\\0", timestamp, "5EB63BBBE01EEED093CB22BB8F5ACDC3",  
        "90:A2:DA:0D:1E:52", idVendor, idProduct, bcdDevice);  
    for(i=0; jsonMsg[i]!=0; i++);  
    i++;  
  
    if (client.connect(serverName, serverPort)){ //connessione al server per  
        invio lettura sensore temperatura  
        Serial.println("CONNESSO...");  
        delay(200);  
        InvioJSONRequest(jsonMsg, i);  
        delay(200);  
        client.stop();  
        Serial.println("...FINE INVIO!");  
    }  
}
```

Lo schema della scheda è il seguente:

Per la parte degli attuatori si rimanda alla sezione 2.7.

## 2.3 Networking

I sensori sono dei dispositivi in grado di rilevare determinati fenomeni fisici attraverso l'uso dei trasduttori. Con l'evoluzione della tecnologia, il mercato dei sensori offre non solo prodotti sempre più precisi e affidabili, ma anche funzionalità sempre più sofisticate, e nuovi tipi di modalità di comunicazione, tra cui la possibilità di comunicare attraverso le tecnologie sia wired che wireless.



Made with Fritzing.org

Figura 2.16: Schema circuitale della scheda

### 2.3.1 Ethernet

Ethernet è il tipo di rete locale più diffuso al mondo. Con rete locale s'intende un sistema di collegamento tra diversi computer, collocati all'interno del medesimo edificio, entro edifici contigui oppure nell'arco di pochi chilometri nel caso in cui non esistano confini di riferimento precisi. Tale sistema consente lo scambio diretto di dati in formato elettronico tra più di due computer, senza ricorrere al passaggio di dischetti. Il numero di stazioni deve essere per lo meno tre perché se i computer fossero soltanto due non si potrebbe più parlare di rete, bensì di collegamento diretto, da punto a punto, come quello che si crea quando si usano particolari tipi di cavo seriale o parallelo per trasferire dati da un portatile a un desktop. La natura generale di qualsiasi LAN (Local Area Network - rete locale) e quella di Ethernet in particolare è di consentire il libero colloquio con qualsiasi macchina collegata e di trasmettere la stessa informazione contemporaneamente a tutte le macchine in ascolto (broadcasting). Ethernet non è necessariamente la migliore delle tecnologie possibili, ma si è dimostrata la più economica e la più facile da utilizzare, il che ne ha decretato l'enorme

successo a tutti i livelli d'impiego e in qualsiasi area geografica del mondo.

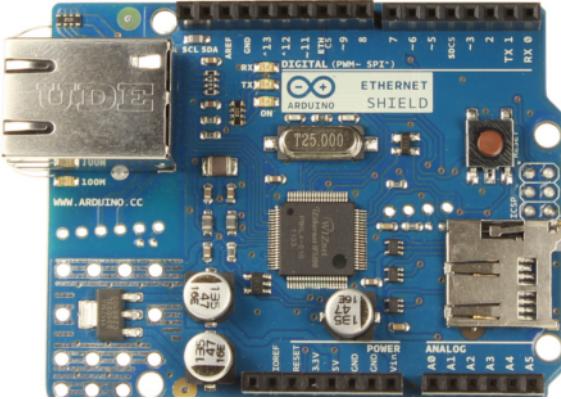


Figura 2.17: Ethernet Shield di Arduino

La shield Ethernet Arduino (Fig 2.17) utilizzata è quella ufficiale, in grado di gestire funzionalità di rete IP con i protocolli TCP e UDP e in grado di gestire fino 4 connessioni simultanee. Questo shield offre un'altra funzione oltre alla comunicazione Ethernet ed è la lettura/scrittura su un MicroSD. Purtroppo questi due componenti dello shield condividono le porte di comunicazione, quindi se abbiamo intenzione di utilizzarli nello stesso programma dobbiamo avere l'accortezza di abilitare/disabilitare l'uno o l'altro a seconda di ciò che intendiamo usare in quel momento.

Per gestire lo shield Ethernet, ci viene incontro la libreria Ethernet fornita da Arduino che mette a disposizione i seguenti comandi principali:

*Ethernet.begin(mac)* : Questo è il comando che setta gli indirizzi dello shield, con la versione 1.0 delle librerie, è stato implementato il supporto al protocollo DHCP, per la configurazione automatica degli indirizzi IP. Dato che ormai da alcuni anni ogni router/modem integra un server DHCP, possiamo tranquillamente appoggiarci a questa funzione così da non dover riprogrammare la scheda per cambiare IP.

*Client.connect(server,porta)*: Con questo comando, si stabilisce la connessione al server di cui si fornisce l'indirizzo o il nome (che viene poi risolto in automatico tramite DNS) e la porta a cui connettersi. Stabilita la connessione è possibile inviare e ricevere i dati.

*Client.print(char[])/ Client.println(char[]):* Sono i principali comandi che vengono utilizzati per comporre i dati da inviare tramite la comunicazione ethernet. Il comando println differisce dal comando print, per l'aggiunta in coda al messaggio di un Line Feed, cioè un “a capo”.

Nel nostro caso abbiamo utilizzato il seguente codice per impostare il client Ethernet. Arduino fornisce direttamente la libreria Ethernet per gestire connessioni TCP.

```
...
EthernetClient client;//dichiaro una variabile globale per il client ethernet
...

void setup() {
    ...
    // start the Ethernet connection:
    Serial.println("...Configuro la connessione ethernet...");
    Ethernet.begin(mac); //configuro lo shield ethernet
    Serial.println("Connessione Configurata.");
    delay(1000); //aspetto un secondo per far avviare lo shield ethernet
    Serial.println("Programma Avviato, Setup Terminato!");
    Serial.print("My IP address: ");
    Serial.println(Ethernet.localIP());
}

void loop() {
    ...
    sprintf(timestamp, "%d", random(1, 32000));

    sprintf(jsonMsg,"{\\"timestamp\\":\\"%s\\",\\"checksum\\":\\"%s\\",\\"mac\\":\\"%s\\",\\"%s
    \":%s}\\0",
    timestamp, "5EB63BBBE01EEED093CB22BB8F5ACDC3", "90:A2:DA:0D:1E:52", sensor,
    valBuffer);

    if (client.connect(serverName, serverPort)){ //connessione al server per invio
```

```
    lettura sensore temperatura
    Serial.println("CONNESSO...");
    delay(200);
    InvioJSONRequest(jsonMsg, i);
    delay(200);
    client.stop();
    Serial.println("...FINE INVIO!");
}
}
```

La seguente funzione serve per inviare al server la stringa JSON contenente i dati letti.

Ogni volta che invia un dato attende la risposta dal server.

```
void InvioJSONRequest(char* jsonString, int lungh) {
    client.print("POST /dsgm2m/api/m2m.php HTTP/1.0\r\n");
    client.print("Content-Length: ");
    client.print(lungh);
    client.print("\r\n\r\n");
    client.print(jsonString);

    Serial.println("...RISPOSTA...");

    boolean currentLineIsBlank = true;
    int exit=0;
    int i=0;

    if(richiedorisposta==1) {
        while(client.connected()) {
            if(client.available() > 0) {
                char c = client.read();
                Serial.write(c);
                if(exit==1) {
                    bufferRisp[i] = c;
                    i++;
                }
            }
        }
    }
}
```

```
    }

    // if you've gotten to the end of the line (received a newline
    // character) and the line is blank, the http request has ended,
    // so you can send a reply

    if (c == '\n' && currentLineIsBlank) {
        if(exit==1) {
            break;
        }
        else exit = 1;
    }

    if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
    }
    else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
    }
}

}

}
```

### 2.3.2 Wireless

Con il termine wireless si indica una comunicazione tra dispositivi senza l'uso di cavi. Per estensione sono detti wireless i rispettivi sistemi o dispositivi di comunicazione che implementano tale modalità di comunicazione. I sistemi tradizionali basati su connessioni cablate sono invece detti wired. Generalmente il wireless utilizza onde radio a bassa potenza; tuttavia la definizione si estende anche ai dispositivi, meno diffusi, che sfruttano la radiazione infrarossa o il laser. La motivazione che spinge a realizzare dispositivi e reti wireless è l'abolizione del cablaggio che può essere in alcuni casi fastidioso per sovraccarico di fili nei

dispositivi a corto raggio, ma soprattutto estremamente oneroso a livello economico. Basti pensare alla difficoltà degli operatori (provider) di estendere e rendere ubiquitaria la banda larga in Internet; ciò è dovuto a causa dell'elevato costo della messa in posa dei portanti fisici in relazione ai possibili introiti futuri derivanti dall'investimento. Un singolo dispositivo di ricetrasmissione radio può invece coprire un'ampia zona di utenza ad un prezzo di impianto notevolmente più basso. Reti e dispositivi wireless (radiomobili e reti cellulari) consentono dunque il superamento del vincolo del cablaggio e allo stesso tempo consentono anche la mobilità del servizio all'utente. Il prezzo da pagare rispetto alle comuni reti a dispositivi cablati è quello di una qualità del servizio (QoS) generalmente inferiore e maggiori problematiche relative alla sicurezza della comunicazione e al possibile inquinamento elettromagnetico. All'interno delle reti wireless possiamo trovare sia dispositivi WiFi che ZigBee. Con il termine WiFi (abbreviativo di Wireless Fidelity) si indica la tecnica e i relativi dispositivi che consentono a terminali di utenza di collegarsi tra loro attraverso una rete locale in maniera wireless (WLAN), basandosi sulle specifiche dello standard IEEE 802.11. Con il termine ZigBee si intende il nome di una specifica per un insieme di protocolli di comunicazione ad alto livello che utilizzano piccole antenne digitali a bassa potenza e basato sullo standard IEEE 802.15.4 per wireless personal area networks (WPAN). Questa evoluzione tecnologica, comunicazione wireless, consente ai sensori di poter abbandonare i cavi, ciò rende possibile la realizzazione delle reti di sensori senza fili, Wireless Sensor Network (WSN), che possono avere applicazioni in moltissimi settori. Se da una parte l'assenza di cavi incrementa i possibili campi applicativi ottenendo soluzioni che prima erano irrealizzabili o poco pratiche, dall'altra apre altre problematiche che possono limitare l'uso di tali dispositivi, ad esempio problemi di alimentazione e consumo, problemi di interferenze, ecc.

### 2.3.3 WiFi - IEEE 802.11

Lo scopo di questo protocollo è fornire connettività wireless a dispositivi o stazioni che richiedono un'installazione rapida, siano essi portatili, o ancora dispositivi montati su veicoli

mobili all'interno di una local area. L'IEEE 802.11 definisce le funzioni e i servizi richiesti ai dispositivi compatibili con lo standard. Esso definisce: le procedure MAC e le tecniche di trasmissione dei segnali attraverso il livello fisico che potrà essere la radio frequenza (RF) o la radiazione infrarossa (IR). La comunicazione è di tipo orientata a pacchetto. Inoltre vengono descritte delle procedure per fornire un certo livello di riservatezza delle informazioni.

I dispositivi utilizzati nella tesi sono:

- Roving Networks WiFly RN-XV (Fig. 2.18) Il modulo RN-XV WiFly è un dispositivo WiFi con precaricato un firmware che semplifica l'integrazione e il tempo di sviluppo. Esso è basato sulla tecnologica Roving Networks RN-171 ed è equipaggiato con un processore a 32 bit, uno stack TCP/IP, un real-time clock, 8 Mbit di flash memory e 128 KB di RAM. Richiede solo una configurazione iniziale per accedere alla rete e iniziare a trasmettere/ricevere i dati sulla UART. Il modulo richiede poca energia e varia il suo raggio di potenza tra i livelli di 0 e +12 dBm. Può essere configurato in WiFi (come un router) oppure attraverso l'UART usando comandi ASCII. Sono supportati diversi algoritmi di autenticazione, incluso WEP, WPAPSK (TKIP), WPA2.
  
- Wireless Shield (Fig. 2.19) La Wireless Shield abilita la comunicazione wireless attraverso un modulo montato sopra di essa. Come la Ethernet Shield, dispone di una connessione on-board per una SD-card, utile se si vogliono salvare dati fissi da inviare sulla rete. La shield ha uno switch con il quale si seleziona il tipo di comunicazione: USB o Micro. La prima permette di programmare direttamente dal computer il dispositivo senza dover rimuovere il modulo dalla shield, Micro invece permette la comunicazione diretta del dispositivi WiFi con Arduino (disabilitando la comunicazione seriale USB). Questo switch è utile per la programmazione on-board e deve essere settata opportunamente per garantire il corretto funzionamento della scheda.



Figura 2.18: Modulo WiFi RN-XV della Roving Networks

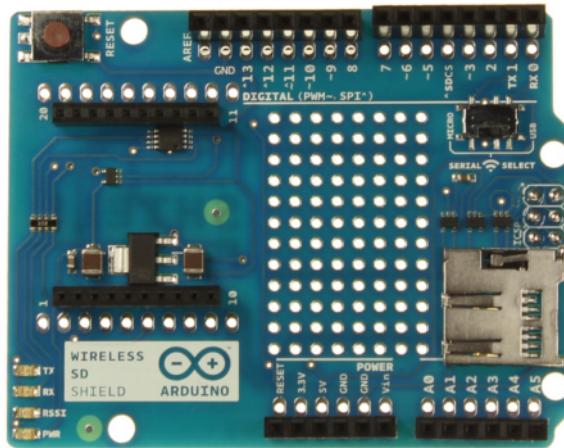


Figura 2.19: Wireless Shield di Arduino

Con questa configurazione è possibile creare un HTTP client dal quale inviare i diversi dati al server. Il modulo WiFi è già pronto all'uso (non ha bisogno di configurazioni particolari, salvo per necessità diverse). Per sfruttarne le sue potenzialità sono state utilizzate le librerie WiflyHQ. Sono librerie pronte all'uso per Arduino e fornisco diversi esempi per creare una comunicazione sia HTTP che TCP/UDP.

La libreria fornisce diverse funzioni per configurare e gestire il modulo WiFly e mandare e ricevere dati su una connessione TCP/UDP.

Codice d'esempio per configurare e usare l'interfaccia seriale messa a disposizione dal dispositivo:

```
Serial.begin(9600);  
wifly.begin(&Serial, NULL);
```

Configurazione e uso di una seriale software per interfacciarsi al dispositivo:

```
#include <SoftwareSerial.h>  
SoftwareSerial wifiSerial(8,9);  
  
wifiSerial.begin(9600);  
wifly.begin(&wifiSerial);
```

Collegarsi ad una rete WiFi;

```
wifly.setSSID("mySSID");  
wifly.setPassphrase("myWPApassword");  
wifly.enableDHCP();  
wifly.join();
```

Creare una rete WiFi Ad Hoc con 10 canali:

```
wifly.createAdhocNetwork("myssid", 10);
```

Mandare pacchetti UDP:

```
wifly.setIpProtocol(WIFLY_PROTOCOL_UDP);  
wifly.sendto("Hello, world", "192.168.1.100", 2042);
```

Aprire una connessione TCP, mandare alcuni dati, e chiudere la connessione:

```
wifly.setIpProtocol(WIFLY_PROTOCOL_TCP);  
wifly.open("192.168.1.100", 8042);  
wifly.println("Hello, world!");  
wifly.close();
```

Ricevere dati UDP e TCP (assumendo di avere una seriale software):

```
if (wifly.available() > 0) {  
    Serial.write(wifly.read());  
}
```

Facile manipolazione di diverse opzioni ricevute con il metodo multiMatch():

```
if (wifly.available() > 0) {  
    int match = wifly.multiMatch_P(100, 3, F("button"), F("slider="), F("switch="));  
    switch (match) {  
        case 0: /* button */  
            Serial.print(F("button: pressed"));  
            break;  
        case 1: /* slider */  
            int slider = wifly.parseInt();  
            Serial.print(F("slider: "));  
            Serial.println(slider);  
            break;  
        case 2: /* switch */  
            char ch = wifly.read();  
            Serial.print(F("switch: "));  
            Serial.println(ch);  
            break;  
        default: /* timeout */  
            break;  
    }  
}
```

Nell'ambito di questo progetto è stato utilizzato il seguente codice. Esso invia una stringa JSON al server e attende una risposta. Quest'ultima conterrà un valore che indica il timeout entro quale inviare la prossima lettura.

```
...  
//CREO LA STRINGA JSON CHE MI SERVIRA PER INVIARE I DATI AL SERVER  
Serial.println("CREO JSON");
```

```
sprintf(jsonMsgHead, "{\"timestamp\":%s,\"checksum\":%s,\"mac\":%s\"\0",
    timestamp, "5EB63BBBE01EED093CB22BB8F5ACDC3", "00:06:66:71:d2:68");
sprintf(jsonMsgBody, ",\"light\":%s,\"gas\":%s,\"temperature\":%s,\"humidity\":%s
}\"\0",
    lightBuffer, smokeBuffer, tempBuffer, humBuffer);
...
void InvioWIFIHttp(char* jsonStringHead,char* jsonStringBody, int lungh) {
    Serial.println("CREO RICHIESTA");

    wifly.print("POST /dsgm2m/api/m2m.php HTTP/1.0\r\n");
    wifly.print("Content-Length: ");
    wifly.print(lungh);
    wifly.print("\r\n\r\n");
    wifly.print(jsonStringHead);
    wifly.print(jsonStringBody);

    Serial.println("RISPOSTA DEL SERVER");
    //ATTENDO LA RISPOSTA DAL SERVER
    while(wifly.available()==0) {}

    if(wifly.available() > 0) {
        char buf[200] = "buffer";
        int exit = 0;

        while(exit<2) {
            wifly.gets(buf, sizeof(buf));
            Serial.println(buf);
            if(buf[0]==0) {
                exit++;
            }
            if(buf[0]=='{') {
                delay(50);
                long timeSend = parsingJSONString(buf, sizeof(buf));
            }
        }
    }
}
```

```
timeSend *= 1000;

//ATTENDO QUESTO VALORE PER POI RESETTARE LA SCHEDA
delay(timeSend);

}

}

}

}
```

La risposta del server è utile perchè contiene il valore di quanto si deve attendere per il successivo invio dei dati.

Limitazioni con WiFly RN-XV rev 2.32 firmware:

1. Non si può determinare l'indirizzo IP del client TCP che si è connesso.
2. Cambiare la porta locale durante l'esecuzione non influenza nulla finchè non si salva e si riavvia il dispositivo.
3. Potrebbe non funzionare la chiusura di una connessione TCP. Il client deve rimanere connesso e inviare ulteriori dati.
4. È supportata solo una connessione TCP alla volta.
5. Cambiare il timeout di svuotamento (set comm time x) non porta cambiamenti fino al riavvio del dispositivo.

#### 2.3.4 ZigBee - IEEE 802.15.4

ZigBee è un protocollo che opera nelle frequenze radio assegnate per scopi industriali, scientifici e medici (ISM): 868 MHz in Europa, 915 MHz negli Stati Uniti e 2,4 GHz nella maggior parte del resto del mondo. Questa tecnologia ha lo scopo di essere più semplice e più economica di altre WPAN come, ad esempio, Bluetooth. Il nodo ZigBee del tipo più complesso richiede solo il 10% del codice di programmazione necessario per un tipico nodo Bluetooth

o WiFi, mentre il più semplice dovrebbe richiederne intorno al 2%. Tuttavia, attualmente le dimensioni reali sono più alte e si aggirano intorno al 50% del codice necessario per Bluetooth. I produttori di chip ZigBee prevedono dispositivi da 128 kB. Nel 2005 il costo stimato per il ricetrasmettitore di un nodo ZigBee era di circa \$1.10 per il produttore, contando grossi volumi. La maggior parte dei dispositivi ZigBee richiedono però anche un microcontrollore, che fa alzare il costo totale. Quando fu lanciato (1998), per il Bluetooth si prevedeva un costo di \$4-\$6 per grandi volumi, mentre il prezzo attuale per la fascia consumer è oggi sotto i \$3. ZigBee Alliance [41] ha cominciato a lavorare sulla versione 1.1. che mira ad avvantaggiarsi dei miglioramenti della specifica 802.15.4b, fornendo maggiore flessibilità nella scelta dei metodi di autenticazione e crittografia. I protocolli ZigBee sono progettati per l'uso in applicazioni embedded che richiedano un basso transfer rate e bassi consumi. L'obiettivo attuale di ZigBee è di definire una Wireless mesh network non ad-hoc, economica e autogestita che possa essere utilizzata per scopi quali il controllo industriale, le reti di sensori, domotica, le telecomunicazioni, ecc. La rete risultante avrà un consumo energetico talmente basso da poter funzionare per uno o due anni sfruttando la batteria incorporata nei singoli nodi. Il modulo XBee (Fig.2.20), che sfrutta il protocollo sopra ci-



Figura 2.20: Modulo Xbee

tato, fornisce un modo alternativo per trasferire i dati senza l'uso di fili. Esso utilizza un

ricetrasmettitore wireless a 2,4 GHz per comunicare con un altro modulo XBee. Inoltre, i moduli sono in grado di comunicare con più di un XBee contemporaneamente. In questo modo è possibile creare una rete di moduli, fintanto che sono in serie.

I dettagli tecnici e le caratteristiche del modulo XBee sono:

- 802.15.4 è il protocollo creato IEEE.
- Data rate di 250Kbps.
- Può essere usato all'interno e all'esterno.
- Il range di visibilità è da 30m a 100m per i moduli XBee standard e 100m a 2Km per i moduli XBee Pro (a seconda di dove viene utilizzato e la visibilità da un XBee al XBee successivo).

La XBee standard ha una potenza di 1 mW di trasmissione e la Pro XBee ha una potenza di trasmissione 60mW. L'interfaccia di comunicazione è di tipo seriale. Nessuna configurazione è richiesta “out of the box”.

La velocità di trasmissione predefinita è 9600bps. Però è possibile modificare la configurazione attraverso un software fornito direttamente dal produttore.

Il modulo XBee è un accessorio di notevole importanza se si sta cercando di comunicare o controllare i dispositivi a basso costo, in ambienti limitati e soprattutto senza l'uso di fili. Un piccolo problema quando si utilizza la comunicazione wireless può essere quello delle interferenze. Quando in una trasmissione, un segnale (i dati) viene inviato in un contesto reale, a volte i dati possono venire perturbati da fattori noti e sconosciuti. Questi fattori possono essere considerati come rumore o interferenze che possono comprendere un segnale a radiofrequenza, conduzione del segnale o una combinazione di altri fattori come il tempo e che quindi bloccano la comunicazione. Nel nostro caso, per risolvere il problema, il modulo XBee utilizza un particolare protocollo senza fili per comunicare con un altro modulo XBee noto come DSSS o Direct Sequence Spread Spectrum. In termini

semplici, la tecnologia utilizzata in XBee utilizza un modello di trasmissione di frequenze multiple (frequency hopping) per evitare interferenze con l'XBee ricevente. Poiché entrambi i dispositivi comprendono il modello di frequenza, allora si presume che i dati possano essere ricevuti con successo con un'interferenza minima.

Con Arduino l'implementazione della comunicazione con questi moduli è molto semplice. Avendo a disposizione un Arduino board, una Wireless Shield (configurata in microcontrollore) e un modulo XBee è possibile iniziare una comunicazione senza configurazione alcuna. Di seguito un esempio di programma per il trasmettitore. XBee non fa altro che sfruttare la serial di Arduino per comunicare con gli altri dispositivi nella stanza (a tale proposito è necessario impostare la wireless shield in modalità micro).

```
void setup() {  
    //settare il baud rate della comunicazione Arduino a 9600  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.print("L");           //manda il carattere L  
    delay(2000);                //attende 2 secondi  
    Serial.print("H");           // manda il carattere H  
    delay(2000);                //attende 2 secondi  
    Serial.print ("X");          // manda il carattere X  
    delay(2000);                //attende 2 secondi  
}
```

La programmazione è molto semplice, come si può notare. Il modulo ricevitore avrà bisogno di un modo per sapere che sta ricevendo dati corretti. Utilizza anche in questo caso la seriale di Arduino.

```
void setup() {  
    //settare il baud rate della comunicazione Arduino a 9600  
    Serial.begin(9600);  
}
```

```
void loop() {  
    //Se ci sono dati sulla seriale li vado ad analizzare  
    if (Serial.available()) {  
        delay(10); //DEVO ASPETTARE UN PO PER RIEMPIRE IL BUFFER  
        inputSize = Serial.available();  
  
        //Leggo tutti i dati ricevuti e li vado a salvare in un buffer  
        for (i = 0; i < inputSize; i++) {  
            buffer[i] = Serial.read();  
            buffer[i+1] = '\0';  
        }  
  
        Serial.print("SIZE: ");  
        Serial.print(inputSize);  
        Serial.print(" - BUFFER: ");  
        Serial.println(buffer);  
    }  
}
```

Così facendo si è instaurata una rete Zigbee sfruttando la comunicazione seriale di Arduino.

Nel nostro caso la trasmissione dei dati raccolti da Arduino verso il server viene fatta con un modulo Xbee (Sender) che invia i dati ad un altro Arduino su cui è montato un altro Xbee (Receiver) che, attraverso un'interfaccia ethernet, invia i dati al server.

## 2.4 Server

I servizi lato server sono stati sviluppati nel linguaggio PHP, acronimo ricorsivo di PHP: Hypertext Preprocessor. Esso è un linguaggio di scripting interpretato, con licenza d'uso Open Source.

Su una macchina presente all'interno dei laboratori dell'Università di Parma è stata installato un server Apache virtuale raggiungibile all'indirizzo *monfortino.ce.unipr.it/dsgm2m*.

Il server rimane in attesa di richieste provenienti dalle schede Arduino che inviano i dati dei sensori alla pagina `/api/m2m.php`. Una volta ricevute le informazioni vengono salvate in un database sviluppato in MySQL. In figura 2.21 è presente lo schema del database utilizzato in questo progetto.

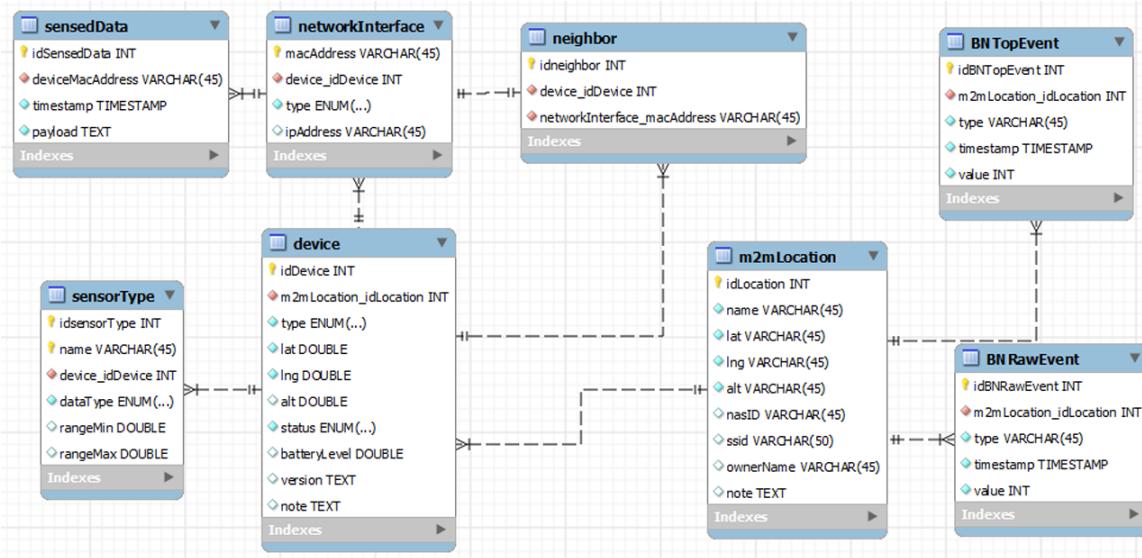


Figura 2.21: Modello ER del database utilizzato

- **sensedData**, tabella più importante perchè contiene tutti i log mandati dalle schede Arduino;
  - idSensedData INT,
  - deviceMacAddress VARCHAR(45),
  - timestamp TIMESTAMP,
  - payload TEXT
- **networkInterface**, tabella che contiene il tipo d'interfaccia di rete che monta ogni dispositivo che comunica con il server;
  - macAddress VARCHAR(45),

- deviceidDevice INT,
  - type ENUM(...),
  - ipAddress VARCHAR(45)
- **neighbor**, tabella che contiene le schede vicine ad altre;
    - idneighbor INT,
    - deviceidDevice INT,
    - networkInterfacemacAddress VARCHAR(45)
- **device**, tabella che contiene tutti i dispositivi;
    - idDevice INT,
    - m2mLocationidLocation INT,
    - type ENUM(...),
    - lat DOUBLE,
    - lng DOUBLE,
    - status ENUM(...),
    - batteryLevel DOUBLE,
    - version TEXT,
    - note TEXT
- **deviceConfiguration**, tabella che contiene le configurazioni di ogni scheda (tipo ogni quanto deve inviare i dati al server)
    - idConfiguration INT,
    - deviceidDevice INT,
    - targetUrl VARCHAR(150),

- updatePeriod DOUBLE
- **sensorType**, tabella che contiene tutti i sensori;
  - idSensorType INT,
  - name VARCHAR(45),
  - deviceidDevice INT,
  - dataType ENUM(...),
  - rangeMin DOUBLE,
  - rangeMax DOUBLE
- **m2mLocation**, tabella contenente le località di dove sono installati i dispositivi
  - idLocation INT,
  - name VARCHAR(45),
  - lat VARCHAR(45),
  - lng VARCHAR(45),
  - alt VARCHAR(45),
  - nasID VARCHAR(45),
  - ssid VARCHAR(45),
  - ownerName VARCHAR(45),
  - note TEXT
- **BNRawEvent**, tabella contenente gli eventi di basso livello generati per la rete di Bayes
  - idBNRawEvent INT
  - m2mLocation-idLocation INT

- type VARCHAR(100)
  - timestamp timestamp CURRENT-TIMESTAMP
  - value INT
- **BNTopEvent**, tabella contenente gli eventi di alto livello generati dalla rete di Bayes
    - idBNTopEvent INT
    - m2mLocation-idLocation INT
    - type VARCHAR(100)
    - timestamp timestamp CURRENT-TIMESTAMP
    - value TINYINT

L’obiettivo del server è quello di raccogliere i dati dai sensori, dare accesso ed elaborare i dati grezzi e infine mettere a disposizione i dati elaborati sotto forma di grafici o tabelle. Di seguito è descritta l’API principale utilizzata per interfacciarsi con il DB per l’inserimento ed il recupero di dati.

- *addNewLog*, permette di inserire all’interno del database un lettura proveniente da una scheda Arduino;
- *getLocationInfoWithId*, restituisce le informazioni sulla localizzazione in base al suo ID;
- *getDeviceInfoWithId*, restituisce le informazioni del dispositivo in base al suo ID;
- *getLastHourSensedData4DeviceId*, restituisce i dati di un dispositivi letti nell’ultima ora;
- *getLastSensedData4DeviceId*, restituisce gli ultimi dati letti di un dispositivo;
- *getAllSensedData4DeviceId*, restituisce tutti i dati letti di un determinato dispositivo;

- `getAllM2MLocationsAndDevices`, restituisce tutte le locazioni e i dispositivi presenti nel DB;

#### 2.4.1 Web Client

È stato sviluppato per questo progetto un web client dal quale è possibile monitorare i dati ricevuti dalle schede. Si può accedere ad esso dall'indirizzo `http://monfortino.ce.unip.it:8080/dsgm2m`. Appena entrati viene mostrata la pagina di login (Fig. 2.22) dalla quale è possibile immettere le proprie credenziali per accedere al sito WEB.



Figura 2.22: Pagina di login

Una volta entrati viene visualizzata una mappa dalla quale è possibile selezionare il device di cui si vogliono monitorare i dati provenienti dai sensori (nel nostro caso l'unico device è presente nel laboratorio DSG, vedi Fig. 2.23).

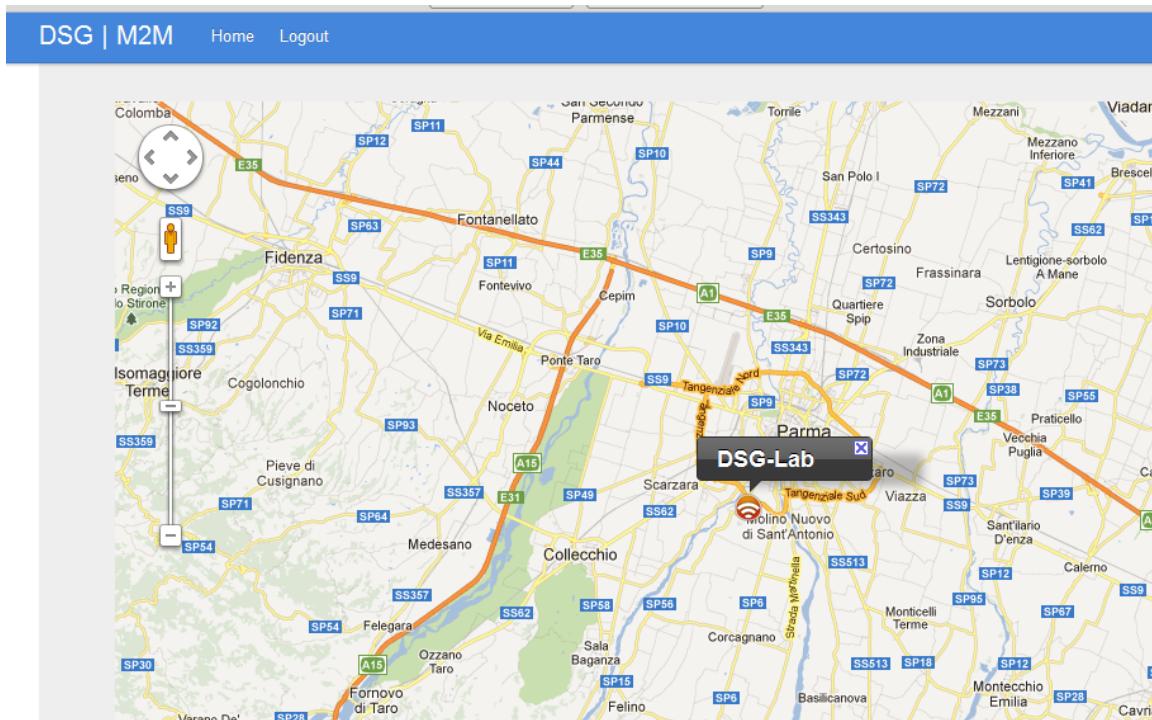


Figura 2.23: Pagina principale del sito Web

Cliccando sul device si passa ad un'altra pagina che ci mostra i grafici delle ultime letture dei sensori. In figura 2.24 possiamo notare alcuni grafici dei sensori dislocati nel laboratorio.

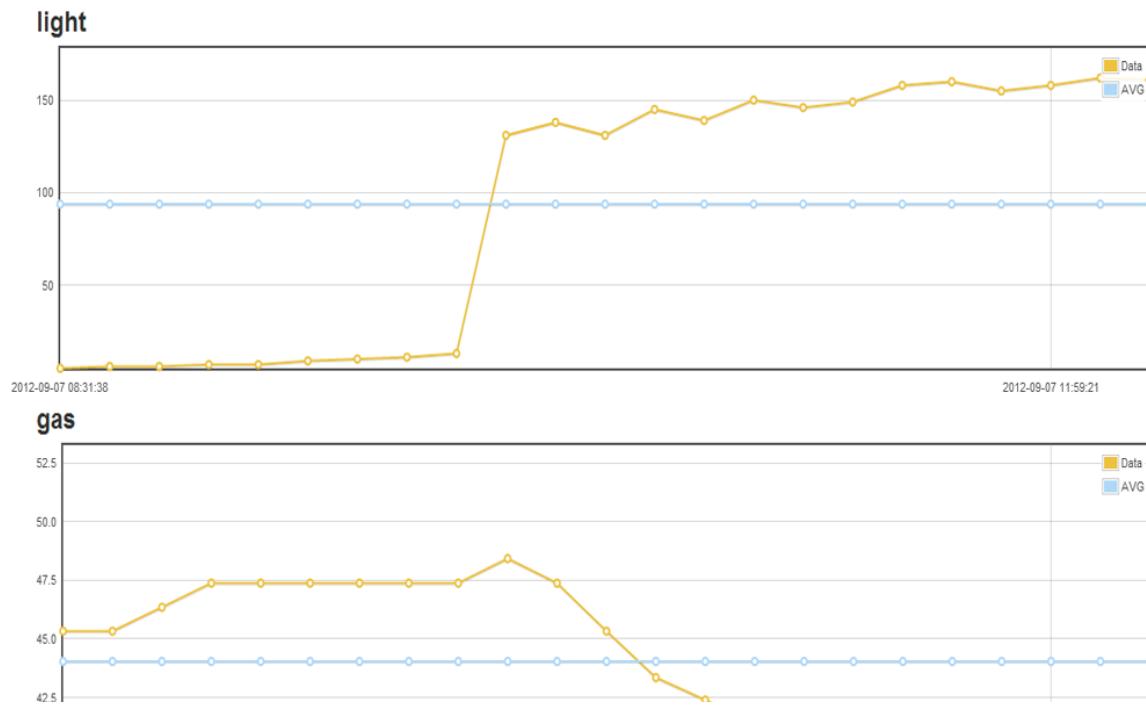


Figura 2.24: Alcuni grafici che vengono visualizzati dal sito WEB

Il server dispone inoltre di un modulo Networked Autonomic Machine (NAM) per la gestione delle reti Bayesiane che è descritto nel paragrafo 2.6.1.

## 2.5 Applicazione mobile

L'applicazione sviluppata per questo progetto consente, attraverso un'activity (cioè una rappresentazione di una schermata), di monitorare lo stato dell'ambiente visualizzando i dati ricevuti dalle schede Arduino. In figura è mostrata la schermata principale. All'avvio viene richiesto il nome utente con il quale accedere al server. Questo sarà utile per ricevere le notifiche dal server quando dovrà segnalare degli eventi di alto livello. Android offre la possibilità di sfruttare il servizio GCM con il quale è possibile inviare notifiche istantanee ai dispositivi. GCM è l'acronimo di “Google Cloud Messaging” (Google I/O, 28 Giugno 2012), da cui si comprende che il servizio opera in ambiente cloud di Google offrendo, tra i diversi vantaggi, un elevato livello di affidabilità. Lo scopo principale di GCM è quello sia di inviare un nuovo contenuto al software in esecuzione sul device Android, ma anche di segnalare a quest'ultimo la presenza di nuovi dati sul server per un successivo recupero e visualizzazione della nuova informazione sul dispositivo mobile.

Un'altra caratteristica interessante di GCM è quella che l'applicazione Android destinataria di un messaggio non deve essere in esecuzione per riceverlo: il sistema operativo, ricevuta la segnalazione, si farà carico di instradarla all'applicazione specifica. L'architettura di GCM prevede l'interazione di tre moduli software:

- Il server è l'applicazione, solitamente di terze parti, che si occupa di inviare i messaggi ai dispositivi Android che si sono registrati per la ricezione dei messaggi.
- Google GCM rappresenta il software che si pone come intermediario tra i servers di terze parti e le applicazioni android, svolgendo principalmente compiti di instradamento delle informazioni. Per poter realizzare l'instradamento, il server GCM deve

ricevere ed immagazzinare le opportune informazioni sia sulla presenza dei server che sulla presenza dei client accreditati ai diversi servizi push.

- L'applicazione Android è la destinataria dei messaggi di tipo push e offrirà, tra le altre funzionalità, anche quella di recupero e gestione dell'informazione presente sul server.

Il protocollo di comunicazione tra le parti prevede una prima fase di accreditamento sia del server che dei client interessati sul server GCM. Successivamente può instaurarsi la comunicazione vera e propria attraverso l'invio di un messaggio push dal server al GCM. Quest'ultimo dopo aver ricavato le informazioni dei device di destinazione, procede a consegnare la segnalazione ai device interessati. Da quanto visto emerge che il server non ha vincoli implementativi per quanto riguarda il linguaggio e le tecnologie utilizzate per lo sviluppo in quanto deve solo occuparsi di inviare messaggi con il protocollo atteso da GCM.

L'applicazione sviluppata per questo progetto consiste in poche activity dove è possibile monitorare lo stato dell'ambiente ed eventualmente modificare lo stato degli attuatori.

Una volta installata l'applicazione sul dispositivo viene richiesto un nome utente che serve per registrarsi al servizio GCM di Google presente anche sul server (vedi Fig. 2.25).

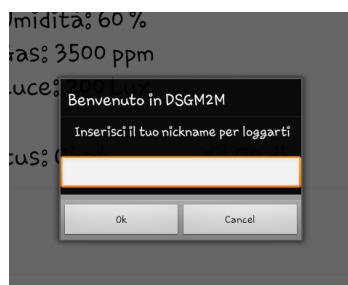


Figura 2.25: Form di login alla prima apertura dell'applicazione

Una volta immesso il nome utente viene mostrata un'activity(Fig. 2.26) dalla quale è possibile vedere il nome e la porta del server a cui ci si collega, le informazioni riguardanti i sensori di temperatura, umidità, gas e luce, lo stato della stanza proveniente dal modulo NAM (spiegato nei paragrafi successivi) e la lettura del sensore del suono. Inoltre è disponibile un pulsante per la modifica dello stato degli attuatori.

nibile una casella di testo dove vengono mostrati i TAG del personale che accede al locale (movimenti rilevati con il sensore PIR ma anche attraverso TAG RFID che USB Host). Il tasto di Refresh serve per aggiornare le letture sopra descritte.

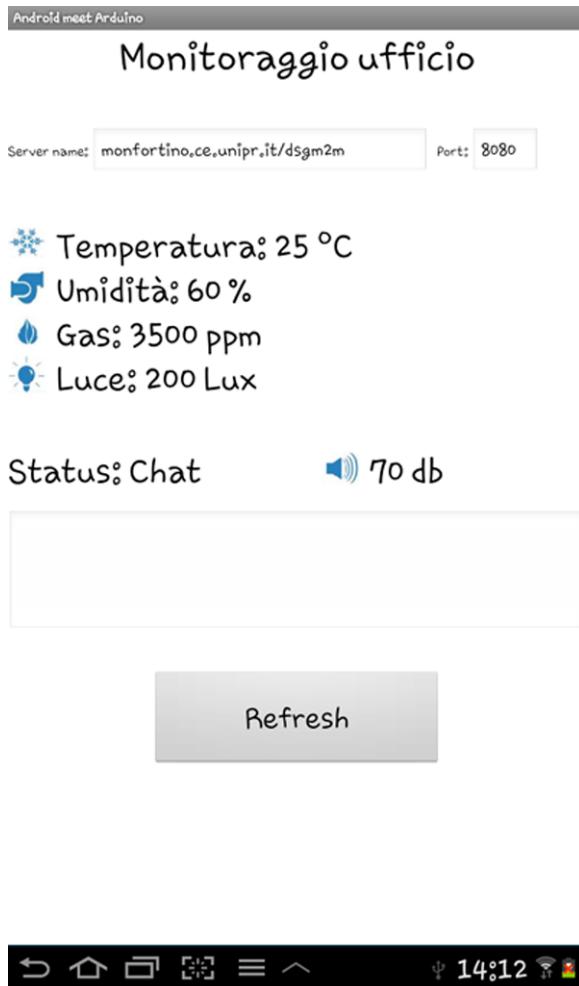


Figura 2.26: Activity principale dell'applicazione

Premendo il tasto menù è possibile accedere a due sezioni:

- Impostazioni(Fig. 2.27), dalla quale è possibile modificare alcune impostazioni dell'applicazione;
- Attuatori(Fig. 2.28), dove è possibile modificare lo stato degli attuatori presenti

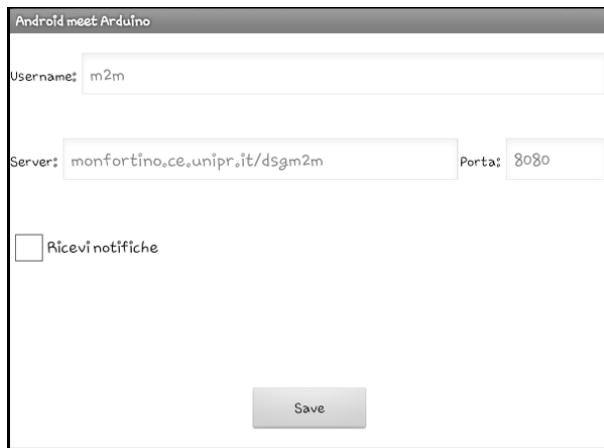


Figura 2.27: Activity dove si possono modificare le impostazioni dell'applicazione



Figura 2.28: Activity dove si può modificare lo stato degli attuatori

nell'ambiente monitorato. Nel nostro caso sono solamente le luci, le ventole e le finestre.

Il server, ogni volta che rileva eventi di alto livello (vedi sezione 2.6.1), invia ai dispositivi Android registrati una notifica contenente un messaggio relativo all'evento stesso (Fig.2.29).

In base all'evento notificato, l'applicazione mette a disposizione diverse alternative:

- Chiamata di emergenza nel caso venga segnalato un incendio, un'intrusione, ecc.
- Azionare gli attuatori per cercare di risolvere la situazione segnalata, come fuga di gas, cattive condizioni ambientali, ecc.
- Vedere chi è presente all'interno della stanza nel caso ci sia attività lavorativa o di riunione

## 2.6 Data Mining

Il modulo di Data Mining a lato server si occupa di analizzare i dati provenienti dai sensori e, in base ad essi, predire qual'è lo stato del locale in cui essi sono inseriti. È possibile,

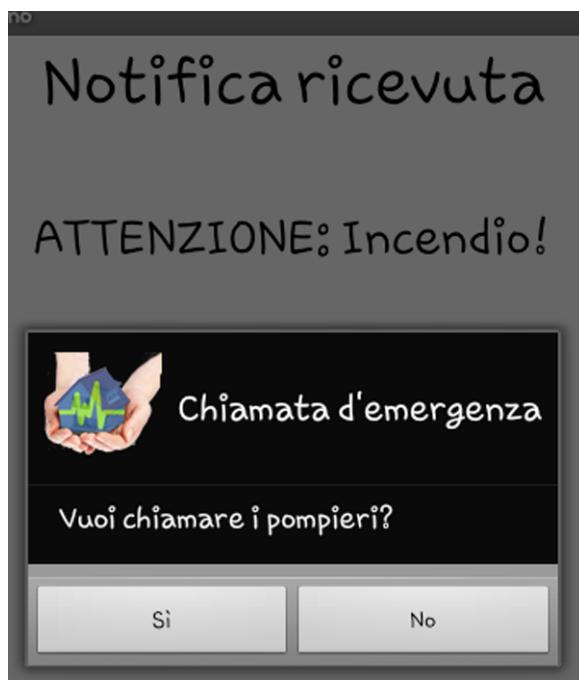


Figura 2.29: Notifica d'incendio ricevuta al terminale

quindi, azionare autonomamente degli attuatori solo in base alla lettura dei sensori. Le reti di Bayes sono un ottima soluzione per questo scopo perchè permettono di valutare una rete in base a stati di basso livello (nel nostro caso i sensori) per poi produrne risultati di alto livello. Al fine di poter analizzare i dati, valutare l'accuratezza delle classificazioni o delle associazioni e verificarne quindi le capacità predittive, è stato utilizzato lo strumento di data mining WEKA, per il quale si fornisce una breve illustrazione del suo utilizzo rimandando al sito web del progetto WEKA [42] ed al libro di I. H. Witten e E. Frank [43] per maggiori dettagli.

Lo strumento WEKA (Waikato Environment for Knowledge Analysis) è una collezione di algoritmi di apprendimento automatico che possono essere applicati sia direttamente ad un insieme di dati attraverso l'interfaccia con la quale l'utente interagisce con il sistema, oppure richiamandoli dal proprio codice di programmazione in linguaggio Java. WEKA, sviluppato dall'Università di Waikato in Nuova Zelanda, è giunto alla versione 3.4, nella

versione collegata al libro di I. H. Witten e E. Frank ed alla versione 3.5.8 nella versione di sviluppo; richiede che sul sistema sia installato l'ambiente di run time Java RE almeno nella versione 1.4 ed esegue quindi su diverse piattaforme operative quali Linux, Windows, Mac Os. È disponibile con licenza “GNU General Public License” ed è liberamente scaricabile da <http://www.cs.waikato.ac.nz/ml/WEKA/> da dove si può facilmente accedere inoltre a numerose pagine di supporto e documentazione. WEKA fa uso di una propria terminologia il cui significato, per i termini di uso più frequente, viene di seguito specificato:

- **classe** (o concetto): rappresenta ciò che stiamo cercando, cioè l'informazione che deve essere ricavata dal modello automatico; è un'informazione di input nel senso che ognuna delle istanze utilizzate per l'addestramento deve contenere il valore della classe di appartenenza, rispetto a tutti i valori della classe che rappresenta il concetto in esame;
- **istanze**: insieme di informazioni (attributi) che descrivono i casi in esame;
- **attributi**: serie di valori ognuno dei quali descrive un particolare aspetto delle varie istanze;
- **rappresentazione della conoscenza**: è il modo con il quale la conoscenza appresa in modo automatico viene rappresentata; tipicamente vengono rappresentate attraverso liste di decisioni, regole di associazione, regole basate su alberi di decisione, regole basate su istanze (vicinanza del caso in esame con altre istanze nel set di training), cluster di istanze vicine, cluster gerarchici;
- **algoritmi di apprendimento**: gli algoritmi che implementano lo schema di apprendimento, sono classificabili in varie tipologie a seconda dei metodi utilizzati:
  - tipicamente statistici,
  - divide-et-impera,
  - alberi di copertura,

- associativi,
  - lineari,
  - basati sulle istanze
  - basati sulla clusterizzazione;
- **training set:** è l’insieme di tutte le istanze che sono disponibili per l’algoritmo al fine dell’apprendimento;
  - **test set:** è l’insieme di istanze utilizzate dall’algoritmo per valutarne la sua credibilità.  
Il test set deve essere diverso dal training set;
  - **credibilità:** la credibilità del risultato viene valutata attraverso specifici metodi di controllo. Gli strumenti più diffusi sono: cross-validation, leave-one-out, bootstrap. È necessario valutare anche il costo di un’errata classificazione. Per questo si usano tabelle di contingenza, matrici di confusione, ROC analisi;
  - **error rate:** è la proporzione tra gli errori fatti sull’intero set delle istanze;

Il software WEKA può gestire i dati di input per l’apprendimento in diversi modi: collegandosi ad un database con standard SQL attraverso i driver java JDBC, importando i dati da file CSV, leggendoli da sorgenti XML o da URL, e altro ancora. Il modo nativo, che spesso è conveniente da usare nella fase iniziale di analisi dei dati, è il formato “Attribute-Relation File Format (ARFF)”; si tratta di un file di testo composto da un’intestazione che descrive il tipo di attributi utilizzati e da un elenco di istanze usate per l’addestramento, una per ogni riga, in ognuna delle quali si ripetono i valori degli attributi secondo l’ordine descritto nella testata. L’esempio qui sotto riportato illustra il formato dati ARFF.

#### Esempio di file ARFF

```
%QUESTA RIGA SERVE PER DICHIARARE IL NOME DELLA RETE CHE SI STA CREANDO  
@relation RandomNet
```

```
%ORA SI DEVONO INSERIRE TUTTI GLI ATTRIBUTI, SIA DI ALTO CHE BASSO LIVELLO,  
DELLA RETE.
```

```
%PER OGNI ATTRIBUTO BISOGNA ANCHE DEFINIRE LO STATO IN CUI POTREBBE TROVARSI  
@attribute INTRUSION {True,False}  
@attribute CHAT {True,False}  
@attribute FIRE {True,False}  
@attribute WORK {True,False}  
@attribute GAS_LEAK {True,False}  
@attribute BAD_CONDITIONS {True,False}  
@attribute S:SOUND {LSound,MSound,HSound}  
@attribute T:DAY_TIME {Morning,Midday,Afternoon,Evening,Night}  
@attribute P:ID_PEOPLE {Presence,Not_Presence}  
@attribute P:PIR {Catch,NotCatch}  
@attribute S:HUMIDITY {LHumidity,MHumidity,HHumidity}  
@attribute T:ISHOLIDAY {True,False}  
@attribute S:TEMPERATURE {LTemperature,MTemperature,HTemperature}  
@attribute S:SMOKE {LSmoke,MSmoke,HSmoke}  
@attribute S:BRIGHTNESS {LBrightness,MBrightness,HBrightness}
```

```
%DI SEGUITO VIENE INSERITA UNA LISTA DI POSSIBILI "SOLUZIONI" CON LE QUALI E'  
POSSIBILE ISTRUIRE LA RETE
```

```
@data  
True,False,False,False,False,LSound,Midday,  
Not_Presence,Catch,LHumidity,True,LTemperature,LSmoke,LBrightness  
True,False,False,False,False,LSound,Night,  
Not_Presence,Catch,LHumidity,True,MTemperature,LSmoke,LBrightness  
False,True,False,False,False,MSound,Morning,  
Presence,Catch,MHumidity,False,MTemperature,LSmoke,LBrightness  
False,True,False,False,False,MSound,Afternoon,  
Presence,Catch,MHumidity,False,MTemperature,LSmoke,LBrightness
```

Una volta preparati i dati ci sono diversi modi per interagire con il sistema. Per il nostro scopo di analisi è conveniente utilizzare l'interfaccia di esplorazione (explorer). Le restanti

funzionalità sono perlopiù adatte ad un utilizzo diverso degli stessi algoritmi presenti in explorer, come l’interfaccia a riga di comando (CLI), l’interfaccia di comparazione delle performance di differenti schemi di apprendimento (experimenter) e l’interfaccia di combinazione dei flussi di apprendimento che consente di connettere in sequenza le varie fasi (input dei dati, preprocessamento, classificazione, valutazione) e salvare il flusso operativo (knowledge flow gui); l’immagine riportata di seguito in figura 2.30 rappresenta l’interfaccia “explorer” di WEKA. Esso dispone di strumenti di pre-processamento dei dati (filtri), di classificazione, di regressione, di clusterizzazione, di analisi delle regole associative, di selezione degli attributi rilevanti e di visualizzazione ai fini di sommarizzazione dei dati. Inoltre WEKA mette a disposizione anche strumenti per la costruzione di propri schemi di apprendimento.

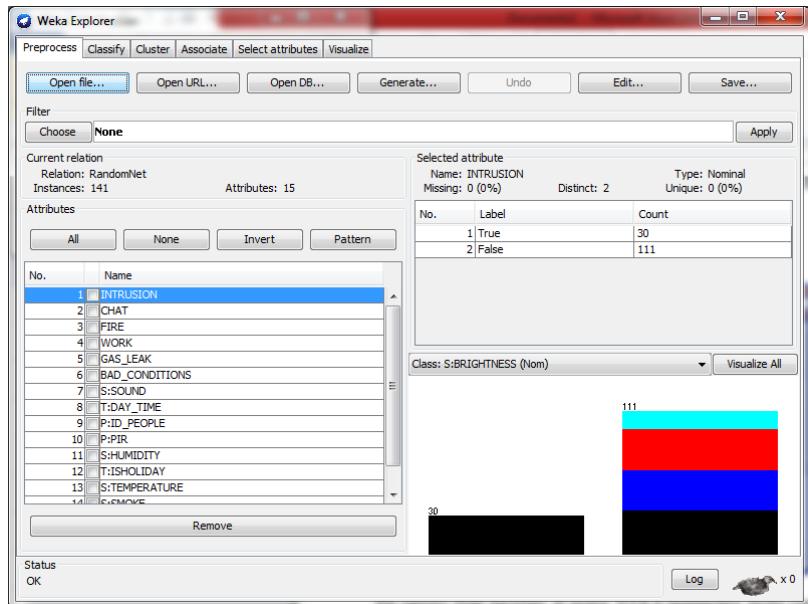


Figura 2.30: schermata interfaccia di WEKA explorer

Per ognuno degli algoritmi di analisi WEKA, implementa anche controlli di credibilità e strumenti di comparazione finalizzati alla verifica delle capacità di apprendimento.

### 2.6.1 Reti bayesiane

In questo progetto per definire il modello di conoscenza, abbiamo utilizzato le reti Bayesiane. Esse sono un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto. Per esempio una rete Bayesiana potrebbe rappresentare la relazione probabilistica esistente tra i sintomi le malattie. Dati i sintomi (effetti), la rete può essere usata per calcolare la probabilità della presenza di diverse malattie (cause) utilizzando la formula di Bayes 2.31.

$$p(B_j|A) = \frac{p(B_j \cap A)}{p(A)} = \frac{p(A|B_j) p(B_j)}{\sum_1^n p(A|B_i) p(B_i)}$$

Figura 2.31: Formula di Bayes

Nel nostro caso, date le letture dei sensori, si determina con una certa probabilità in che stato si trova la stanza.

Le reti Bayesiane sono definite tramite la specificazione di due componenti:

- **La componente qualitativa:** un grafo diretto aciclico (DAG), indicato con  $G=(V,A)$  detto struttura della rete:

- $V$  sono i nodi che sono in corrispondenza biunivoca con l'insieme  $X$  di variabili aleatorie;
- Gli archi  $A$  sono coppie ordinate di elementi di  $V$ .

Ogni arco denotato con  $X_i -> X_j$  rappresenta la dipendenza condizionata esistente tra i due nodi; i genitori di un nodo  $X_i$  sono denotati da  $\text{Pa}(X_i)$ , i figli con  $\text{Ch}(X_i)$

- **La componente quantitativa:** un insieme di distribuzioni locali di probabilità, ciascuna associata ad una variabile e condizionata ad ogni configurazione dei suoi

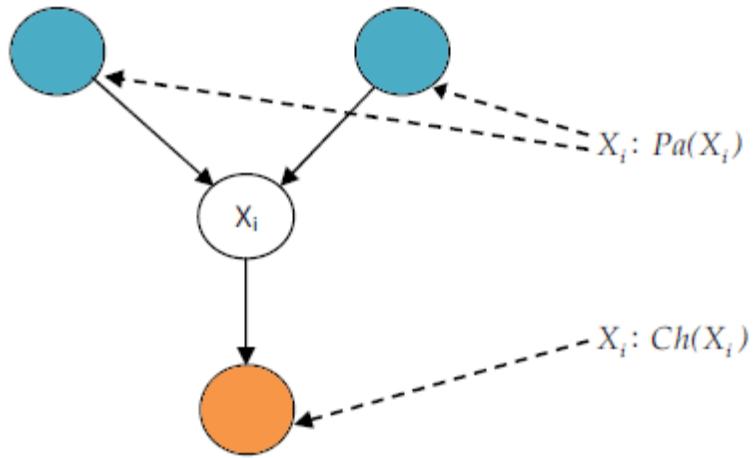


Figura 2.32: Esempio di rete bayesiana

genitori. L’insieme delle distribuzioni locali di probabilità specificano la distribuzione congiunta dell’insieme di variabili. Queste distribuzioni di probabilità sono dette anche parametri delle reti bayesiane.

Esse sono contraddistinte da alcune caratteristiche:

- Esprimono legami non deterministici tra le variabili aleatorie;
- Le conoscenze preliminari sul sistema permettono, almeno parzialmente, di determinare la struttura della rete;
- La distribuzione di ogni variabile è influenzata solamente dalle distribuzioni dei suoi diretti vicini all’interno della struttura. Quindi un nodo ha una tabella di probabilità condizionata che quantifica gli effetti che i genitori hanno sul nodo. Un nodo che non ha genitori contiene una tabella di probabilità marginale;
- Se il nodo è discreto, contiene una distribuzione di probabilità sugli stati della variabile che rappresenta;
- Il grafo non ha cicli diretti.

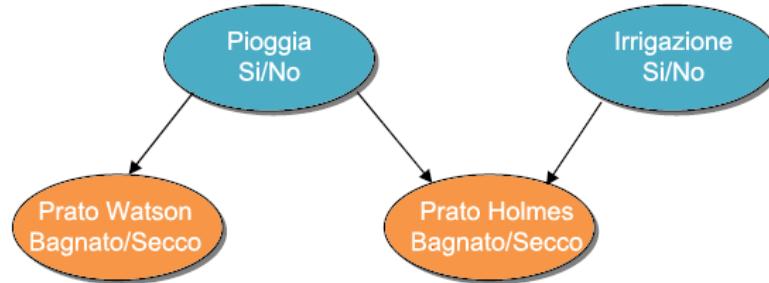
L'utilizzo delle reti Bayesiane ha dei vantaggi: la rappresentazione grafica e la struttura tra variabili aleatorie risulta intuitiva e di facile comprensione; sono utilizzabili anche per insiemi di dati incompleti; facilitano le combinazioni del dominio di conoscenza dei dati, permettendo la possibilità di specificare dei giudizi soggettivi di esperti sul modello.

La in/dipendenza è intesa come uno strumento che identifica le strutture di relazione tra le variabili aleatorie in esame e focalizza l'attenzione su ciò che è rilevante per studiare un fenomeno. Per descrivere interamente una situazione bisogna tener conto di altre due componenti: il condizionamento e la verosimiglianza. Il primo serve per evidenziare come il comportamento di una variabile può modificare l'andamento di un'altra. La seconda invece serve per identificare quali situazioni sono induttivamente più probabili di altre. L'esempio che segue ha lo scopo di chiarire questi concetti:

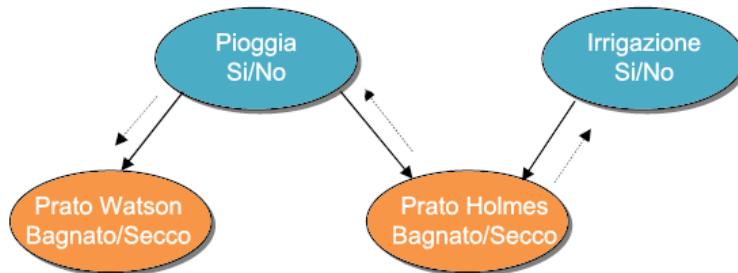
1. Holmes e Watson sono vicini di casa;
2. Una mattina Holmes andando al lavoro nota che il suo prato è bagnato. Si chiede se ha lasciato l'impianto d'irrigazione acceso o se ha piovuto;
3. Guardando il giardino di Watson nota che anche questo è bagnato;
4. Holmes pensa: “Poiché anche il giardino del vicino è bagnato, probabilmente questa notte ha piovuto”;
5. Pensa inoltre: “la pioggia spiega come mai il mio prato è bagnato, e quindi probabilmente l'impianto di irrigazione è spento”.

Quanto appena descritto è la trascrizione di una situazione che può sembrare banale quanto consueta. Si tratta ora di convertire ogni punto riportato in una struttura di relazioni di in/dipendenza.

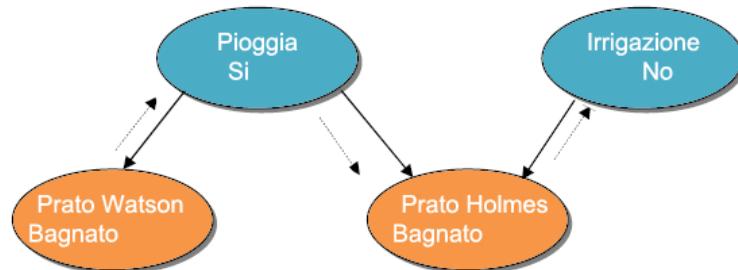
La situazione iniziale si può così rappresentare:



Holmes apprende che il suo prato è bagnato e questo genera una propagazione dell'informazione.



Il fatto che il prato di Holmes sia bagnato può derivare da due situazioni: ha piovuto oppure l'impianto d'irrigazione è rimasto acceso. Per individuare la causa Holmes osserva il prato del vicino e nota che è bagnato:



Sapendo che anche il prato di Watson è bagnato, è più verosimile che abbia piovuto. Il tipo di evidenza disponibile può influenzare la dipendenza o indipendenza tra due variabili

aleatorie: la presenza di pioggia e l'aver lasciato l'impianto d'irrigazione acceso sono solitamente quantità non correlate, ma l'evidenza che il prato sia bagnato, la rende dipendente perché l'assenza di pioggia rende più verosimile il fatto che l'impianto sia rimasto acceso.

Nel nostro progetto abbiamo utilizzato una rete che rispetta lo schema in figura 2.33.

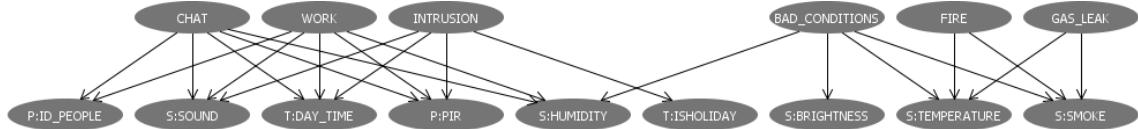


Figura 2.33: Rete di bayes usata per il progetto

Questo schema presenta due livelli di probabilità: i livelli di basso livello rappresentano le letture dei sensori spiegati nelle sezioni precedenti. Per ogni livello di questo tipo abbiamo diversi stati in cui potrebbe trovarsi. Questi stati sono stati discretizzati in modo che per ogni stato valga un certo range di valori (esempio LTemperature varrà se ci sono meno di 20°C):

- S:SOUND = LSound,MSound,HSound
- T:DAY-TIME = Morning,Midday,Afternoon,Evening,Night
- P:ID-PEOPLE = Presence,NotPresence
- P:PIR = Catch,NotCatch
- S:HUMIDITY = LHumidity,MHumidity,HHumidity
- T:ISHOLIDAY = True,False
- S:TEMPERATURE = LTemperature,MTemperature,HTemperature
- S:SMOKE = LSmoke,MSmoke,HSmoke
- S:BRIGHTNESS = LBrightness,MBrightness,HBrightness

Per gli eventi di alto livello invece abbiamo ipotizzato delle situazioni tipo in cui la stanza potrebbe trovarsi. Se la rete determina con una certa probabilità uno di questi eventi, allora il sistema prenderà provvedimenti adeguati (tipo se rileva un incendio potrebbe azionare l'impianto opportuno e inoltrare una chiamata ai pompieri).

- INTRUSION = True,False
- CHAT = True,False
- FIRE = True,False
- WORK = True,False
- GAS-LEAK = True,False
- BAD-CONDITIONS = True,False

Per creare questa rete abbiamo suddiviso il lavoro in tre diverse fasi:

- **Creazione del dataset:** è stato creato un file contenente centinaia di righe contenenti situazioni possibili che si potrebbero creare all'interno dell'ambiente. Le diverse situazioni sono state ipotizzate in base a test effettuati sulle letture rilevate dai sensori nelle prime settimane di funzionamento del sistema. Partendo da questo dati è stato possibile scrivere un file .ARFF2.6 il quale sarà utile per strutturare la rete con WEKA.
- **Struttutazione della rete con WEKA:** attraverso il tool "Bayes net editor" è stata possibile disegnare una rete come mostrata in figura 2.33. Successivamente è stato utilizzato lo strumento "Explorer" con il quale è stato caricato nel preprocessore il file creato al punto precedente. Poi, sfruttando la struttura disegnata precedentemente, è stato classificato il tutto con un algoritmo Bayesiano in modo da poter ottenere i diversi pesi della rete mostrata in figura 2.33. Questo processo genera un file XML contenente sia la struttura della rete, sia i pesi di ogni singolo nodo.

- **Apprendimento della rete con un software ad-hoc:** una volta creato il file XML della rete, è stato possibile sfruttare un software ad-hoc scritto in Java il quale, analizzando la rete, è in grado di determinare lo stato più probabile della stanza in base alla lettura dei sensori.

Il software utilizza le librerie fornite da WEKA contenente diverse classi per gestire le reti di Bayes. Esso è stato creato come nodo per una rete NAM e infatti sfrutta anche classi della libreria NAM4J.

NAM4J [44] è un Middleware Java che permette lo sviluppo di macchine in rete in modo autonomo. Il framework NAM consente la caratterizzazione di reti i cui nodi sono forniti di moduli funzionali e di servizi. Rispetto ad altri strumenti di modellazione, NAM consente di specificare la portabilità di moduli funzionali e servizi tra i nodi. Il formalismo NAM può essere utilizzato per caratterizzare semanticamente e confrontare gli elementi di un sistema dinamico altamente distribuito.

La figura seguente illustra la struttura interna di un modulo NAM. Questi moduli funzionali utilizzano le risorse e forniscono servizi atomici, che possono essere aggregati per fornire servizi composti.

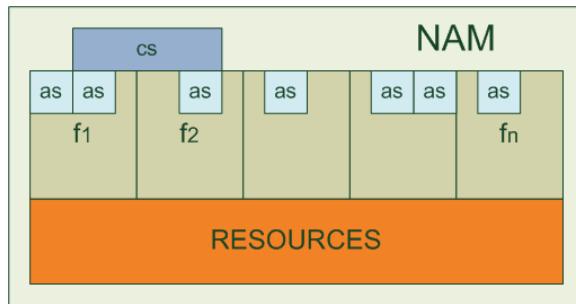


Figura 2.34: Struttura di un modulo NAM

La figura seguente illustra invece un modulo funzionale che interagisce con l'ambiente, che può includere sia moduli locali e che moduli Nam remoti, ma anche qualsiasi cosa al di fuori di quello della rete NAM (ad esempio operatori, utenti finali, il tempo, ecc.) L'interazione con l'ambiente si basa su due pilastri: la percezione, che significa context-

awareness, e la decisione, che è la capacità di inferire agli eventi un nuovo contesto, le chiamate di servizio o di uscita (ad alcuni utenti, o per una richiesta di servizio).

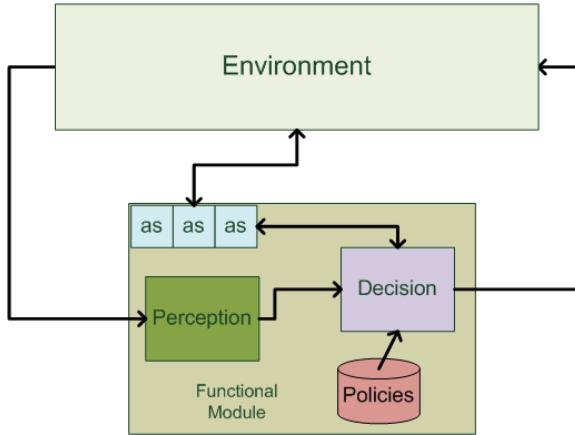


Figura 2.35: Esempio di un modulo NAM

Sulla base di questi concetti è stato sviluppato questo modulo che consente l'interazione del sistema con la rete Bayesiana sopra esposta.

Il seguente codice è stato sviluppato per interagire con la rete di Bayes. Attraverso la classe `ActivityMonitorFunctionalModule` andiamo ad impostare quale rete analizzare. Successivamente impostiamo la soglia di validità entro cui si può ritenere valido un evento di alto livello (se sotto quella soglia non viene segnalato nulla). Poi come esempio andiamo ad impostare i diversi stati degli attributi di basso livello per generare un esempio di prova.

```

ActivityMonitorFunctionalModule am =
    new ActivityMonitorFunctionalModule(null, "reteBN.xml", 0.7);

// VALORI DI PROVA PER VERIFICARE LO STATO D' INCENDIO
am.getSystemReadings().put("S:LIGHT", 0);
am.getSystemReadings().put("S:GAS", 0);
am.getSystemReadings().put("S:TEMPERATURE", 2);
am.getSystemReadings().put("S:HUMIDITY", 0);
am.getSystemReadings().put("S:SOUND", 1);
am.getSystemReadings().put("P:PIR", 0);
  
```

```
am.getSystemReadings().put("P:ID_PEOPLE", 1);
am.getSystemReadings().put("T:DAY_TIME", 0);
am.getSystemReadings().put("T:IS_HOLIDAY", 0);

//CLASSIFICO LA RETE IN BASE AI DATI IMMESSI
ClassificationResult result = am.performClassification(am.
    getSystemReadings());
double highestMargProb = result.getMargProb();
System.out.println("NAMActivityMonitor: MargProb: " + highestMargProb);
System.out.println("NAMActivityMonitor: Threshold = " + am.
    getProbThreshold());
if (highestMargProb > am.getProbThreshold()) {
    System.out.println("NAMActivityMonitor: the most probable
        situation is "
        + result.getName() + " with Pm = " + highestMargProb);
}
```

Nel progetto, il modulo legge dal server una stringa JSON contenente le ultime letture salvate. Con questi dati va ad impostare i pesi delle variabili di basso livello e calcola un evento di alto livello. Questo livello viene poi rimandato indietro al server e viene chiesto di mandare una notifica, ai dispositivi collegati con il servizio GCM, nel caso ci sia un evento significativo.

## 2.7 Attuatori

Un attuatore è un dispositivo attraverso cui un agente agisce su un ambiente. In senso lato, un attuatore è definito come un qualsiasi dispositivo che converte energia da una forma ad un'altra, in modo che questa agisca nell'ambiente fisico al posto dell'uomo. Anche un meccanismo che mette qualcosa in azione automaticamente è detto attuatore. Nel nostro caso parleremo di attuatori come una scheda configurata appositamente per azionare dei

dispositivi che si trovano in un ambiente lavorativo. Come esempio abbiamo considerato i seguenti meccanismi:

- Luci all'interno della stanza, simulate attraverso dei LED a 12v;
  - Impianto di ventilazione, simulato attraverso tre ventole a 5v;
  - Sistema di apertura degli infissi, simulato attraverso un servo motore.

I comandi di accensione/spegnimento dei diversi dispositivi potranno arrivare in due modalità:

- Direttamente on-board, attraverso dei bottoni e potenziometri appositi messi sulla scheda montata. In modo da lasciare un certo grado di libertà all'interazione umana. Con Arduino è molto semplice gestire questa parte:

```
//CONTROLLO IL POTENZIOMETRO E SETTO IL SERVOMOTORE

void setServo(){

    valServo = analogRead(potpin);           // reads the value of the
                                              potentiometer (value between 0 and 1023)

    valServo = map(valServo, 0, 1023, 0, 179); // scale it to use it with
                                              the servo (value between 0 and 180)

    myservo.write(valServo);                // sets the servo position

    according to the scaled value

    delay(15);

}

//CON IL BOTTONE VADO AD AZIONE LE VENTOLE

void setFan(){

    valFan = digitalRead(buttonPin);

    //check if input is HIGH

    if ((valFan == HIGH) && (old_val == LOW)) {

        buttonState = 1 - buttonState; //SE HO UN PASSAGGIO DA ALTO A BASSO

        INVERTO LO STATO DEL BOTTONE
    }
}
```

```
    delay(50);  
}  
  
old_val = valFan;  
  
if (buttonState == 1) {  
    // turn LED on:  
    digitalWrite(fanPin, HIGH);  
}  
else {  
    // turn LED off:  
    digitalWrite(fanPin, LOW);  
}  
}content...
```

- Attraverso il portale Web o l'applicazione Android. La scheda dispone di una Ethernet Shield con la quale è stata possibile creare una configurazione sia client (per inviare i dati dall'USB Host), sia server per permettere la ricezione dei dati utili per azionare gli attuatori. Il codice utilizzato per instaurare il server su Arduino è il seguente:

```
...  
//DICHIARAZIONE DELLE VARIABILI  
EthernetClient client;      //Dichiaro una variabile globale per il client  
ethernet  
EthernetServer server(80); //Port 80 is http.  
...  
client = server.available();  
if (client){  
    Serial.println("Richiesta in arrivo..");  
    WaitForRequest(client);  
    ParseReceivedRequest();  
    PerformRequestedCommands();
```

```
    client.stop();
}

//ATTENDO UNA RICHIESTA DAL SERVER
//DOVREI RICEVERE UN BUFFER CONTENENTE "GET /cmd/param1/param2 HTTP/1.0".
void WaitForRequest(EthernetClient client) {
    bufferSize = 0;

    while (client.connected()) {
        if (client.available()) {
            char c = client.read();
            if (c == '\n')
                break;
            else
                if (bufferSize < bufferMax)
                    buffer[bufferSize++] = c;
                else
                    break;
        }
    }

    PrintNumber("bufferSize", bufferSize);
}
```

Con la seguente funzione si controlla la richiesta ricevuta e si leggono i vari comandi arrivati. Il server deve inviare una richiesta nel formato GET cmd/param1/param2 HTTP/1.0.

```
void ParseReceivedRequest() {
    Serial.println("in ParseReceivedRequest");
    Serial.println(buffer);

    char* slash1;
    char* slash2;
```

```
char* slash3;
char* space2;

slash1 = strstr(buffer, "/") + 1; // Look for first slash
slash2 = strstr(slash1, "/") + 1; // second slash
slash3 = strstr(slash2, "/") + 1; // third slash
space2 = strstr(slash2, " ") + 1; // space after second slash (in case
                                 there is no third slash)
if (slash3 > space2) slash3=slash2;

PrintString("slash1",slash1);
PrintString("slash2",slash2);
PrintString("slash3",slash3);
PrintString("space2",space2);

// strncpy does not automatically add terminating zero, but strncat does!
So start with blank string and concatenate.

cmd[0] = 0;
param1[0] = 0;
param2[0] = 0;
strncat(cmd, slash1, slash2-slash1-1);
strncat(param1, slash2, slash3-slash2-1);
strncat(param2, slash3, space2-slash3-1);

PrintString("cmd",cmd);
PrintString("param1",param1);
PrintString("param2",param2);
}
```

Con il seguente codice si controlla che comando ho ricevuto e si esegue la relativa operazione.

```
void PerformRequestedCommands() {
    if ( strcmp(cmd,"digitalWrite") == 0 ) RemoteDigitalWrite();
```

```
if ( strcmp(cmd,"digitalRead") == 0 ) RemoteDigitalRead();
if ( strcmp(cmd,"analogRead") == 0 ) RemoteAnalogRead();
if ( strcmp(cmd,"analogWrite") == 0 ) RemoteAnalogWrite();
}
```

Se si riceve il comando digitalWrite viene eseguita la seguente funzione che scrive il valore param2 sul pin param1.

```
void RemoteDigitalWrite(){
    int ledPin = param1[0] - '0'; // Param1 should be one digit port
    int ledState = param2[0] - '0'; // Param2 should be either 1 or 0
    digitalWrite(ledPin, ledState);

    //-- Send response back to browser --
    server.print("D");
    server.print(ledPin, DEC);
    server.print(" is ");
    server.print( (ledState==1) ? "ON" : "off" );

    //-- Send debug message to serial port --
    Serial.println("RemoteDigitalWrite");
    PrintNumber("ledPin", ledPin);
    PrintNumber("ledState", ledState);
}
```

Se si riceve il comando digitalRead, viene eseguita questa funziona che legge il pin param1.

```
void RemoteDigitalRead(){
    int ledPin = param1[0] - '0'; // Param1 should be one digit port
    int ledState; // Param2 should be either 1 or 0
    ledState = digitalRead(ledPin);
```

```
//-- Send response back to browser --
server.print("D");
server.print(ledPin, DEC);
server.print(" is ");
server.print( (ledState==1) ? "ON" : "off" );

//-- Send debug message to serial port --
Serial.println("RemoteDigitalRead");
PrintNumber("ledPin", ledPin);
PrintNumber("ledState", ledState);
}
```

Se si riceve il comando analogRead, viene eseguita questa funzione che legge il valore analogico del pin param1.

```
void RemoteAnalogRead(){
    int analogPin = param1[0] - '0'; // Param1 should be one digit analog
    port
    int analogValue = analogRead(analogPin);

    //-- Send response back to browser --
    server.print("A");
    server.print(analogPin, DEC);
    server.print(" is ");
    server.print(analogValue,DEC);

    //-- Send debug message to serial port --
    Serial.println("RemoteAnalogRead");
    PrintNumber("analogPin", analogPin);
    PrintNumber("analogValue", analogValue);
}
```

Se si riceve il comando analogWrite, viene eseguita questa funzione che scrive il valore

param2 sul pin analogico param1.

```
void RemoteAnalogWrite() {
    int analogPin = param1[0] - '0'; // Param1 should be one digit analog
    port
    int analogValue = param2[2] - '0';
    analogValue += (param2[1] - '0')*10;
    analogValue += (param2[0] - '0')*100;
    analogWrite(analogPin, analogValue);

    //-- Send response back to browser --
    server.print("A");
    server.print(analogPin, DEC);
    server.print(" is ");
    server.print(analogValue,DEC);

    //-- Send debug message to serial port --
    Serial.println("RemoteAnalogWrite");
    PrintNumber("analogPin", analogPin);
    PrintNumber("analogValue", analogValue);
}
```

# Capitolo 3

## Testing e risultati sperimentali

### 3.1 Introduzione

In questo capitolo vengono presentati i test effettuati sulle tecnologie esposte nel capitolo precedente. In particolare nella sezione 3.2 vengono mostrate le misure effettuate sulle singole schede per testare i consumi, per verificare eventuali bug di comunicazione e di programmazione e di mostrare problemi di memoria sorti durante la progettazione. Nella sezione 3.2.4 vengono spiegati i test effettuati con l'applicazione Android. Nella sezione 3.2.5 sono mostrati i test fatti sulle reti di Bayes per notificare eventuali falsi positivi durante l'interpretazione delle letture dei sensori.

### 3.2 Testing

La board Arduino UNO oltre al microcontrollore integra altri componenti come il convertitore seriale-USB, che non possono essere messi in standby direttamente dalla board e quindi continuano a consumare corrente anche quando la stazione è nella fase di “sleep”; in particolare, mettendo la scheda in standby il consumo si abbassa solo di 10mA: 34mA al posto di 44mA. Con le batterie stilo AA da che si trovano in commercio l'autonomia delle schede

arriva solo a poche ore (al massimo un paio di giorni).

Se si considera che una board con alcuni componenti installati consuma circa 50mA, con una batteria da 2000mA si potrebbe alimentare la scheda Arduino per circa 40 ore ( $2000\text{mAh} / 50\text{mA} = 40\text{h}$ ). Il progetto non è composto solo da Arduino, ma anche da qualche shield, led o componente aggiuntivo che fa aumentare l'assorbimento.

### 3.2.1 Test sulle schede Arduino con il modulo XBEE

Alcuni test sono stati effettuati per controllare il consumo di ogni scheda progettata. I primi test riguardano il consumo di una scheda Arduino con una Wireless Shield e un modulo Xbee montato sopra di essa. Il modulo invia ogni 60 secondi circa un dato ad un altro modulo Xbee ricevente. Si sono fatti due test differenti provando a mettere in sleep la board (lasciando in modalità NO SLEEP il modulo Xbee). Nelle due tabelle riportate di seguito si può vedere come il processo di scarica di una batteria da 12v sia abbastanza veloce non permettendo così un installazione outdoor del dispositivo progettato.

Tabella 3.1: Arduino con **sleep** di 60 secondi

| Tempo (minuti) | Stato batteria (12v) |
|----------------|----------------------|
| 0              | 100.00%              |
| 60             | 87.99%               |
| 120            | 82.20%               |
| 180            | 79.64%               |
| 240            | 74.87%               |
| 300            | 67.36%               |
| 360            | 61.50%               |
| 420            | 55.64%               |
| 480            | 49.78%               |

Tabella 3.2: Arduino con **delay** di 60 secondi

| Tempo (minuti) | Stato batteria (12v) |
|----------------|----------------------|
| 0              | 100.00%              |
| 60             | 85.51%               |
| 120            | 62.42%               |
| 170            | 48.43%               |

Il grafico in fig. 3.1 mette a confronto i dati appena riportati:

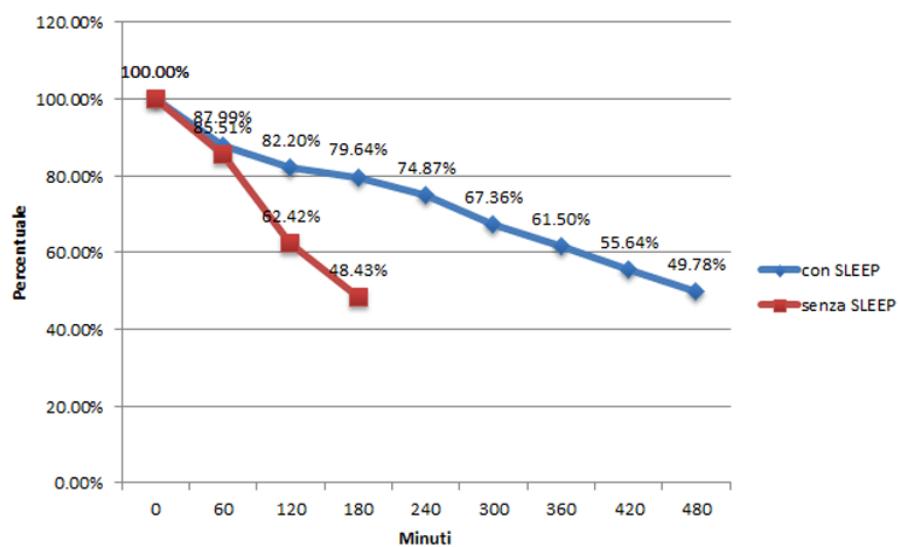


Figura 3.1: Grafico del processo di scarica di una batteria da 12V con il modulo Xbee

Per ottenere risultati più soddisfacenti si dovrebbe utilizzare una versione di Arduino standalone (cioè senza board) e mettere in sleep anche il dispositivo Xbee. In questa modalità, sia il microcontrollore Arduino che il modulo Xbee consumano solo qualche uA (<http://www.jsjf.demon.co.uk/xbee/xbee.pdf>). Con questo valore si arriverebbe ad un consumo totale decisamente inferiore. Per esempio se il consumo fosse attorno a 10uA durante la modalità sleep e invio dei dati ogni secondo con un consumo di circa 50mA, allora una batteria da 2000mAh si scaricherebbe in 2400h (cioè circa 100 giorni).

Alcuni test sulla durata delle batterie con Xbee sono disponibili online sul sito:  
<http://www.faludi.com/projects/arduino-and-xbee-battery-test-results/>.

### 3.2.2 Test sulle schede Arduino con il modulo WiFly

Lo stesso test è stato effettuato con il modulo WiFly esposto nella sezione precedente. Anche in questo caso si invia un pacchetto di dati al server ogni minuto.

Tabella 3.3: Arduino con **delay** di 60 secondi

| Tempo (minuti) | Stato batteria (12v) |
|----------------|----------------------|
| 0              | 100.00%              |
| 60             | 93.43%               |
| 120            | 84.00%               |
| 240            | 69.77%               |
| 300            | 62.10%               |
| 360            | 50.60%               |

Tabella 3.4: Arduino con **sleep** di 60 secondi

| Tempo (minuti) | Stato batteria (12v) |
|----------------|----------------------|
| 0              | 100.00%              |
| 60             | 96.31%               |
| 120            | 91.43%               |
| 180            | 88.08%               |
| 240            | 84.12%               |
| 600            | 60.32%               |
| 660            | 56.35%               |
| 720            | 52.40%               |

Il grafico in fig. 3.2 mette a confronto i dati appena riportati:

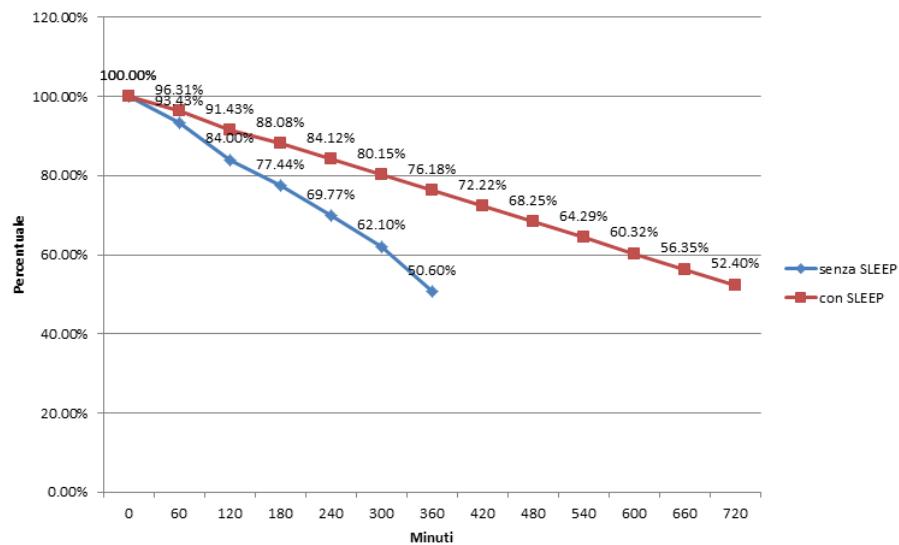


Figura 3.2: Grafico del processo di scarica di una batteria da 12V con il modulo WiFi

Come detto nel precedente test, anche per questo modulo c'è la possibilità di metterlo in sleep in una configurazione standalone (con solo il microcontrollore Arduino). Il datasheet di questo modulo dichiara un consumo di 4uA in modalità sleep mode, 38mA in ricezione e 180mA in trasmissione. Come si può notare il consumo in trasmissione è minore rispetto ai dispositivi Xbee perchè il dispositivo WiFly entra in una modalità di basso consumo automaticamente quando ha inviato i dati. Con i test effettuati non è possibile vedere tale differenza perchè avremmo bisogno di una configurazione standalone con la quale è possibile mettere in sleep i dispositivi indipendentemente dal consumo della board e delle shield installate sopra di essa.

### 3.2.3 Problemi di memoria

Un altro problema sorto durante la programmazione delle schede è stato quello della memoria. Arduino UNO dispone di una memoria flash di 32KByte. Per applicazioni importanti che richiedono diverse librerie e diversi dispositivi, è un po limitante. Nel nostro caso abbiamo riscontrato delle problematiche durante la progettazione della scheda di rilevamento ambientale 2.2.1. Sono stati necessari diversi accorgimenti a riguardo, come limitare l'uso delle variabili, la loro dimensione, ma anche l'uso delle librerie in modo limitato (non si è potuto implementare sulla scheda la bufferizzazione su scheda SD in mancanza di connessione di rete perchè la libreria necessaria per la gestione di questa memoria era troppo grande).

Utilizzando invece la scheda Arduino MEGA ADK non si dovrebbe incorrere a queste problematiche perchè la memoria flash è di 256Kbyte, abbastanza sufficiente anche per applicazioni più consistenti.

Un'altra questione sollevata durante la programmazione della scheda di rilevamento ambientale è stata quella della memoria del dispositivo WiFly. Esso dispone di una RAM da 128Kbyte che viene utilizzata per memorizzare le variabili e il programma utilizzato per la comunicazione Wireless. Per questo motivo è stato necessario un dimensionamento dei buf-

fer che sono state utilizzati durante la comunicazione. Se si eccede con le dimensioni, si sovrascrive la memoria programma e il dispositivo si resetta automaticamente interrompendo ciò che stava facendo, rendendo impossibile l'invio dei dati.

### 3.2.4 Test dell'applicazione Android

L'applicazione sviluppata è stata provata in diversi ambienti e con differenti tipologie di connessioni. Non sono stati riscontrati problemi rilevanti. I tempi di risposta dal server per inviare sia i dati utili per le letture che l'evento più probabile sono pressochè immediati (circa 1000-1100 millisecondi), mentre la risposta degli attuatori si aggira al massimo attorno ai due secondi.

### 3.2.5 Test delle reti di Bayes

La rete di Bayes sviluppata necessita di un insieme di dati di allenamento piuttosto grande per una corretta validazione; nel nostro caso per scelta implementativa il dataset è stato costruito partendo da dati raccolti durante le prime settimane di funzionamento del sistema. Questa scelta ha il vantaggio di permettere l'apprendimento degli scenari sin da subito, ma espone al rischio di rilevare eventi non corretti, dovuti principalmente sia alle limitate dimensioni del dataset ma anche a una rilevazione di dati su un periodo limitato di tempo (i dati raccolti durante un periodo dell'anno sono sicuramente differenti da un altro periodo).

Gli scenari ipotizzati sono stati i seguenti:

- Nel caso i sensori rilevino alta temperatura all'interno del locale e una medio/alta presenza di fumo allora è probabile ci sia un incendio in atto (FIRE);
- Nel caso i sensori rilevino un'alta concentrazione di gas allora probabilmente c'è una fuoriuscita di gas (GAS LEAK);
- Nel caso i sensori rilevino presenza all'interno dei locali durante orario non lavorativo allora probabilmente c'è un intrusione (INTRUSION);

- Nel caso i sensori rilevino una temperatura bassa rispetto la media, un umidità non nella norma, ecc..., allora probabilmente c'è una condizione ambientale da risolvere (BAD CONDITIONS);
- Nel caso i sensori rilevino presenza nella stanza durante l'orario di lavoro e il suono rilevato è basso allora si in una condizione di lavoro (WORK), altrimenti se il suono è medio probabilmente c'è una riunione (CHAT);

I test effettuati per valutare l'efficienza della rete di Bayes sono stati solo su alcuni eventi di alto livello (altri sono difficili da simulare). Abbiamo sottoposto i sensori a diverse prove per simulare le situazioni sopra esposte, che potrebbero andare a stimolare la rete progettata. Per ogni test sono stati valutate diverse situazioni in base alla frequenza di campionamento dei sensori. Più questa è alta, più è accurata la veridicità dell'evento che vogliamo rilevare. Di contro, se aumentiamo questo valore, aumenta anche il consumo dei dispositivi perché campionano i dati più velocemente. Quindi bisogna trovare un compromesso tra frequenza di campionamento e consumo.

Per l'evento **GAS LEAK** abbiamo simulato un fuori uscita di gas bruciando vicino al sensore dello stagno che, producendo del fumo, faceva salire il livello di gas all'interno della stanza. I risultati sono stati i seguenti:

Tabella 3.5: Test dell'evento GAS LEAK su 20 casi per frequenza di campionamento

| Frequenza di campionamento (sec) | Eventi falsi | Eventi veri |
|----------------------------------|--------------|-------------|
| 20                               | 3            | 17          |
| 60                               | 4            | 16          |
| 300                              | 4            | 16          |

Per gli eventi **WORK** e **CHAT** abbiamo simulato una normale attività lavorativa all'interno del locale, sia solamente attraverso il riconoscimento con il sensore PIR ma anche con il passaggio di un TAG RFID. I risultati sono meno soddisfacenti degli altri perché ci

sono due variabili in più da prendere in considerazione (PIR e RFID). Infatti se la rete rileva solo un movimento del sensore PIR restituisce una probabilità bassa dell'evento WORK o CHAT (attorno a 0.5). Mentre se c'è anche il TAG RFID la probabilità aumenta anche fino a 1.

Tabella 3.6: Test dell'evento WORK su 20 casi per frequenza di campionamento

| Frequenza di campionamento (sec) | Eventi falsi | Eventi veri |
|----------------------------------|--------------|-------------|
| 20                               | 3            | 17          |
| 60                               | 4            | 16          |
| 300                              | 6            | 14          |

Tabella 3.7: Test dell'evento CHAT su 20 casi per frequenza di campionamento

| Frequenza di campionamento (sec) | Eventi falsi | Eventi veri |
|----------------------------------|--------------|-------------|
| 20                               | 0            | 20          |
| 60                               | 2            | 18          |
| 300                              | 3            | 17          |

Per l'evento **INTRUSION** abbiamo simulato, durante l'orario non lavorativo, un passaggio di una persona all'interno del locale e quindi un conseguente rilevamento da parte del sensore PIR. I risultati sono i seguenti:

Tabella 3.8: Test dell'evento INTRUSION su 20 casi per frequenza di campionamento

| Frequenza di campionamento (sec) | Eventi falsi | Eventi veri |
|----------------------------------|--------------|-------------|
| 20                               | 1            | 19          |
| 60                               | 0            | 0           |
| 300                              | 2            | 2           |

Come si può notare nelle tabelle qui sopra, la bontà dei risultati si ottiene con una frequenza di campionamento elevata. Nella tabella e nei grafici seguenti vengono mostrati le percentuali degli eventi verificati.

Tabella 3.9: Percentuali sulla veridicità dei test in base alla frequenza di campionamento

| Frequenza di campionamento (sec) | Eventi falsi | Eventi veri |
|----------------------------------|--------------|-------------|
| 20                               | 8.75%        | 91.25%      |
| 60                               | 12.50%       | 87.50%      |
| 300                              | 18.75%       | 81.25%      |

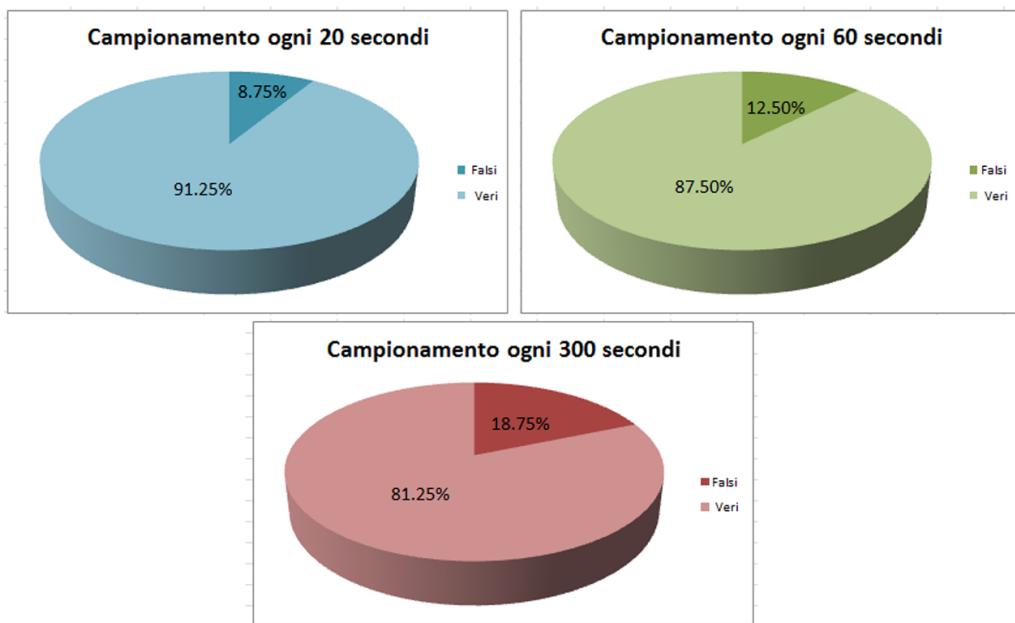


Figura 3.3: Grafici dei test eseguiti sulla rete di Bayes

Nel complesso abbiamo un'accuratezza della rete che si aggira attorno al 86% (come mostrato in fig.3.4).

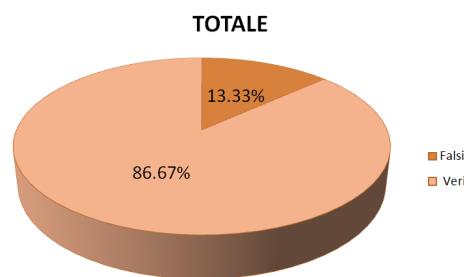


Figura 3.4: Accuratezza della rete di Bayes

Per una frequenza di campionamento elevata, i dati elaborati dalla rete vengono verificati con maggior successo. Questo è dovuto al fatto che i dati vengono elaborati istantaneamente rispetto alle rilevazioni. Mentre se la frequenza di campionamento è troppo elevata si rischia di non rilevare l'evento correttamente. Ad esempio per una fuga di gas o per un movimento rilevato dal PIR, non è sufficiente campionare il dato ogni cinque minuti. Si rischia di non prendere in considerazione dati che in realtà sono utili per segnalare determinati eventi.

Un ottimo compromesso tra consumi e bontà dei risultati è tenere la frequenza di campionamento intorno al minuto in modo tale da permettere l'elaborazione di tutti i dati, ma allo stesso tempo di mandare in sleep i dispositivi che consumano di più.

# Conclusioni e sviluppi futuri

## 4.1 Obiettivi raggiunti

Lo scopo di questa Tesi è mostrare come le piattaforme Arduino e Android possano essere utilizzate in un sistema di Ambient Intelligence per il monitoraggio dei sensori e l'interazione con gli attuatori. Con Arduino è stato possibile integrare i sensori preposti in fase di progettazione e di far comunicare le schede attraverso le tre tecnologie (Ethernet, WiFi e Xbee) esposte nei capitoli precedenti. Con Android invece è possibile ora monitorare lo stato della stanza in cui sono inseriti i sensori ovunque ci si trovi. Unico requisito è avere una connessione Internet.

A scopo dimostrativo è stato realizzato un servizio di monitoraggio ambientale basato su reti di Bayes. Il sistema realizzato riesce nell'intento di contribuire all'esplorazione di nuove tecnologie per l'Ambient Intelligence. La soluzione proposta è il primo tassello di un sistema per l'apprendimento di comportamenti e abitudini degli utenti, in grado di anticipare i bisogni migliorando le proprie performance nel tempo. Il numero di errori compiuti dal sistema si può ridurre implementando un sistema di apprendimento automatico del dataset e quindi un auto adattamento alle abitudini delle persone che vivono nel locale monitorato. Questo è un fattore chiave per permettere la diffusione di sistemi di questo tipo, dato che nessuno gradirebbe un sistema che continuamente tenta di predire, spesso sbagliando, ciò che si vuol fare. È importante sottolineare quanto un sistema di questo tipo necessiti di test approfonditi e duraturi, in modo da perfezionare l'apprendimento e valutare più accu-

ratamente i parametri di sistema. Avere a disposizione un dataset di maggiori dimensioni darebbe sicuramente più valore alla fase di validazione, permettendo valutazioni sulla bontà del sistema più accurate.

## 4.2 Sviluppi futuri

Un possibile sviluppo potrebbe essere quella di portare il server, che attualmente è in esecuzione su un pc, direttamente su un controller embedded (tipo Raspberry PI), ottimizzandone le performance e l'adattabilità in qualsiasi contesto.

Un altro sviluppo potrebbe essere quello di integrare nuovi sensori come una videocamera IP, ampliare il raggio d'azione del lettore RFID, migliorare l'applicazione Android per monitorare diversi ambienti, integrare nel sistema attuatori reali (e non simulati).

Un ulteriore sviluppo potrebbe essere quello di applicare anche altre tecniche di apprendimento della rete Bayesiana. È chiaro che se la rete inizia ad avere più sensori o più variabilità si potrebbe pensare di effettuare un apprendimento automatico in cui servirebbero dei feedback degli utenti. Cioè se viene notificato un evento FIRE e l'utente ne sancisce la bontà, si può utilizzare tale informazione per migliorare le prestazioni della rete.

# Indice delle Figure

|      |  |    |
|------|--|----|
| 1.1  | RepRap - Stampante 3D open source a basso costo . . . . .                | 7  |
| 1.2  | Arduino Uno . . . . .  | 8  |
| 1.3  | Sketch d'esempio con l'Arduino IDE . . . . .                             | 9  |
| 1.4  | Arduino MEGA ADK dall'alto . . . . .                                     | 10 |
| 1.5  | Esempio di Arduino Extreme Shielding . . . . .                           | 12 |
| 1.6  | scheda Flyport prodotta da OpenPicus . . . . .                           | 13 |
| 1.7  | Architettura di una scheda FlyPort . . . . .                             | 15 |
| 1.8  | Wasp mote Sensorboards . . . . .   | 16 |
| 1.9  | Scheda Raspberry PI . . . . .  | 17 |
| 1.10 | esempio di smart home . . . . .  | 22 |
| 1.11 | Flusso delle informazioni e schema generale di un sistema AmI . . . . .  | 24 |
| 1.12 | Topologie di reti WSN . . . . .  | 26 |
| 1.13 | Intereazione tra mondo fisico e il mondo web . . . . .                   | 29 |
| 1.14 | Schema di funzionamento di dispositivi Mavia . . . . .                   | 31 |
| 1.15 | Architettura Android . . . . .   | 34 |
| 2.1  | Architettura generale del sistema considerato . . . . .                  | 40 |
| 2.2  | Scheda per il rilevamento ambientale realizzata in laboratorio . . . . . | 41 |
| 2.3  | Sensore DHT22 . . . . .  | 42 |
| 2.4  | Sensore di precisione di luminosità . . . . .                            | 43 |

|  |    |
|--|----|
| 2.5 Sensore di fumo . . . . .  | 44 |
| 2.6 Schema di comunicazione della scheda . . . . .                                   | 45 |
| 2.7 Schema circuitale della scheda . . . . .   | 46 |
| 2.8 Schema circuitale della scheda . . . . .   | 47 |
| 2.9 Scheda con Xbee ricevente e invia su Ethernet realizzata. . . . .                | 48 |
| 2.10 Scheda di riconoscimento con Xbee mittente realizzata. . . . .                  | 48 |
| 2.11 PIR motion sensor . . . . .   | 49 |
| 2.12 RFID shield già montata sopra la scheda Arduino . . . . .                       | 52 |
| 2.13 Precision sound sensor . . . . .  | 52 |
| 2.14 Scheda con prototipi di attuatori realizzata in laboratorio . . . . .           | 54 |
| 2.15 Come collegare il transistor per funzionare con Arduino . . . . .               | 55 |
| 2.16 Schema circuitale della scheda . . . . .  | 60 |
| 2.17 Ethernet Shield di Arduino . . . . .  | 61 |
| 2.18 Modulo WiFly RN-XV della Roving Networks . . . . .                              | 67 |
| 2.19 Wireless Shield di Arduino . . . . .  | 67 |
| 2.20 Modulo Xbee . . . . .   | 72 |
| 2.21 Modello ER del database utilizzato . . . . .                                    | 76 |
| 2.22 Pagina di login . . . . .   | 80 |
| 2.23 Pagina principale del sito Web . . . . .  | 81 |
| 2.24 Alcuni grafici che vengono visualizzati dal sito WEB . . . . .                  | 82 |
| 2.25 Form di login alla prima apertura dell'applicazione . . . . .                   | 84 |
| 2.26 Activity principale dell'applicazione . . . . .                                 | 85 |
| 2.27 Activity dove si possono modificare le impostazioni dell'applicazione . . . . . | 86 |
| 2.28 Activity dove si può modificare lo stato degli attuatori . . . . .              | 86 |
| 2.29 Notifica d'incendio ricevuta al terminale . . . . .                             | 87 |
| 2.30 schermata interfaccia di WEKA explorer . . . . .                                | 91 |
| 2.31 Formula di Bayes . . . . .  | 92 |

|   |     |
|---|-----|
| 2.32 Esempio di rete bayesiana . . . . .  | 93  |
| 2.33 Rete di Bayes usata per il progetto . . . . .                                | 96  |
| 2.34 Struttura di un modulo NAM . . . . .   | 98  |
| 2.35 Esempio di un modulo NAM . . . . .   | 99  |
| <br>  |     |
| 3.1 Grafico del processo di scarica di una batteria da 12V con il modulo XBee . . | 110 |
| 3.2 Grafico del processo di scarica di una batteria da 12V con il modulo WiFi .   | 112 |
| 3.3 Grafici dei test eseguiti sulla rete di Bayes . . . . .                       | 117 |
| 3.4 Accuratezza della rete di Bayes . . . . .                                     | 117 |

# Bibliografia

- [1] What is commodity hardware, “<http://wiki.answers.com/q/what-is-commodity-hardware>,” 2012.
- [2] Enrico Signoretti, “Reality check - <http://juku.it/articles/lopen-hardware-e-il-futuro-dellinnovazione-tecnologica.html>,” 2012.
- [3] Open Source HardWare Alliance, “[www.oshwa.org](http://www.oshwa.org)” .
- [4] Arduino, “Arduio site - [www.arduino.cc](http://www.arduino.cc),” 2012.
- [5] Wikipedia, “List of arduino compatibles - <http://en.wikipedia.org/w/index.php?title=list-of-arduino-compatibles&oldid=505397266>,” 2012.
- [6] Arduino, “Arduio hardware - [www.arduino.cc/en/main/hardware](http://www.arduino.cc/en/main/hardware),” 2012.
- [7] Massimo Banzi, *BetaBook, il manuale di Arduino*, Apogeo, 2009.
- [8] Massimo Banzi, *Getting Started with Arduino*, Make Books, 2009.
- [9] Maik Schmidt, *Il manuale di Arduino*, Apogeo, 2011.
- [10] Massimo Banzi, “Arduino la guida ufficiale,” 2012.
- [11] Openpicus, “<http://www.openpicus.com/site/technology/overview>,” 2012,  
<http://www.openpicus.com/site/technology/overview>.

- [12] <http://en.wikipedia.org/wiki/Openpicus>, “<http://en.wikipedia.org/wiki/openpicus>,” 2012, <http://en.wikipedia.org/wiki/Openpicus>.
- [13] Libelium, “<http://www.libelium.com/products/wasp mote>,” 2012, <http://www.libelium.com/products/wasp mote>.
- [14] Luca Rossi, “Libelium si allontana dal progetto arduino con waspmote,” 2012, <http://it.emcelettronica.com/liberl um-si-allontana-dal-progetto-arduino-con-il-nuovo-wasp mote>.
- [15] Raspberry, “<http://www.raspberrypi.org>” .
- [16] DIY, “<http://www.diynetwork.com/>” .
- [17] “(<http://hackaday.com/2011/07/19/chilean-teen-builds-automatic-earthquake-alarm/>),” 2012.
- [18] “<http://it.emcelettronica.com/quadricottero-basato-su-arduino>,” 2012.
- [19] “<http://www.youtube.com/watch?v=atxx8sewui4>,” 2012.
- [20] “<http://www.youtube.com/watch?v=pynnonagu5i>,” 2011.
- [21] “<http://blog.ponoko.com/2012/07/28/diy-dna-with-arduino/>,” 2012.
- [22] “(<http://www.clloks.com/2010/09/sign-language-glove/>),” 2012.
- [23] “(<http://arduino.cc/blog/2010/07/29/txtbomber-prints-messages-on-walls/>),” 2012.
- [24] “(<http://blog.makezine.com/2008/09/29/house-plant-measures-the/>),” 2012.
- [25] “(<http://arduino.cc/blog/2011/05/30/tutorial-arduino-uno-googles-adk/>),” 2011.
- [26] “<http://digitalhabits.it/>,” 2012.
- [27] Mark Weiser, “Hot topics: Ubiquitous computing,” 1993.

- [28] S. Das D.J. Cook, *Smart Environments: Technologies, Protocols and Applications.*, John Wiley and Sons - Ed. 1, 2003.
- [29] Emile Aarts, “Ambient intelligence: A multimedia perspective.,” 2004.
- [30] A. Mukherjee D. Saha, “Pervasive computing: A paradigm for the 21st century,” 2003.
- [31] Marco Beltrame Università degli studi di Pavia, *Sviluppo di una rete wireless di sensori per il monitoraggio di strutture in tempo reale*, Ph.D. thesis, 2008.
- [32] Juniper, “Opportunities in smart systems and m2m - mtantow.com/2011/01/opportunities-in-smart-systems-and-m2m/,” 2011.
- [33] Kevin Ashton, *That 'Internet of Things' Thing*, RFID Journals, 2009.
- [34] Movia, “www.movia.biz,” .
- [35] Guglielmo, “www.guglielmo.biz,” .
- [36] Nerdydog, “(http://www.domotichome.net/),” .
- [37] “http://blog.makezine.com/arduino/grbl/,” .
- [38] CNC Controller, “(http://hackaday.com/2012/06/08/android-cnc-controller/),” .
- [39] Game controller, “(http://www.dancingpixelstudios.com/sixaxiscontroller/about.html),” .
- [40] “(http://arduino.cc/playground/main/dhtlib),” .
- [41] “www.zigbee.org/,” .
- [42] Sito ufficiale di WEKA, “http://www.cs.waikato.ac.nz/ml/weka/,” .
- [43] I.H.Witten E E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.

- [44] Michele Amoretti, “<http://dsg.ce.unipr.it>,” .

# Ringraziamenti

Un ignaro lettore potrebbe pensare che, dopo aver scritto una tesi (indipendentemente dalla sua qualità intrinseca), redigere una paginetta di ringraziamenti sia qualcosa di semplice e immediato e, sinceramente, anch'io la pensavo allo stesso modo. Bene, mi sbagliavo. Quelle poche persone che prenderanno in mano questa tesi, infatti, quasi sicuramente finiranno a leggere queste righe; non per chissà quale motivo, ma perchè sono le uniche cose realmente comprensibili se non per chi ha frequentato il mio corso di laurea. Quindi ora mi trovo di fronte al cursore lampeggiante e mi sento un po' emozionato e allo stesso tempo terrorizzato di dimenticare qualcuno di realmente importante.

Per prima cosa vorrei esprimere la mia gratitudine al Prof. Conte, relatore della mia tesi, per l'aiuto fornитomi durante la stesura della tesi. Desidero, inoltre, ringraziare Marco e Michele per la pazienza e il prezioso aiuto datomi durante questi mesi di lavoro a questo progetto.

Ringrazio in particolar modo i miei genitori e mio fratello per avermi dato la possibilità di compiere questo percorso, per essermi sempre stati vicini e per avermi supportato.

Un ringraziamento a tutti gli amici di Cabriolo, in particolare il Don che mi ha sempre spronato e mi è sempre stato vicino in questi anni ed insieme a Massi e Karol, grazie alla loro amicizia fraterna, ed i loro giudizi e consigli, hanno contribuito ad essere quello che sono. Grazie anche a Sabri, Silvia, Maddy, Mary, Erica, Sheggy, Scalzo, Simo, Tommy, Paro, Bruno, Eli, Vale, Vane, al gruppo dei giovani e tutti quelli che mi hanno sopportato in questi anni.

Un ringraziamento a tutti i compagni di università, con cui ho condiviso questi anni, in particolare Pesca, Andrea e il Gabro per ogni momento passato insieme, dai più spensierati a scrivere aforismi su fogli di carta a quelli più seri passati preparando degli esami. Un grazie anche a Fede, i Minghi (sia quello buono che quello cattivo), Menoz, Pigno, Gio, Ste, Tuan e tutti gli altri che ci sono stati per un momento o più.

Infine, ma non per importanza, un ringraziamento alla Sere per la sua estrema pazienza, per il suo infinito amore e per la sua vicinanza nonostante io stesso abbia fatto fatica a sopportarmi in certi momenti.

*Mirko*

