L'uomo, i Robot e l'intelligenza artificiale

Conoscenza o apprendimento?

1 - Un robot non può recar danno a un essere umano

né può permettere che, a causa del proprio mancato intervento,

un essere umano riceva danno.

2 - Un robot deve obbedire agli ordini impartiti dagli esseri umani,

purché tali ordini non contravvengano alla Prima Legge.

3 - Un robot deve proteggere la propria esistenza,

purché questa autodifesa non contrasti

con la prima o con la seconda legge.

(*Isaac Asimov*)

Indice

INTRODUZIONE

1 – STORIA

1.1 – I telefoni cellulari

1.2 – Gli Smartphone

2 – IL SOFTWARE

2.1 – Eclipse

2.2 – Android

2.3 – Brian

2.4 – Codice

3 – L' HARDWARE

3.1 – Tecnologia di trasmissione

3.2 – Arduino

3.3 – Bracciale

3.4 – Ciabatta elettrica

3.5 – Funzionamento finale

BIBLIOGRAFIA

INTRODUZIONE

Da molto tempo diversi film di fantascienza descrivono la storia di macchine intelligenti, autonome, alle volte capaci di dare supporto all'uomo, altre volte in grado di distruggerlo.

Siamo nell'anno 2015 è la tecnologia fantascientifica presente solo pochi anni fa nelle nostre televisioni è diventata pura realtà come ad esempio l'uso di cellulari, videofonini, ologrammi, macchine pseudo-intelligenti capaci di svolgere un determinato compito, se programmate per farlo.

Diverse correnti di pensiero si sono sviluppate nel corso degli anni e si dibattono sulla possibilità che un dispositivo elettronico possa essere capace di pensare o meno, e questo è dovuto alla nascita dello studio dell'intelligenza artificiale ed al cercare di insegnare ad un software a rispondere in modo

“*umanamente corretto*” ad una domanda posta.

Un esempio di software “chiacchierone” lo troviamo con *cleverbot*, raggiungibile al sito web *http://www.cleverbot.com/*.

Ma senza esagerare con l'intelligenza del software, diversi programmi sono stati implementati con caratteristiche minori per dare un aiuto più semplicistico all'uomo e tra questi possiamo trovare il software iOS SIRI che è capace di rispondere ad alcune domande generali ed eseguire diversi compiti e risponde simulando una voce umana, a differenza di cleverbot che usa solo il testo.

Questa idea innovativa ha portato lo sviluppo sempre maggiore di tipi di software in grado di rendere più semplice l'uso di un dispositivo.

L'idea di questa tesi è nata dall'aver notato una mancanza in questi software, ovverosia la possibilità di poter porre una nuova domanda senza bisogno di dover premere un tasto dopo che la risposta precedente sia stata completata; implementando

questa piccola funzionalità il software diventerebbe utilizzabile totalmente a distanza e questo può essere fatto realizzando un dispositivo hardware che può comunicare con il telefono stesso.

Per far sì che il metodo possa tornare ancora più utile, verrà realizzato un secondo dispositivo hardware a cui verrà collegato un qualsiasi dispositivo alimentato a corrente elettrica allo scopo di poterlo accendere e spegnere a distanza.

Infine verrà implementato il codice utilizzando la piattaforma Android, che in questo momento è, insieme a iOS, senza dubbio il sistema operativo più diffuso nel mondo degli smartphone.

CAPITOLO 1

Storia

Dalle dimensioni di pochi centimetri di lunghezza, i moderni telefoni concentrano il frutto della una collaborazione di diverse ingegnosità che coinvolgono i rami della chimica, della matematica, della fisica, dell'elettrotecnica e dell'informatica.

L'idea di comunicare a distanza era nata già ai tempi dell'antica Grecia e dell'impero romano. Prima del XX secolo, non essendo ancora conosciute le leggi di Marxwell sulla propagazione delle onde elettromagnetiche, la comunicazione poteva avvenire solo tramite il trasporto del suono attraverso l'aria.

Fu solo nel 1871 che il fiorentino Antonio Meucci riuscì a

dimostrare il funzionamento del suo “telettrofono”, il primo telefono elettrico.

La prima introduzione pratica del telefono in Italia ebbe luogo a Milano il 30 dicembre 1877 quando fu attivata la linea tra due apparecchi costruiti dai fratelli Gerosa che metteva in contatto una caserma dei pompieri con la stazione di Porta Venezia della tranvia interurbana per Monza. Solo dopo un secolo circa (ebbene si!), questi dispositivi infinitamente grandi divennero portatili, tanto da poter tenere un telefono in una mano.

1.1 – I telefoni cellulari

Il telefono cellulare fu inventato da Martin Cooper, direttore della sezione “Ricerca e Sviluppo” della Motorola, che fece la sua prima telefonata da un cellulare il 3 aprile 1973.

Dopo 10 anni la Motorola decise di produrre un modello dal costo di 4000 dollari.

La comunicazione ha attraversato diversi standard “Generation” (G) come il 2G, il 3G, fino a quando pochi mesi fa è stato ufficialmente introdotto il tanto atteso 4G.

Lo standard 2G è basato sull'uso del GSM (Global System for Mobile communications), è il più diffuso al mondo, è uno standard aperto sviluppato dal CEPT e finalizzato dall' ETSI e mantenuto dal consorzio 3GPP (3 Generation Partnership Project) che fu creato per definire le specifiche tecniche dei dispositivi mobili di terza generazione.

Lo standard GSM porta come principale novità una comunicazione di tipo digitale permettendo una maggiore sicurezza dei dati per mezzo della loro cifratura. L'incremento nella velocità di trasmissione dei dati ha consentito la nascita di nuovi servizi come il servizio SMS o la possibilità di poter effettuare traffico a commutazione di pacchetto grazie allo standard GPRS/EDGE sfruttando il dispositivo come modem per scambiare dati (audio e/o immagini) o navigare in Internet.

Con lo standard GSM nasce anche l'uso delle SIM (Subscriber Identity Module) che permette di poter memorizzare i dati dell'abbonato esternamente ed indipendentemente dal telefono utilizzato.

Tutte queste modifiche hanno soppiantato il vecchio standard analogico TACS e da qui la scelta di un nome per uno standard di “nuova generazione”.

Corre l'anno 2006 quando con la 3G arriva anche lo standard UMTS (Universal Mobile Telecommunications System),

evoluzione del GSM, rimanendo compatibile allo standard 3GPP.

La prima rete UMTS al mondo nasce nel Regno Unito con il nome di 3 .

Il miglioramento rispetto alla GSM si trova innanzitutto nella velocità di trasmissione dei dati grazie all'utilizzo del canale di trasmissione radio W-CDMA che rimpiazza il precedente TDMA (Time Division Multiple Access) della tecnologia GSM che permette di raggiungere una velocità teorica massima di 21Mb/s, e questo aggiunge altri servizi utilizzabili come quello relativo all'uso degli MMS.

1.2 – Gli smartphone

Con i progressi dell'ingegneria elettronica e la possibilità di creare dei circuiti integrati di dimensioni sempre minori, nel 1992 IBM lancia sul mercato *Simon*, il primo cellulare intelligente che unisce la mobilità di un telefono cellulare alle capacità di calcolo di un computer ma solamente dal 1997 Simon e i suoi successori furono conosciuti con il nome di *smartphone*.

Gli smartphone hanno come caratteristica principale la presenza di un sistema operativo che li gestisce che è molto simile ai sistemi operativi di un classico computer.

Il primo sistema operativo che si afferma su larga scala, dopo il Symbian per i primi Nokia, è stato BlackBerry 10.

Oggigiorno disponiamo sul mercato di diverse tipologie di smartphone provenienti principalmente di Samsung e HTC,

Nokia, Apple, che utilizzano come sistemi operativi rispettivamente Android, Windows Phone e iOS.

Sebbene la differenza tra un telefono cellullare e uno smartphone non sia stata inizialmente così netta, oggigiorno quest'ultimo può integrare al suo interno, un microdiffusore (che amplifica il segnale audio in uscita), una scheda Bluetooth, una scheda WiFi, uno slot USB, 2 fotocamere, torcia, contapassi, sensore del battito cardiaco, rilevatore di impronte digitali e molto altro, oltre ovviamente il processore, la memoria dati, la ricetrasmittente, un microfono e una batteria, nonostante le dimensioni siano rimaste sotto i 20 cm x 10 cm x 2 cm.

Viene invece a mancare con l'arrivo degli smartphone il tastierino numerico che è stato soppiantato da schermi tattili ad alta definizione, (conosciuti meglio come display *touchscreen*).

I display touchscreen sono dei dispositivi hardware che inglobano capacità sia di input che di output e dal giorno della

loro invenzione ne sono stati prodotti diversi tipi con diverse tecnologie, usi e costi.

I primi furono quelli a sensore magnetico, ad infrarossi e a videocamere, più recentemente fu sviluppato il touchscreen resistivo mentre oggi la tecnologia più diffusa è senz'altro quella capacitiva.

Quest'ultima sfrutta la variazione di capacità dielettrica (tipica dei condensatori) sul vetro del telefono stesso, il quale viene ricoperto da un sottile strato di ossido metallico trasparente (*ITO,* Indium Thin Oxyde) sulla parte interna. Ai quattro angoli del pannello viene applicata una tensione che si propaga uniformemente su tutta la superficie dello schermo grazie alle speciali proprietà di trasporto dell'ossido di metallo e quando il dito o un qualsiasi materiale conduttore di elettricità si avvicina allo schermo si presenta una variazione di capacità superficiale che viene letta da una matrice di condensatori a film posizionati su un pannello posto sotto la superficie del vetro.

Lo schermo resistivo a differenza di quello capacitivo è composto da due strati di materiale conduttivo che, nel momento in cui un oggetto viene premuto sullo schermo, entrano in contatto permettendo al dispositivo di determinare la posizione dell'oggetto grazie alla misura della resistenza di contatto.

CAPITOLO 2

Il Software

Object Oriented Programming (OOP, o programmazione orientata agli oggetti) è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi.

E' particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare.

Un ambito che più di altri riesce a sfruttare i vantaggi della programmazione ad oggetti è quello delle interfacce grafiche.

OOP inoltre fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre, permette una più facile gestione e manutenzione di progetti di grandi dimensioni, favorisce la modularità e il

riuso del codice.

La programmazione ad oggetti prevede di raggruppare in una zona circoscritta del codice sorgente, detta classe, la dichiarazione delle strutture dati e delle procedure che operano su di esse.

Le classi costituiscono modelli astratti che, in esecuzione, vengono invocate per creazione e/o istanziazione di oggetti software relativi alla classe invocata. Gli oggetti, a loro volta, sono dotati di attributi (dati) e metodi (procedure) seguendo la definizione della classe alla quale si riferiscono.

Linguaggi orientati agli oggetti sono ad esempio Java e C++ poiché, usando la sintassi nativa del linguaggio, permettono di implementare i seguenti tre meccanismi:

- *Incapsulamento*
- *Ereditarietà*
- *Polimorfismo*

L'incapsulamento consiste nella separazione della cosiddetta

interfaccia di una classe dalla corrispondente implementazione.
L'ereditarietà permette di definire delle classi a partire da altre già definite.
Il polimorfismo permette di scrivere un *client* che può servirsi di oggetti di classi diverse, ma dotati di un'interfaccia comune.
Come ogni cosa, in informatica, chi possiede vantaggi possiede svantaggi.
Esistono ovviamente anche degli svantaggi, ad esempio alcuni meccanismi inclusi nella gestione degli oggetti causano un overhead in termini di tempo e memoria che, in determinate situazioni, può portare a problemi di efficienza.
Per quanto adesso esposto alcuni linguaggi, come quelli sopracitati, preferiscono un' implementazione ibrida rispetto all' uso di un OOP puro, prevedendo alcuni tipi di dati primitivi che non vengono considerati come oggetti.

2.1 - Eclipse

E' un ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma. Venne ideato da un consorzio di grandi società quali Ericsson, HP, IBM, Intel, MonteVista Software, QNX, SAP e Serena Software e il nome scelto fu *Eclipse Foundation*.
Lo stile fu pensato per seguire la filosofia dell'*open source*.
Eclipse può essere utilizzato per la produzione di software di vario genere, si passa infatti da un completo IDE per il linguaggio Java ad un ambiente di sviluppo per il linguaggio C++ e persino a plug-in che permettono di gestire gli stili XML, JavaScript, PHP o persino di progettare graficamente una GUI per applicazioni Java.
Tutto ciò rende di fatto il software un ambiente RAD (Rapid Application Developmente).
La piattaforma di sviluppo è incentrata sull'uso di plug-in,

ovvero di componenti software ideate per uno specifico scopo, come ad esempio la generazione di diagrammi UML ed in effetti tutta la piattaforma risulta essere un insieme di plug-in che sono liberamente modificabili e sviluppabili.

Essendo scritto in Java, è stato possibile renderlo disponibile per le piattaforme Linux, HP-UX, AIX, Mac OS X e Windows.

2.2 - Android

E' un sistema operativo per dispositivi mobili sviluppato da *Google Inc.* sulla base di un *Kernel Linux*.

E' stato progettato principalmente per smartphone e tablet, con interfacce utente specializzate per televisori (Android TV), automobili (Android Auto), orologi da polso (Android Wear), occhiali (Google Glass) e molto altro ancora.

Seguendo le orme del Kernel Linux, Android è per la quasi totalità un *free and open source software* (ad esclusione, ad esempio, di driver non liberi inclusi per i produttori di dispositivi) ed è distribuito sotto i termini della licenza libera Apache 2.0.

Google aveva intenzione di entrare nel campo della telefonia mobile e così, solo nel 17 Agosto 2005, pensò bene di acquistare l'azienda che si occupava del progetto Android.

La presentazione ufficiale del sistema operativo, la cui famosa icona rappresentativa divenne il robottino verde, avvenne il 5 Novembre 2007.

Da quel giorno sono state sviluppate molte versioni ognuna delle quali con diverse *release* (aggiornamenti), e, similmente a quanto accade per le distribuzioni di Linux, viene seguito un ordine alfabetico e una precisa convenzione per i nomi che, nel caso di Android, sono quelli di dolci!

La versione 1.5 prese il nome *Cupcake* e venne seguita dalla 1.6, *Donut*.

La serie 2.x avrà come nomi *Eclair*, *Froyo*, *Gingerbread* (rispettivamente 2,1, 2.2, 2.3), mentre la 3.0 si chiamerà *HoneyComb*.

La serie più diffusa sui dispositivi mobili è la 4.x con *Ice Cream Sandwich* (4.0), *Jelly Bean* (4.1) e, in seguito ad un accordo con la Nestlè, *Kit Kat* (4.4).

La versione che è stata rilasciata più recentemente è quella che

ha dato inizio alla serie 5.x, è la versione 5.0 ed è conosciuta come *Lollipop*.

La bellezza di Android risiede anche nel fatto che è un sistema che ha coinvolto vaste comunità di sviluppatori, i quali realizzano applicazioni con l'obiettivo di aumentare le funzionalità dei dispositivi. Queste applicazioni vengono scritte soprattutto in linguaggio Java e XML.

Il Kernel Linux su cui è basato contiene librerie e API scritte in C o in C++, mentre il software è in esecuzione su un framework di applicazioni che include librerie Java compatibili con librerie basate su Apache Harmony.

Android utilizza la *Dalvik Virtual Machine* con un compilatore *just-in-time* per l'esecuzione di *Dalvik dex-code*, il quale viene tradotto da codice bytecode Java.

Essendo comunque un linguaggio *Java-based*, anche le applicazioni scritte in C o in C++, devono essere richiamate dal codice Java e, come tutte le chiamate a sistema fatte in questi

linguaggi, devono essere elaborate dalla Java Virtual Machine.

Tutto queste librerie e funzioni, vanno a produrre alla fine un pacchetto Android che sarà un file con estensione *.APK*.

Analizzando la memoria flash dei dispositivi che utilizzano Android vedremo più partizioni come ad esempio */system* che è la cartella utilizzata per il sistema operativo e */data* per i dati utente le applicazioni.

Per permettere la diffusione di programmi Android su vasta scala, Google ha realizzato *Google Play*, un market ufficiale messo a punto esclusivamente per il sistema operativo Android dove è permesso registrare le proprie APP e, nell'Ottobre del 2012, queste hanno raggiunto quota 700.000 unità.

Agli utenti, di norma, non è permesso accedere alle partizioni di sistema, ovverosia l'utente non possiede i privilegi di root non essendo un super-user, se non esclusivamente in modalità di lettura, in pieno stile Linux.

Tuttavia l'accesso come super-utente è possibile sfruttando

alcune falle della sicurezza del sistema stesso e questo viene utilizzato spesso dalla comunità open source per migliorare la capacità dei dispositivi, anche se questo può, in teoria, consentire ai malintenzionati di installare virus e malware.

Per questo motivo alcune aziende hanno provveduto a realizzare alcuni software di antivirus come AVG Technologies, Avast, McAfee.

Tuttavia, secondo un'analisi effettuata nel 2011, gli antivirus sono inutili grazie al principio di minimo privilegio secondo il quale le app non possono agire a livello di kernel, ma soltanto applicativo e quindi nessuna applicazione installata da fonti di terze parti avrebbe i permessi sufficienti per apportare danni permanenti al sistema.

Analizziamo un semplice esempio:

while(true);

questa semplicissima istruzione, utilizzata nel thread principale di un'applicazione, semplicemente occupa CPU senza svolgere

nessun compito: Android, se riscontra una situazione del genere, dopo alcuni cicli sospende momentaneamente l'applicazione e chiede all'utente se vuole continuare a eseguire l'applicazione.

Al fine di migliorare la sicurezza, Google ha introdotto dei meccanismi automatici di analisi del software per bloccare eventuali applicazioni malevoli presenti nel market Google Play che, secondo alcune analisi, possono comunque essere aggirate.

Android è capace di gestire diverse componenti hardware come accelerometri, giroscopi e sensori di prossimità e anche le applicazioni installate sul dispositivo possono avere i permessi per utilizzarli, come ad esempio la regolazione dello schermo da verticale a orizzontale a seconda di come il dispositivo sia orientato.

Passiamo ora ad analizzare gli aspetti che possono colpire il consumatore medio quando acquista per la prima volta un

dispositivo sul quale gira Android.

L'interfaccia utente si basa sul concetto della *direct-manipulation* consentendo ingressi *single* e *multi-touch* come strisciate, tocchi e pizzichi sullo schermo per manipolarne gli oggetti visibili.

2.3 – Brian

Il lavoro di Alan Turing ebbe vasta influenza sullo sviluppo dell'informatica, grazie alla sua formalizzazione dei concetti di algoritmo e calcolo mediante la *macchina di Turing*, che a sua volta ha svolto un ruolo significativo nella creazione del moderno computer.

Per questi motivi, Alan Turing (matematico, logico e crittografo britannico) è considerato uno dei padri dell'informatica, uno dei più grandi matematici del XX secolo, nonché il padre dell'intelligenza artificiale.

Fu durante questo periodo che si ebbero i maggiori sviluppi di calcolatori che dovevano imitare il funzionamento della mente umana.

Alan Turing fu colui che ideò quello che fui poi denominato *Test di Turing.*

Il test di Turing è un criterio per determinare se una macchina sia in grado di pensare o meno. Tale criterio è stato da lui precisato nell'articolo *Computing machinery and* intelligence, apparso nel 1950 sulla rivista *Mind.*

Nell'articolo Turing prende spunto da un gioco (*il gioco dell'imitazione*) a tre partecipanti: un uomo *A*, una donna *B* e una terza persona *C*.

Quest'ultimo è tenuto separato dagli altri due e tramite una serie di domande deve stabilire qual è l'uomo e quale la donna. Dal canto loro anche A e B hanno dei compiti: A deve ingannare C e portarlo a fare un'identificazione errata, mentre B deve aiutarlo. Affinché C non possa disporre di alcun indizio, le sue risposte devono essere dattiloscritte.

Il test di basa sul presupposto che una macchina si sostituisca ad A. Se la percentuale di volte in cui C indovina chi sia l'uomo e chi la donna è simile prima e dopo la sostituzione di A con la macchina, allora la macchina stessa dovrebbe essere

considerata intelligente.

Brian, nome nato dall'anagramma della parola *brain* (cervello, in inglese), è un software che tenta di imitare al meglio la mente umana e dovrebbe idealmente superare il test di Turing con esito positivo.

L'app Brian è un'applicazione pensata per il totale controllo dello smartphone Android ed è così strutturata:

- Home Page
- Esempi
- Brian
- MiniBrian

L'A*ndroidManifest.xml* del progetto avvia il programma all'avvio stesso del sistema operativo grazie al BroadcastReceiver *Starter.class* e questo a sua volta chiama l'home page dal nome *Brian.class* che è strutturata in modo simile al frontespizio della tesi con l'aggiunta di animazioni di vario tipo (ovviamente non inseribili in formato cartaceo). Un

bottone ci porta nell'activity *Esempi.class* e uno avvia il programma portando in primo piano il file *ElencoAttività.class* che informa - evidentemente – sulle attività svolte da Brian per mezzo del Service *Dialogatore.class* che si occuperà di eseguire l'ascolto della voce e di rispondere – sempre a voce – senza cioè impegnare l'uso del display. Nell'eventualità in cui non fossimo interessati a visualizzare l'elenco delle attività perché dobbiamo svolgere altre funzioni nel telefono oppure perché non vogliamo tenerlo in mano, ci viene in aiuto il tasto *Minimizza* che chiude l'activity lasciando aperto *MiniBrian.class* che, sostanzialmente, visualizza sul display soltanto un bottone trasparente che ci informa sullo stato del servizio (facendo però tutto in *foreground)*. Quando siamo stufi di usare Brian (spero mai) possiamo tranquillamente terminarlo dal pulsante *Chiudi* presente su ElencoAttività.class o dicendo "termina".

Ci sono 2 stati possibili di chiusura:

- Chiusura totale

- Chiusura con avvisi attivi

La chiusura totale si occupa di terminare sia l'ascolto che la risposta.

La chiusura con avvisi attivi termina l'ascolto ma non la risposta così, nell'eventualità in cui abbiamo un appuntamento da non dimenticare, ma non vogliamo che Brian elabori sempre quello che diciamo, lo potremo fare.

Tutto questo meccanismo è possibile sfruttando due librerie che Java ci mette a disposizione *TextToSpeech* e *RecognitionListener*, rispettivamente per ricevere una risposta vocale e digitalizzare la voce sotto forma di un array di byte.

Brian.apk, il file di installazione di Brian, è costituito dal package *com.lec.brian* e contiene diverse classi e tra le principali troviamo *AvvisaAppuntamenti.class* (Service che gestisci il controllo degli appuntamenti per permettere di essere avvisati al momento corretto), *ElaboraScelta.class* (che si

occupa di esaminare ciò che Dialogatore.class ha ascoltato e scegliere la risposta giusta da dare), oltre ovviamente a quelle già precedentemente citate.

Bisogna però soffermarsi sul fatto che Brian è un software ottimizzato ma, anche in questo stato, è particolarmente dispendioso in termini di RAM e di batteria, cose purtroppo normali e ovviabili soltanto con l'uso di hardware potenti in quanto utilizza per tutto il suo periodo di attività il microfono per ascoltare quello che viene detto e, anche se si usa la modalità MiniBrian, tutte le sue funzioni vengono svolte in “silenzio”.

2.4– Codice

Per realizzare l'intero software si è fatto ricorso a 5 linguaggi di programmazione (Android, Wiring, HTML, PHP, SQL) che sono serviti per implementare i due client (cellulare o tablet e Arduino) e il server web.

Poiché sarebbe inverosimile inserire all'interno della tesi tutto il software, in quanto consiste di circa 10.000 righe di codice, verranno menzionati solo i pezzi che giocano un ruolo principale nella gestione dei relè.

Lato Android troviamo un frammento della funzione

private void cambiaStatoRelay(String risultatoEsaminato)

della classe *ElaboraScelta.*

Questa classe annidata si occupa di effettuare una chiamata asincrona al server per cambiare lo stato del relè desiderato

```
new AsyncTask<Integer, Boolean, Boolean>() {
        @Override
        protected Boolean doInBackground(Integer... params) {
        try {
                URL rintracciaUtenteURL = new URL(
                "http://esameluigi.altervista.org/Brian/WebServer.php?relay="
                + params[0] + "&stato=" + params[1]);
                BufferedReader br = new BufferedReader(
                        new InputStreamReader(
                        rintracciaUtenteURL.openStream()));

                String line;
                StringBuilder builder = new StringBuilder();
                while ((line = br.readLine()) != null)
                        builder.append(line);
                return true;
        } catch (Exception e) {
                e.printStackTrace();
                returnn false;
        }
};
        @Override
        protected void onPostExecute(Boolean result) {
        if (result == true)
                setTTSMessage("Cambio stato relè avvenuto.");
        else
                setTTSMessage("Non sono riuscito a cambiare lo stato del relè.");
        while (getContatore() > 1);
                parla(getTTSMessage());
        };
}.execute(relay, state);
```

Lato web troviamo il codice che interagisce con la funzione citata precedentemente.

In base al numero del relè scelto (che viene passato e recuperato tramite URL), si effettua la modifica del valore corrispondente.

```
if($relay)
{
	if($stato==-1)
	{
		$result = mysql_query("SELECT Stato FROM ArduinoWebServer
		WHERE relay='".$relay."'") or die (mysql_error());
		while($riga=mysql_fetch_array($result))
			if($riga['Stato']==0)
				$stato=1;
			else
				$stato=0;
	}
	$result = mysql_query("UPDATE ArduinoWebServer SET Stato='".$stato."'
	WHERE relay='".$relay."'") or die (mysql_error());
}
```

E infine una parte del codice di Arduino, quella che si occupa di impostare la connessione a Internet che servirà poi per recuperare ciclicamente il valore dello stato di ogni relè.

```
byte mac[]={0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
char server[] = {"http://www.esameluigi.altervista.org"};

EthernetClient client;

void setup()
{
	pinMode(4, OUTPUT);
	pinMode(10, OUTPUT);
	pinMode(11, OUTPUT);
	pinMode(12, OUTPUT);
	pinMode(13, OUTPUT);

	Serial.begin(9600);
	Serial.println("Inizzializzazione...");
	if(Ethernet.begin(mac)==0)
		Ethernet.begin(mac, ip, myDns, gateway, subnet);

	if (client.connect(server, 80)) {
		Serial.println("connected");
		client.println("GET /Brian/WebServer.php HTTP/1.0");
		client.println("Host: www.esameluigi.altervista.org");
		client.println("Connection: close");
		client.println();
	} else {
		Serial.println("connection failed");
	}
}

void loop()
{
	if (client.available()) {
		char c = client.read();
	}
}
```

CAPITOLO 3

L' Hardware

In ingegneria elettronica e informatica con il termine "*hardware*" si indica la parte fisica di un dispositivo (computer, smartphone, ecc.) ovvero tutte quelle parti elettroniche, elettriche, meccaniche, magnetiche, ottiche che ne consentono il funzionamento.

In generale si usa definire hardware un qualsiasi componente fisico di una periferica o di un'apparecchiatura elettronica.

Sebbene possa esistere hardware senza software, non è possibile trovare software senza hardware.

Di dispositivi hardware ne esistono davvero tanti, ogni giorno ne nascono di nuovi, e così i prezzi diminuiscono sempre più.

Alla base dell'hardware troviamo i circuiti integrati, *circuiti*

miniaturizzati dove i vari transistor sono stati formati tutti nello stesso istante grazie ad un unico processo fisico-chimico.

I circuiti integrati sono racchiusi all'interno dei *Chip* e grazie alla loro invenzione è nata la microelettronica. Essi possono assolvere a compiti specifici o generici.

Fra i tanti possibili circuiti integrati, è di particolare rilievo il ruolo che svolgono oggi i "microcontrollori". Si tratta di sistemi complessi che integrano in un unico chip una o più unità aritmetico-logiche, una o più aree di memoria e diverse periferiche di input-output. La particolarità più importante di questi dispositivi consiste nella loro programmabilità che li rende utilizzabili nelle situazioni più diverse senza modificare l'hardware ma semplicemente adattando il software.

In questo progetto verranno utilizzati dei microcontrollori della Atmel a cui verranno collegati dispositivi esterni di vario tipo come una scheda bluetooth (dispositivo di trasmissionedi dati), auricolare e microfono (dispositivi audio).

Per semplicità è stato scelto di usare delle schede già preassemblate della serie Arduino.

3.1 – Tecnologia di trasmissione

Fino a qualche anno fa, con l'invenzione dei telefoni fissi, la possibilità di comunicare avveniva grazie a reti cablate e droppini di rame.

Oggigiorno, grazie allo sviluppo della rete di telefonia mobile (wireless), quest'ultima è sempre più utilizzata a svantaggio della cosidetta rete cablata.

Bisognerebbe fare una grande discussione riguardante la natura fisica delle onde elettromagnetiche, tuttavia qui accenneremo soltanto una differenza sostanziale: a maggiore lunghezza d'onda corrisponde una minore frequenza e, viceversa, ad una minore lunghezza corrisponde una maggiore frequenza d'onda.

Facendo leva su questa proprietà, si possono distinguere diverse regioni dello spettro delle onde elettromagnetiche che, sia pur grossolanamente, possono essere identificate in onde

lunghe, medie e corte.

Le frequenze normalmente utilizzate per la comunicazione delle chiamate telefoniche, dell'invio degli SMS e per lo scambio di connessione dati, risiedono nella regione spettrale che va da 800 a 1800 MHz mentre per tecnologie che hanno bisogno di trasmissioni più veloci quali Wi-Fi e Bluetooth vengono utilizzate frequenza leggermente più elevate (2,45 Ghz).

Nel progetto che è stato sviluppato per questo lavoro di tesi viene utilizzata la tecnologia Bluetooth che è descritta nel seguito.

Il Bluetooth è uno standard tecnico-industriale di trasmissione dati per reti personali senza fili WPAN (*Wireless Personal Area Network*). Fornisce un metodo standard, economico e sicuro per scambiare informazioni tra dispositivi diversi attraverso una frequenza radio sicura a corto raggio.

Bluetooth cerca i dispositivi coperti dal segnale radio entro un

raggio di qualche decina di metri mettendoli in comunicazione tra loro. Questi dispositivi possono essere ad esempio palmari, telefoni cellulari, personal computer, portatili, stampanti, fotocamere digitali, console per videogiochi purché provvisti delle specifiche hardware e software richieste dallo standard stesso.

La specifica Bluetooth è stata sviluppata da Ericsson e in seguito formalizzata dalla SIG (Bluetooth Special Interest Group). La SIG, la cui costituzione è stata formalmente annunciata il 20 maggio 1999, è un'associazione formata da Sony Ericsson, IBM, Intel, Toshiba, Nokia e altre società che si sono aggiunte come associate o come membri aggiunti.

Il nome è ispirato a Harald Blåtand (*Harold Bluetooth,* in inglese), re Aroldo I di Danimarca (901 - 985 o 986), abile diplomatico che unì gli scandinavi introducendo nella regione il cristianesimo. Gli inventori della tecnologia devono aver ritenuto che fosse un nome adatto per un protocollo capace di

mettere in comunicazione dispositivi diversi (così come il re unì i popoli della penisola scandinava con la religione).

Il logo della tecnologia unisce infatti le rune nordiche *Hagall* e *Berkanan*, analoghe alle moderne H e B.

Questo standard è stato progettato con l'obiettivo primario di ottenere bassi consumi, un corto raggio d'azione (fino a 100 metri di copertura per un dispositivo di Classe 1 e fino ad un metro per dispositivi di Classe 3) e un basso costo di produzione per i dispositivi compatibili.

Attualmente più di un miliardo di dispositivi montano un'interfaccia Bluetooth.

I telefoni cellulari che integrano chip Bluetooth sono venduti in milioni di esemplari e sono abilitati a riconoscere e utilizzare periferiche Bluetooth in modo da svincolarsi dai classici cavi in rame.

Il protocollo Bluetooth lavora nelle frequenze libere di

2,45GHz.

Per ridurre le interferenze il protocollo divide la banda in 79 canali e provvede a commutare tra i vari canali 1.600 volte al secondo (*frequency hopping*).

La versione 1.1 e 1.2 del Bluetooth gestisce velocità di trasferimento fino a 723,1 kbit/s. La versione 2.0 gestisce una modalità ad alta velocità che consente un trasferimento dati fino a 3 Mbit/s. La versione 4.0 arriva ad una velocita di 24 mbps (3 MB al secondo). Questa modalità però aumenta la potenza assorbita. La nuova versione utilizza segnali più brevi, e quindi riesce a dimezzare la potenza richiesta rispetto al Bluetooth 1.2 (a parità di traffico inviato).

Bluetooth non è uno standard comparabile con il Wi-Fi, dato che questo è un protocollo nato per fornire elevate velocità di trasmissione con un raggio maggiore, a costo di una maggior potenza dissipata e di un hardware molto più costoso. Le due

tecnologie non sono tuttavia concorrenti: il Bluetooth crea una PAN (*Personal Area Network*) mentre il Wi-Fi crea una LAN (*Local Area Network*). Il Bluetooth può essere paragonato al bus USB, quindi ad un sistema di comunicazione fra periferiche, mentre il Wi-FI è un sistema di connessione a larga banda che realizza una rete locale ethernet non cablata.

Ad ogni modo lo standard Bluetooth include anche comunicazioni a lunga distanza tra dispositivi per realizzare delle LAN wireless. Ogni dispositivo Bluetooth è in grado di gestire simultaneamente la comunicazione con altri 7 dispositivi sebbene, essendo il collegamento di tipo master-slave, solo un dispositivo per volta può comunicare con il server. Questa rete minimale viene chiamata *piconet*. Le specifiche Bluetooth consentono di collegare due piconet in modo da espandere la rete. Tale rete viene chiamata *scatternet*. Ogni dispositivo Bluetooth è configurabile per cercare costantemente altri dispositivi e per collegarsi a questi. Può

essere impostata una password per motivi di sicurezza se lo si ritiene necessario.

BMW è stato il primo produttore di autoveicoli a integrare tecnologia Bluetooth nelle sue automobili in modo da consentire ai guidatori di rispondere al proprio telefono cellulare senza dover staccare le mani dal volante. Attualmente molti altri produttori di autoveicoli forniscono di serie o in opzione vivavoce Bluetooth che integrati con l'autoradio dell'automobile permettono di utilizzare il cellulare mantenendo le mani sul volante.

La limitazione della tecnologia Bluetooth è senza alcun dubbio il raggio di azione che, come già detto si limite ad alcuni metri.

Volendo estendere il raggio di azione è necessario fare ricorso alla tecnologia della rete WAN (Wide Area Network), con lo standard Ethernet.

Quest'ultima è in realtà una famiglia di tecnologie

standardizzate per reti locali che ne definisce le specifiche tecniche a livello fisico e a livello *MAC* del modello architetturale di rete *ISO-OSI*.

Il modello ISO-OSI, nato con il nome di *Open Systems Interconnection* è uno standard per reti di calcolatori stabilito nel 1978 dall'*International Organization for Standardization* (ISO) che stabilisce per l'architettura logica di rete una struttura a strati composta da una pila di protocolli di comunicazione di rete suddivisa in 7 livelli. Al fine di produrre una serie di standard per le reti di calcolatori, ISO avviò il progetto OSI, un modello standard di riferimento a formato aperto per l'interconnessione di sistemi di computer.

Ethernet è attualmente il sistema LAN più diffuso per diverse ragioni:

- è nata presto e si è diffusa velocemente
- è economica e facile da usare
- funziona bene e con pochi problemi
- è adeguato all'utilizzo con il protocollo TCP/IP

La codifica usata per i segnali binari è la codifica Manchester.
Ethernet è una tecnologia che fornisce al livello di rete un servizio senza connessione. Il mittente invia il frame nella LAN senza alcun *handshake* iniziale in modalità *broadcast* e viene ricevuto da tutti gli adattatori presenti ma è riconosciuto solo da quello che contiene la sua destinazione nel frame.
Il frame non è esente da errori spesso e viene verificato dal controllo *CRC* e, se non lo supera, viene scartato.
La velocità massima che di può teoricamente raggiungere con questa tecnologia è di 100 Mbit/s ma quando il cavo supera una

lunghezza di 100 m i segnali inviati cominciano a essere incoerenti e la trasmissione è impossibile. Solitamente si ovvia a questo problema con l'aggiunta di *hub* che permettono di amplificare il segnale in uscita.

3.2 – Arduino

Arduino è una scheda elettronica di piccole dimensioni con un microcontrollore e circuiteria di contorno, utile per creare rapidamente prototipi e per scopi hobbistici e didattici.

Il nome della scheda deriva da quello di un bar di Ivrea (che richiama a sua volta il nome di Arduino d'Ivrea, Re d'Italia nel 1002) frequentato da alcuni dei fondatori del progetto.

Con Arduino si possono realizzare in maniera relativamente rapida e semplice piccoli dispositivi come controllori di luci, di velocità per motori, sensori di luce, temperatura e umidità e molti altri progetti che utilizzano sensori, attuatori e comunicazione con altri dispositivi. È fornito di un semplice ambiente di sviluppo integrato per la programmazione. Tutto il software a corredo è libero, e gli schemi circuitali sono

distribuiti come hardware libero.

Si basa su un circuito stampato che integra un microcontrollore con pin connessi alle porte I/O, un regolatore di tensione e quando necessario un'interfaccia USB che permette la comunicazione con il computer. A questo hardware viene affiancato un ambiente di sviluppo integrato multipiattaforma (per Linux, Apple Macintosh e Windows). Questo software permette anche ai novizi di scrivere programmi con un linguaggio semplice e intuitivo derivato da C e C++ chiamato *Wiring*, liberamente scaricabile e modificabile.

Arduino può essere utilizzato per lo sviluppo di oggetti interattivi *stand-alone* e può anche interagire, tramite collegamento, con software residenti su computer, come Adobe Flash, Processing, Max/MSP, Pure Data, SuperCollider.

La piattaforma hardware Arduino è spesso distribuita agli hobbisti in versione pre-assemblata, acquistabile in internet o

in negozi specializzati. La particolarità del progetto è che le informazioni sull'hardware, e soprattutto i progetti, sono disponibili per chiunque: si tratta quindi di un hardware open source, distribuito nei termini della licenza Creative Commons Attribution-Share Alike 2.5. In questo modo, chi lo desidera può legalmente auto-costruirsi un clone di Arduino o derivarne una versione modificata, scaricando gratuitamente lo schema elettrico e l'elenco dei componenti elettronici necessari. Questa possibilità ha consentito lo sviluppo di prodotti Arduino compatibili da parte di piccole e medie aziende in tutto il mondo: è quindi divenuto possibile scegliere tra un'enorme quantità di schede Arduino-compatibili. Ciò che accomuna questi prodotti è il codice sorgente per l'ambiente di sviluppo integrato e le librerie residenti che sono resi disponibili, e concessi in uso, secondo i termini legali di una licenza libera, GPLv2.

Grazie alla base software comune ideata dai creatori del

progetto, per la comunità Arduino è stato possibile sviluppare programmi per connettere, a questo hardware, più o meno qualsiasi oggetto elettronico, computer, sensori, display o attuatori. Dopo anni di sperimentazione, è oggi possibile fruire di un database di informazioni vastissimo.

Il progetto “Arduino” prese avvio in Italia a Ivrea nel 2005, con lo scopo di rendere disponibile, a 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 una significativa riduzione dei costi rispetto ad altri prodotti disponibili sul mercato. A ottobre 2008, in tutto il mondo erano già stati venduti più di 50.000 esemplari di Arduino.

Una scheda Arduino tipica consiste in un microcontrollore a 8-bit AVR prodotto dalla 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 volt e un oscillatore a cristallo a 16 MHz, sebbene alcune implementazioni, come ad esempio la piccola *LilyPad*, abbiano un clock di 8 MHz e facciano a meno dello stabilizzatore di tensione.

Fino a oggi, sono state commercializzate 14 versioni dell'hardware Arduino:

- Serial Arduino, programmata con una porta seriale DB9.

Fa uso del microcontroller ATmega8.

- Arduino Extreme, con interfaccia di programmazione USB, facente uso del chip ATmega8.
- Arduino Mini, una versione in miniatura facente uso di un ATmega168 a montaggio superficiale.

- Arduino Nano, una versione ancor più piccola della Mini, utilizzante lo stesso controller ATmega168 SMD e alimentata tramite USB.

- LilyPad 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 NG, con un'interfaccia USB per programmare e usare un ATmega8.

- Arduino NG plus, con interfaccia di programmazione USB, con un ATmega168.

- Arduino BT, con interfaccia di programmazione Bluetooth e con un ATmega168.

- Arduino Diecimila, con interfaccia di programmazione USB e con un ATmega168 in un package DIL28.

- 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 Due, che fa uso di un Atmel SAM3X8E ARM Cortex-M3 CPU.

L'ambiente di sviluppo integrato (IDE) di Arduino è un'applicazione multipiattaforma scritta in Java, ed è derivata dall'IDE creato per il linguaggio di programmazione Processing e per il progetto Wiring. È concepita per iniziare alla programmazione artisti e altri neofiti, che siano a digiuno di pratica nello sviluppo di 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 in una sola passata e con un solo click. In genere non vi è bisogno di creare dei Makefile o far girare programmi dalla riga di comando.

Per il progetto della tesi verranno utilizzati l'Arduino Pro Mini per il bracciale (date le sue dimensioni molto ridotte) e l'Arduino UNO per la ciabatta elettrica (per la possibilità di inserire un maggior numero di collegamenti e non essendo

indespensabile una dimensione ridotta del dispositivo).

3.3 – Bracciale

La scomodità di Brian risiede nel fatto che per riuscire a elaborare correttamente la nostra voce, il dispositivo nel quale è in esecuzione deve avere il microfono posizionato in prossimità della sorgente sonora.
Realizzando un hardware comodo da portare che ingloba al suo interno il microfono e un auricolare avremo la possibilità di allontanare da noi il telefono e di riuscire tuttavia a comunicare con il software.

Per la costruzione del bracciale avremo bisogno dei seguenti elementi:

- Arduino Pro Mini
- BreadBoard Bluetooth
- BreadBoard USB
- Microfono
- Auricolare
- LED
- Batteria

Arduino Pro Mini è un micro-controllore con 14 *pin digitali* di input/output (di cui 6 possono essere *PWM* in output), 8 *pin analogici* di input e un pulsante di reset.

Ogni pin può fornire e ricevere un massimo di 40 mA e possiede ha una resistenza interna compresa tra i 20 e i 50 kOhms.

La versione che useremo noi avrà come voltaggio in ingresso 5V con una velocità di clock di 16MHz.

La *Flash Memory* ha una dimensione di 16 kb di cui 2 kb sono usati per il bootloader e una *EEPROM* di 512 byte.

Tutto questo lo ritroviamo in una board di 0,7" x 1,3".

Il bracciale sarà un semplice hardware comandato dal processore *Atmega168* dell'Arduino Pro Mini.

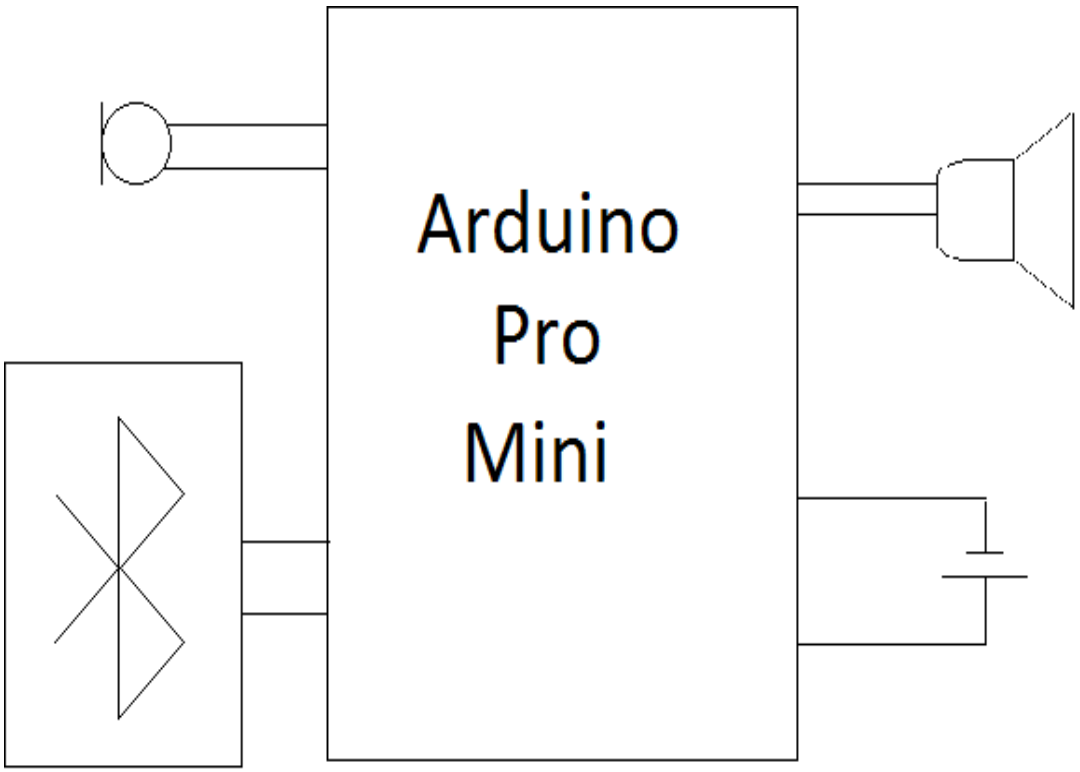
Al processore verrà inizialmente collegata la BreadBoard USB allo scopo di caricare in memoria il programma. Dopo aver finito questa prima fase, collegheremo la BreadBoard

Bluetooth per interfacciarsi con il Bluetooth del *device* nel quale è installato Brian, il microfono per poter fare captare il nostro messaggio vocale e l'auricolare per ascoltare la risposta del programma.

Un LED darà informazioni sullo stato (attivo – accesso, disattivo - spento) di Brian.

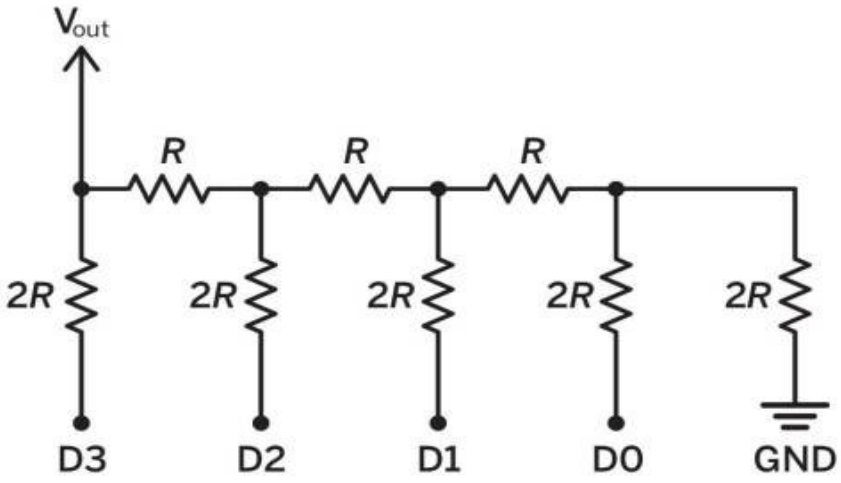
Ovviamente dovremo anche collegare una batteria da 5V per poter avviare il Pro Mini.

Il tutto viene schematizzato nel seguente schema a blocchi

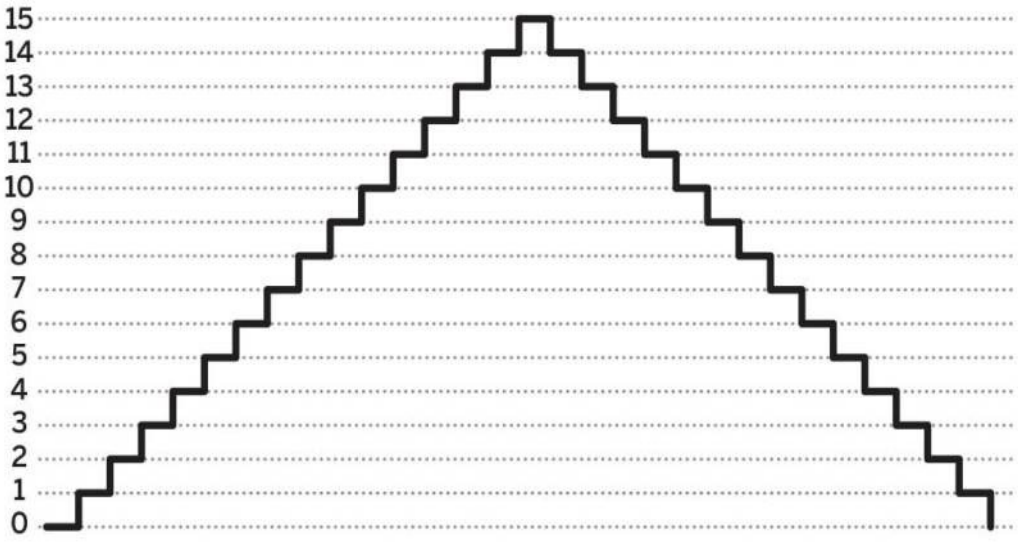


Purtroppo il progetto è troppo grande per la realizzazione di questo schema in quanto l'Arduino è sprovvisto di pin analogici in uscita; avremo dunque bisogno di convertire i segnali da digitali a analogici per mandare il segnale all'auricolare.

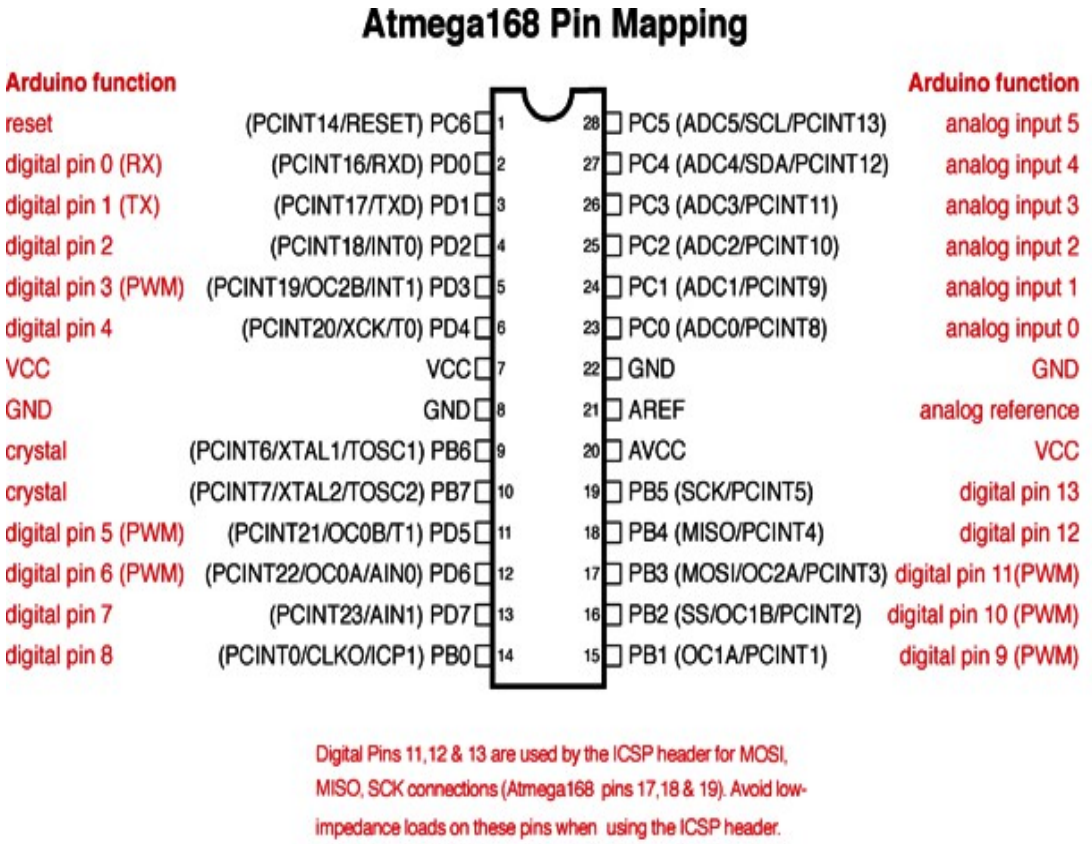
Avremo bisogno quindi di convertire i segnali audio in invio e ricezione negli 8 pin digitali a disposizione realizzando un semplice circuito *DAC* a 4 ingressi rappresentato dalla figura di seguito



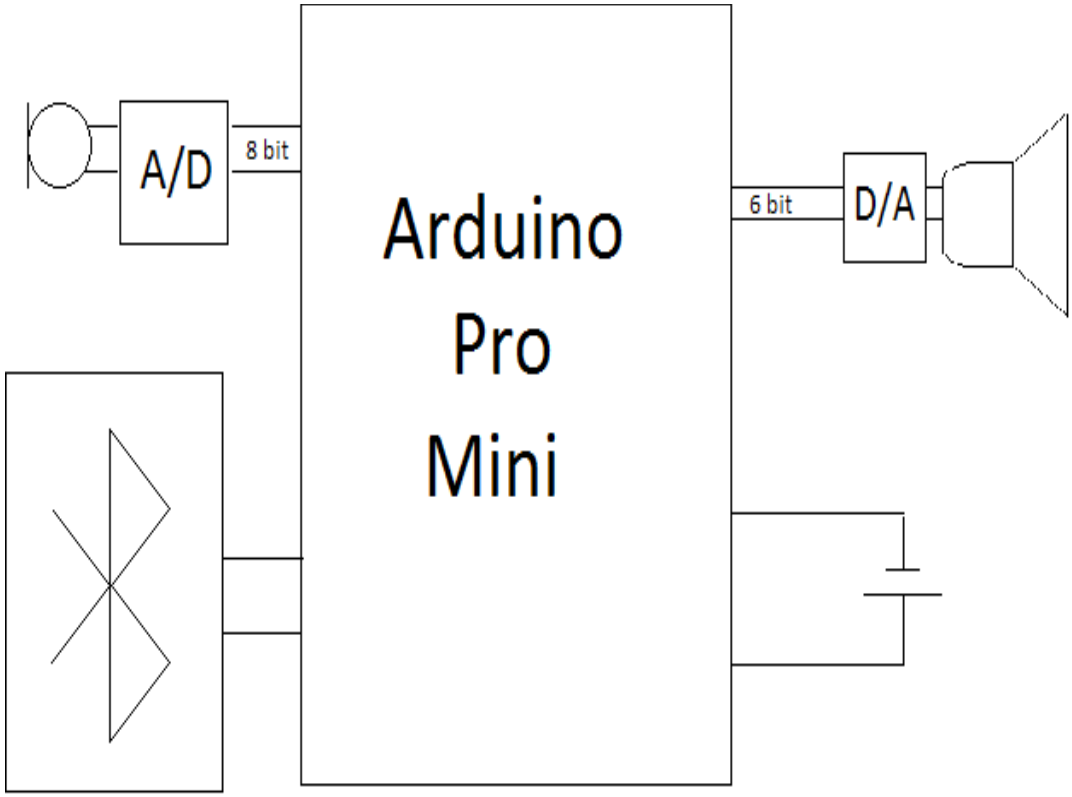
Questo circuito permettere di trasformare i segnali in ingresso in una discreta approssimazione di un'onda analogica come in questo schema



Qui di seguito è presente la mappatura dei pin del nostro processore.



Dopo aver aggiunto questi dettagli andremo a ridisegnare uno schema a blocchi più dettagliato del nostro bracciale



3.4 – Ciabatta elettrica

Una comune ciabatta elettrica è dotata di alcune prese nelle quali inserire i più svariati oggetti che abbiano bisogno di elettricità per funzionare come ad esempio TV, frigorifero, lavatrice, forno, macchinetta del caffè applicando una differenza di potenziale di 220-230 V.

Tuttavia, in questo modo, se vogliamo accendere o spegnere un apparecchio avremo bisogno di avvicinarci alla nostra ciabatta e attaccare o staccare la spina del dispositivo.

Senza alcun dubbio non è una funzionalità che vogliamo perdere, ma vogliamo aggiungerne un'altra, come l'attivazione e la disattivazione a distanza.

Per la costruzione della ciabatta elettrica avremo bisogno dei seguenti componenti:

- Arduino Uno
- Arduino Ethernet Shield
- Arduino Relè

Arduino UNO è dotato del processore Atmega328, un micro-controllore con 14 *pin digitali* di input/output (di cui 6 possono essere *PWM* in output), 6 *pin analogici* di input e un pulsante di reset.

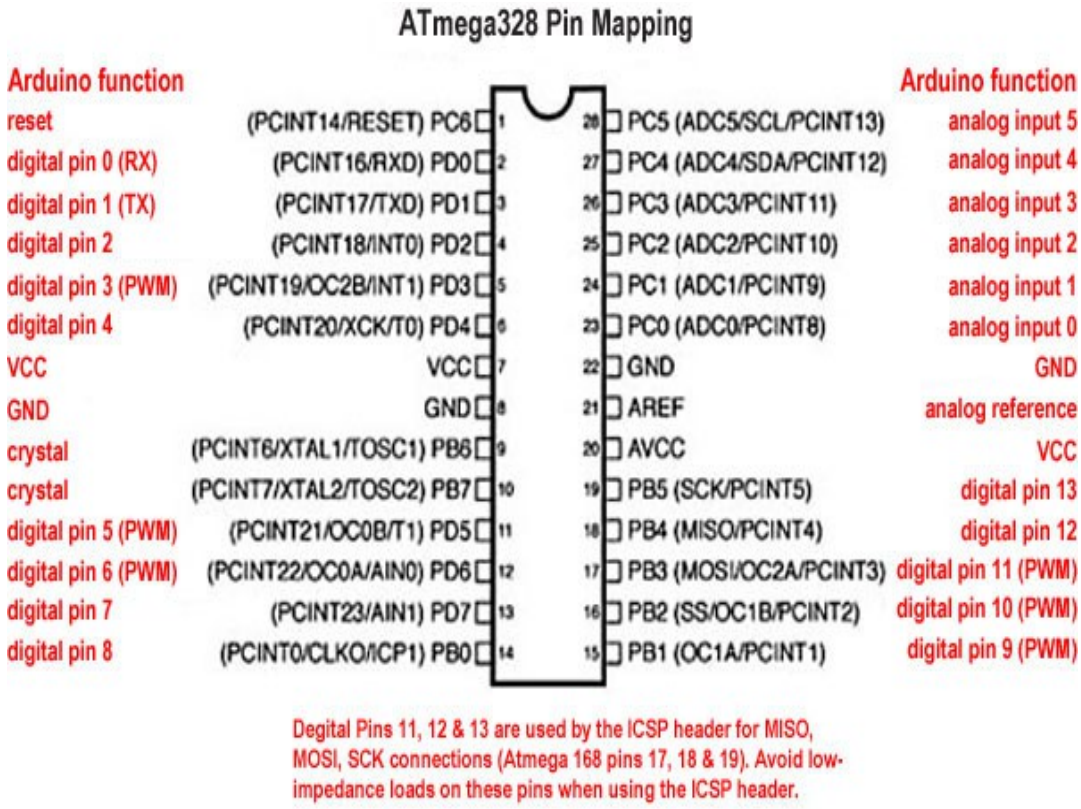
Ogni pin può fornire o ricevere un massimo di 40 mA ha una resistenza interna compresa tra i 20 e i 50 kOhm.

Il Voltaggio nominale in ingresso è di 7V-12V e funziona ad una velocità di clock di 16MHz.

La *Flash Memory* ha una dimensione di 32 kb di cui 0,5 kb sono usati per il *bootloader* e una *EEPROM* di 1kb.

Tutto questo lo ritroviamo in una board di 2,10" x 2,70".

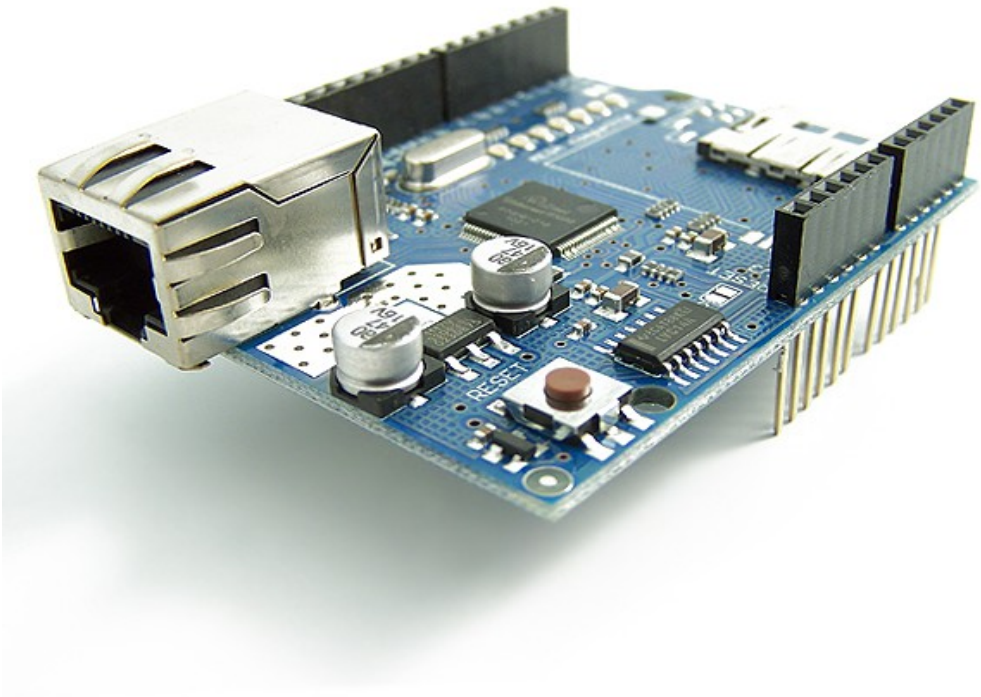
Qui di seguito è presente la mappatura dei pin del nostro processore.



La scelta di questa tipologia di Arduino è stata fatta perché essendo la ciabatta un progetto che non richiede piccole dimensioni, possiamo usare una board più estesa con un maggior numero di collegamenti quindi più facilmente gestibile.

Si era inizialmente pensato di adottare la comunicazione Bluetooth, così come per il bracciale, ma in questo caso ci potrebbe tornare più utile una comunicazione a lungo raggio e questo ha fatto cadere la scelta sull'uso della rete Internet.

Sarà la breadboard ethernet fornita da Arduino a permetterci di comunicare a distanza con il nostro dispositivo in quanto questa scheda dalle dimensioni ridotte è dotata di una porta ethernet con tecnologia POE (Power On Ethernet) ed è facilmente collegabile all'Arduino UNO e\ all'Arduino MEGA.



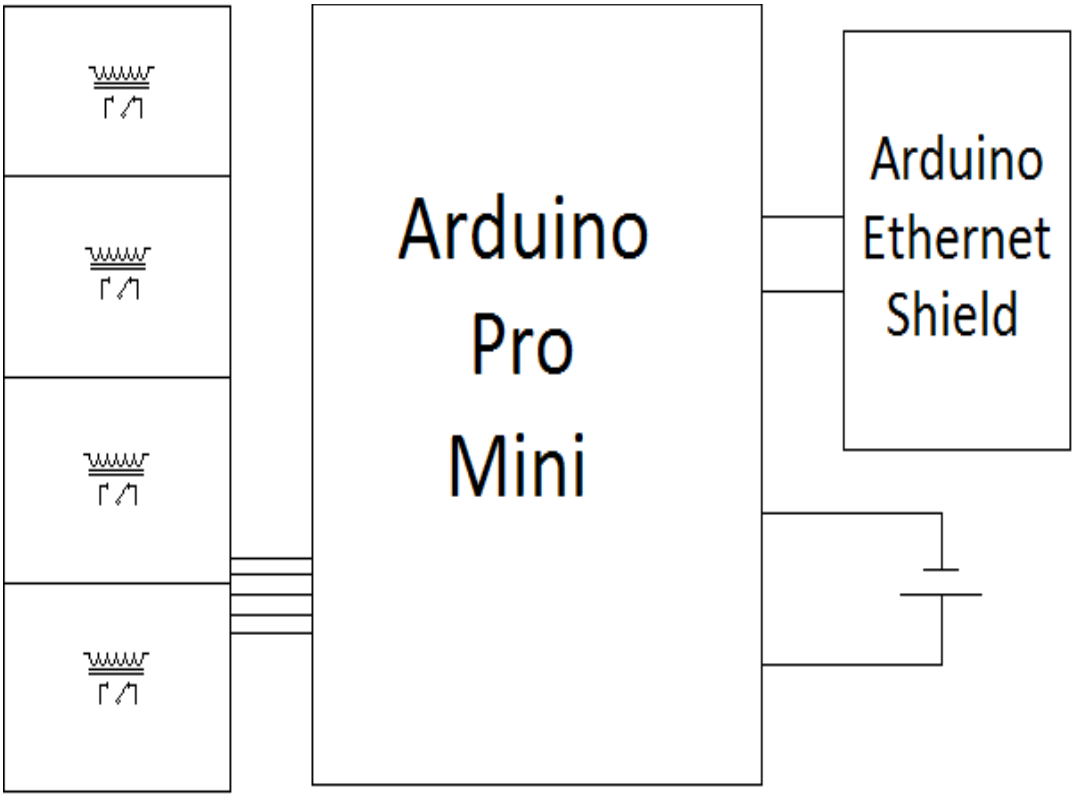
Infine collegheremo all'hardware anche una scheda di 8 relè fornita anch'essa da Arduino e ad ogni relè collegheremo un dispositivo.

Possiamo notare la presenza di 10 pin in input: alle estremità abbiamo i pin VCC e GND che si occuperanno dell'alimentazione, mentre i pin centrali servono a gestire i relè singolarmente.

Alla base di ogni relè è presente un piccolo led che si accende quando questi è attivo.

La tensione massima per ognuno varia in base alla scelta della corrente (alternata o continua), mentre l'intensità massima di corrente rimane sempre di 10A.

Il seguente schema illustra molto semplicemente il collegamento che viene realizzato per il progetto.



“Ma questo telefono può fare pure il caffè?”

Chi non ha mai sentito dire questa frase? Bene, con il collegamento Brian-Ciabatta sarà possibile farlo: se Brian sarà avvisato di accendere una determinata presa della ciabatta, manderà all'Arduino un byte con valore identificativo in modo unico, il processore Atmega328 elaborerà il messaggio ricevuto e farà passare corrente nel relè a cui è collegata la nostra caffettiera, il relè, eccitandosi, farà passare la corrente fino alla presa e il caffè sarà pronto in pochi minuti!

3.5 – Funzionamento finale

E' arriviamo così alla fase di test:

il software di Brian è in perenne ascolto di un qualsiasi messaggio; andiamo a vedere cosa succede quando gli si chiede di cambiare lo stato di un relè!

Il software Brian quando sente la parola "*relè*" cerca nella frase che ha sentito il numero corrispondente; se trova il numero effettua una connessione a internet verso un database di appoggio presente su altervista raggiungibile all'indirizzo:

http://esameluigi.altervista.org/Brian/WebServer.php

In base alla scelta richiesta, Brian si occuperà di inviare una richiesta ad una funzione PHP presente in quel sorgente che a sua volta permetterà di variare un byte della colonna *Stato* della tabella *ArduinoWebServer* impostandolo a *1* se bisogna

accendere il relè o a *0* se bisogna spegnerlo.

Fatto questo, l'Arduino UNO, il quale è connesso a Internet tramite la shield ethernet, che è in continuo ascolto di variazioni alla tabella, si accorgerà di uno stato diverso a quello che ha memorizzato e invierà il byte cambiato al relativo relè accendendolo o spegnendolo.

E' stato così realizzato un velocissimo e comodo sistema di domotica davvero molto utile!

Bibliografia

- *Earthshine Design Arduino Starters Kit Manual - A Complete Beginners Guide to the Arduino*

  ©2009 M.R.McRoberts Published 2009 by Earthshine Design. Design: Mike McRoberts First Edition - May 2009 Revision 1 - July 2009 Revisiion 2 - September 2009

- *Getting started with Arduino*

  Brad Kendall - August 2013

- *Realizzazione di un sistema di controllo degli accessi con Arduino*

  Agostini et al., TSRR vol. 3 (2011) 1-10

- *Sensoric subsystem of automated guided vehicle: TCP communication between SIMATIC S7 PLC and Arduino (2015)*

  Kajan, M., Sovcik, J., Duchon, F., Mrafko, L., Florek, M., Beno, P.

- *Design of small smart home system based on arduino (2015) Proceedings - 2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar, EECCIS 2014*

  Adriansyah, A., Dani, A.W.

- *Arduino implementation of automatic tuning in PID control of rotation in DC motors (2015) Lecture Notes in Electrical Engineering*

  Mendes Antunes, R.N., Ferreira, D.S., Gonçalves Santos, I.I., Das Neves Sousa, A.R., Augusto, J.S.

- *Design of microcontroller based multi-functional relay for automated protective system (2014) SCES 2014: Inspiring Engineering and Systems for Global Sustainability*

  Ramarao, G., Telagamsetti, S.K., Kale, V.S.

Fonti Online

- *www.wikipedia.it*
- *www.arduino.cc*
- *http://makezine.com/projects/make-35/advanced-arduino-sound-synthesis/*
- *http://developer.android.com/develop/index.html (Google)*