Tecniche Audiovisive: Sistema Speech-to-Text con Arduino e Python

Fabrizio Bonavita

Communications Engineering
mat: 1771782

25 marzo 2018

Sommario

Lo scopo del progetto é incentrato sulla realizzazione di un dispositivo in grado di riconoscere ed elaborare un pattern vocale per gestire l'attivazione di diversi sensori. Il dispositivo in questione converte il pattern vocale in una stringa e ricerca un matching tra un insieme di stringhe di default per eseguire una certa azione sui sensori collegati al dispositivo stesso.
*I sensori utlizzati sono un termometro digitale (*Texas Instruments LM35*) per effettuare misurazioni della temperatura in ambiente confinato e tre Led di colori differenti. Inoltre, si utilizza una scheda di prototipazione* Arduino Uno *per interfacciare il dispositivo tra i sensori e il pc. La parte software è del tutto implementata in Python tramite il quale si genera l'interfaccia grafica (GUI) con la libreria* Tkinter.
Il sistema "Speech-to-text" implementato si basa sulla libreria software "Google Cloud Speech API" che permette la conversione di audio in stringhe.

I. Introduzione

Questo progetto pone le basi per la realizzazione di un dispositivo intelligente in grado di assistere l'uomo nei compiti più semplici ed usuali di tutti i giorni.
Nel mondo dell'IoT è di vitale importanza progettare e trasformare gli oggetti di uso comune in macchine "intelligenti" che permettano una connessione profonda tra il dispositivo elettronico e l'utente. Proprio per questo, si sviluppa un dispositivo basato su un sistema "Speech-2-Text" in grado di tradurre il linguaggio umano in stringhe per permettere l'accensione o spegnimento di alcuni Led o la semplice acquisizione della temperatura in un ambiente. Il dispositivo realizzato è composto da un sensore di temperatura *Texas Instruments LM35* connesso insieme a tre LED di colori differenti alla scheda di pototipazione *Arduino Uno*. L'hardware descritto è stato programmato tramite python per creare un'interfaccia grafica con la libreria software in Python: *Tkinter*.
L'acquisizione del linguaggio e la conseguente traduzione in testo viene effettuata sfruttando le *Google Cloud Speech API*. Nelle sezioni seguenti viene introdotto il funzionamento delle varie parti hardware con i dettagli relativi alla progettazione e realizzazione del prototipo e in aggiunta viene descritto il codice sorgente nelle sue varie parti.

II. Progettazione Hardware

i. Cosè Arduino?

Arduino è una piattaforma "Open-Source" di sviluppo basata su un microcontrollore della ATMEL.
Ideata nel 2005 da un team italiano di Ivrea, Arduino nasce con lo scopo di definire un dispositivo hardware di controllo programmabile

che rappresentasse una valida alternativa ai più costosi oggetti di prototipazione.
Le caratteristche principali di Arduino sono:

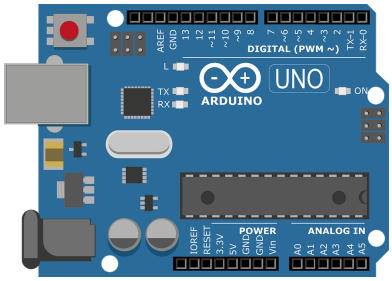

Figura 1: *Schema della scheda di prototipazione Arduino Uno*

- Interazione con il mondo circostante mediante I/O analogici e digitali
- Software e hardware "Open Source"
- Linguaggio di programmazione simile al "C" con una IDE molto intuitiva
- Possibiltà di comunicare dati in input e output con moltissimi altri linguaggi di programmazione per mezzo della comunicazione seriale (Matlab, C, C++, Java, Python, Processing)

Ogni scheda arduino è composta da un microcontrollore caratterizzato da una quantità eterogenea di PIN che permettono la comunicazione con diversi sensori in commercio.
I PIN si suddividono in analogici e digitali e la programmazione della scheda avviene in comunicazione seriale per mezzo della porta USB presente su ognuna di esse.

ii. Texas Instrument LM35

Il sensore di temperatura *Texas Instrument LM35* è un preciso circuito integrato con la particolarità di misurare un voltaggio in output linearmente proporzionale alla temperatura in gradi centrigradi (°C). Il dispositivo, infatti, non richiede nessuna calibrazione esterna per migliorare l'accuratezza della misura ed ha un range di lavoro che va da $-55\,°C$ a $150\,°C$.
Inoltre, la bassa impedenza in uscita, l'output lineare e la precisa calibrazione lo rendono un dispositivo molto semplice da utilizzare. La

Figura 2: *Sensore di temperatura Texas Instruments LM35*

conversione del valore di tensione in temperatura in gradi Celsius (°C) è caratterizzata dalla seguente formula:

$$T(°C) = tensione\,(mV) \cdot \frac{5000\,mV}{1023 \cdot 10\,(mV)}$$

dove 1023 sono i livelli di quantizzazione pari a $2^{10\,(bit)}$ in cui il valore 0 corrisponde a 0 V e il valore 1023 corrisponde a 5 V.
Come si nota in *Figura 3* il sensore è costituito da tre terminali: il primo per l'alimentazione con un range ammissibile dai 4 V ai 20 V, il secondo per la massa e il terzo per l'uscita della tensione proporzionale alla temperatura rilevata.

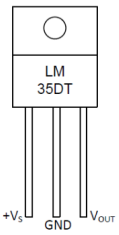

Figura 3: *I tre terminali del sensore di temperatura Texas Instruments LM35*

III. Progettazione Software

i. Google Cloud Speech API

In generale, le Application Programming Interface (API) formano un set di strumenti specifici per il completamento di un determinato compito all'interno di un programma.
Sviluppata ed utilizzata per diversi prodotti in commercio, *Google Cloud Speech API* è una libreria software che permette di convertire l'audio acquisito da un microfono in testo riconoscendo oltre 80 lingue e varianti.
La capacità di convertire un pattern vocale

in un testo si basa su efficaci modelli di *deep neural networks* in continuo apprendimento in modo tale che anche termini nuovi vengano assimilati ed inseriti nel sistema. Lo "Speech-to-text" è utilizzato in una moltitudine di casi che includono l'assistenza vocale intelligente su gli smartphone, la trascrizione audio e la domotica.

ii. Python

Python è un linguaggio di programmazione ad alto livello, orientato agli oggetti, adatto per molti tipi di sviluppo software come lo sviluppo di applicazioni distribuite, lo scripting e la computazione numerica.
Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche.

I principali vantaggi che lo hanno portato al successo è il forte supporto all'integrazione con altri linguaggi e programmi e la presenza di una estesa libreria standard in aggiunta alla facilità di apprendimento nell'utilizzo. Molti programmatori Python ritengono che il linguaggio incoraggi allo sviluppo di codice di qualità e manutenibilità superiori.

iii. GUI: Tkinter

Tkinter è la GUI di default di Python, Graphical User Interface (Interfaccia Grafica per l'Utente), ed è integrata in Python fin dalle sue origini. Ha notevoli doti di semplicità e portabilità, ottimi widget come canvas per la grafica strutturata, e text per l'editing avanzato del testo; ed è ben documentato sia in linea che in pubblicazioni cartacee. Sebbene non sia l'unico toolkit disponibile per le GUI in Python, è comunque il più utilizzato.
Come per ogni GUI, anche Tkinter ha il proprio insieme di termini da utilizzare come:

- Finestre: area dello schermo controllata da un programma
- Controllo: dispositivo di comando associato ad una finestra che può essere visibile (es. widget) o non visibile (es. timer)
- Widget: elemento di controllo, corrispondente ad un elemento visibile, che può essere manipolato dall'utente o dal programmatore (Label, Button, Frame, Text Entry, Text Box, Radio Button, Canvas, Image ecc.)
- Eventi: segnali di input generati da pressioni dei tasti della tastiera, pressioni dei tasti dei dispositivi di puntamento (mouse ed assimilati), attivazione dei timer.

IV. Realizzazione

i. Componenti hardware per la creazione del dispositivo

Come già sopradescritto, il dispositivo viene concepito come un sistema di acquisizione e conversione di pattern vocali per gestire diversi sensori collegati al dispositivo stesso con le seguenti caratteristiche:

- facilità di gestione e posizionamento della parte hardware
- struttura solida saldata all'interno di un box in plastica per evitare danneggiamenti da urti accidentali
- interfaccia grafica user-friendly
- gestione del dispositivo di tipo "safe" con inserimento di password vocale

Figura 4: *Foto del prototipo*

Il prototipo realizzato, per quanto riguarda la parte hardware, è composto dalla scheda *Arduino Uno* a cui sono collegati il sensore di temperatura *Texas Instruments LM35*, e tre Led di stato di colore rosso, giallo e verde.
L'intero dispositivo è inserito in un box in plastica per evitare danneggiamenti e allo stesso tempo per permettere un fissaggio delle varie componenti attaverso dei fori sul box. Co-

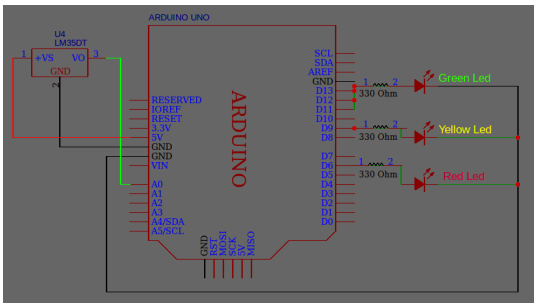

Figura 5: *Schema circuitale del dispositivo hardware*

me si nota nella *Figura 4* le connessioni tra le componenti e i *pin* di *Arduino Uno* sono le seguenti:

- *Texas Instruments LM35*: l'output per la lettura dei valori di tensione proporzionali alla temperatura sono effettuati connettendo il terminale V_{out} al pin analogico A_0 che acquisisce il segnale con velocità di circa 9000 $\frac{sample}{s}$. Inoltre è alimentato con 5 V e il terminle di massa è connesso al *pin* "GND".

- *Led:* i tre Led utilizzati di colore rosso, giallo e verde sono connessi rispettivamente ai *pin* digitali D_6, D_9, D_{11} e sono alimentati con una tensione costante generata con la *Pulse Width Modulation (PWM)*.
  Inoltre, per ogni led si interpone una resistenza di valore pari a 330 Ω e si chiude il circuito in un ulteriore punto di massa presente nella scheda *Arduino Uno*.

Infine, si utilizza come dispositivo esterno, ospitante il software sviluppato in Python, un laptop connesso alla base hardware tramite connessione seriale con un cavo USB.

ii. Interfaccia grafica

La *GUI* creata con Tkinter è stata sviluppata seguendo alcuni criteri di base come il facile utilizzo da parte dell'utente interessato, la lettura immediata della conversione del discorso in testo inviata direttamente su una label apposita, la lettura immediata dei risultati, la facile comprensione nell'acquisizione real time della temperatura e la gestione del dispositivo in modo "safe" attraverso l'inserimento di una password vocale ad ogni run del programma.

Figura 6: *GUI generata con Tkinter*

Come si nota dalla *Figura 6*, la *GUI* è composta da due oggetti principali: un *pushButton* e una *label*. Il primo è il tasto *REC* che, ad ogni pressione per mezzo del mouse, permette di attivare la fase di "Speech-to-text" dopo aver immesso la password che inizializza la "conversazione". Il secondo oggetto, in verde, è una semplice label "display" che guida l'utente in ogni passo da compiere facilitando la comprensione nella gestione del dispositivo.
Le frasi standard che si susseguono sul display sono:

1) *Rec and Speak:* invita a premere il tasto *REC* per avviare la conversazione
2) *Password:* al primo giro il dispositivo richiede la password per accertarsi che l'utente sia effettivamente il proprietario dell'oggetto; a questo punto si ripreme *REC* e in base alla password ripetuta dall'utente si verificano due possibilità:
   2.1) *Permission Accepted:* nel caso di password corretta
   2.2) *Permission Denied:* nel caso di password errata.
3) *Say Something:* nel caso di password corretta sul display appare questa stringa che invita a dichiarare l'azione che il dispositivo deve svolgere. Da questo momento ci sono altre due possibilità:

3.1) *You Said "...":* il dispositivo dichiara di aver capito una certa parola o frase e, nel caso corrisponda ad una possibile azione, svolge il compito ordinatogli e scrive sul display l'azione appena compiuta
3.2) *Could not understand audio:* il dispositivo dichiara di non aver compreso la frase o parola detta dall'utente.

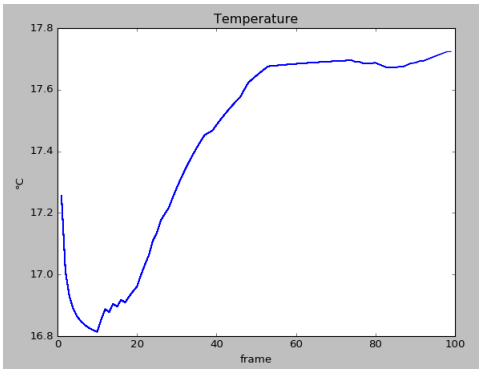

Figura 7: *Acquisizione real-time della temperatura con* Texas instruments LM35

Tra le varie funzioni possibili, si richiede anche un'acquisizione real-time della temperatura dell'ambiente nel quale è posizionato il dispositivo come si vede nella *Figura 7*.

V. Codice

Il Codice sorgente si può suddividere in due sezioni principali: nella prima si definiscono tutte le caratteristiche della *GUI* e si generano tutti gli oggetti necessari; nella seconda si implementano gli eventi principali che devono avvenire dopo la pressione del pushButton *REC*. Qui di seguito si riportano alcune delle parti principali del programma.
Per prima cosa si inseriscono le librerie necessarie al funzionamento di *Arduino*, *Tkinter* e per le *Google Cloud Speech API:*

```
from tkinter import *
from pyfirmata import Arduino, util
import numpy as np
import matplotlib.pyplot as plt
import time
import speech_recognition as sr
from tkinter.font import Font
```

Successivamente si genera l'interfaccia grafica definendo le variabili necessarie per lo svolgimento delle possbili azioni del dispositivo e si creano il *pushButton* e la *Label*:

```
self.mioGenitore = genitore
self.mioContenitore1 = Frame(genitore)
self.mioContenitore1.config(bg="black")
self.mioContenitore1.pack()
genitore.geometry("700x70")
```

La variabile *genitore* definisce l'intera intefaccia nella quale si inserisce il Frame *mioContenitore1* che a sua volta conterrà il pushButton e la Label:

```
#pulsante REC:
self.pulsante1 = Button(self.mioContenitore1)

#Label Display:
self.label1 = Label(self.mioContenitore1)
```

i quali conterrano tutti gli attributi necessari per la veste grafica (colore, font, testo, spessori e geometria e anche il comando da attivare alla pressione del tasto *REC* (*command=self.REC*).

i. PushButton *REC*

Dopo aver costruito l'interfaccia con i suoi oggetti, si passa all'implementazione degli eventi possibili inserendo la funzione *REC* che si suddivide in 3 sottoparti principali come descritto nella sezione *IV.ii "Interfaccia Grafica"*:

```
def REC(self):

    #passo 1) Password
    if self.flagPW==0:
     with sr.Microphone() as source:
            self.label1.config(text="Password:")
            radice.update_idletasks()
            audio = self.r.listen(source)

        try:
            self.label1.config(text=str("You said: "
```

```
            + self.r.recognize_google(audio)))
           radicе.update_idletasks()

     except sr.UnknownValueError:
         self.label1.config(text=
         "Could not understand audio")
         radice.update_idletasks()

     if str(self.r.recognize_
     google(audio))== "communications
    engineering":

         self.label1.config(text=
         "Permission Accepted")
         radice.update_idletasks()
         self.flagPW=1

     else:
         self.label1.config(text=
         "Permission Denied")
         radice.update_idletasks()
         self.flagPW=0
```

Nel primo passo si verifica solamente se l'utente è il proprietario del dispositivo e nel caso in cui la password sia corretta si passa al secondo step:

In base alla parola compresa dal dispositivo si esegue un'azione nel caso in cui la stringa sia presente tra i possibili eventi da svolgere. Si riportano degli esempi presenti nella funzione *REC*:

```
 #1) Accendo o spengo il led rosso
if str(self.r.recognize_google(audio))
=="red":

  self.label1.config(text=str("You said: "
   + self.r.recognize_google(audio)))
  radice.update_idletasks()
  if self.flagR==0:
     self.RL.write(1)
     self.label1.config(text="Red Led turned on")
     radice.update_idletasks()
     self.flagR=1
   else:
     self.RL.write(0)
     self.label1.config(text="Red Led turned off")
     radice.update_idletasks()
     self.flagR=0
```

```
else:
#passo 2) se riconosco la password inizio
ad ascoltare:


    with sr.Microphone() as source:
       self.label1.config(text=
             "Say Something")
       radice.update_idletasks()
       audio = self.r.listen(source)

    try:
       self.label1.config
       (text=str("You said: "
       + self.r.recognize_google(audio)))
       radice.update_idletasks()

     except sr.UnknownValueError:
       self.label1.config(text=
       "Could not understand audio")
       radice.update_idletasks()
```

```
#2) Acquizione e plot in real time
del valore di temperatura

if str(self.r.recognize_google(audio))
=='temperature on':

   self.label1.config(text="Real Time
     plot available")
   radice.update_idletasks()

   for i in range(100):
     iterator=util.Iterator(self.board)
     iterator.start()
     self.board.analog[0].enable_reporting()
     a= self.board.analog[0].read()
     self.volt[i]=a
     self.cels[i]=self.volt[i]*490.2

     if np.isnan(self.cels[i]) == 1:
        self.cels[i]=0
```

```
    time.sleep(0.1)

    if i>0:
       self.t.append(i)
       self.media = (self.media*(i-1)+
        self.cels[i])/i
       self.y.append(self.media)

      if self.cels[i]>22:
          plt.plot(self.t, self.y,color
            = 'red')
       else:
          plt.plot(self.t,self.y,color
           'blue')

       plt.autoscale()
       plt.xlabel("frame")
       plt.ylabel("°C")
       plt.title("Temperature")
       plt.pause(0.3)


#4) Converto i gradi Celsius (°C)
 in Fahrenheit (°F)

if str(self.r.recognize_google(audio))
=="conversion":
   self.mediaF =
   np.round(self.mediaC*1.8+32,2)
   self.label1.config(text=
   ("It's: "+ str(self.mediaF)+"°F"))
```

Qui di sopra sono riportati quattro esempi di come il dispositivo cerca il match con una certa stringa e in caso di matching positivo svolge le operazioni interne. Le stringhe di base che vengono utilizzate sono:

- *red*: accende o spegne il Led rosso
- *yellow*: accende o spegne il Led giallo
- *green*: accende o spegne il Led verde
- *turn on all*: accende contemporaneamente tutti e tre i Led
- *turn off all*: spegne contemporaneamente tutti e tre i Led
- *disco:* gioco di luci in cui i Led si spengono e accendono in modo alternato ed in sequenza per 10 iterazioni
- *temperature on*: acquisizione real-time della temperatura nell'ambiente per mezzo del termometro. L'acquisizione dura 100 iterazioni e ed ha un passo di 0.3 s tra un valore e il successivo.
- *what temperature is it*: ritorna sul display Label la temperatura della stanza in gradi Celsius (°C)
- *conversion*: converte la temperatura in gradi Celsius in grafi Fahrenheit (°F)

VI. Conclusioni

Il dispositivo realizzato illustra in modo evidente l'importanza di ampliare il progetto trasversalmente per ottenere un'interfaccia uomo-macchina che svolga i compiti di tutti i giorni nelle nostre case. Il sistema in questione può essere adottato in tutte le tecnologie relative alla domotica per migliorare l'efficacia nello svolgimento delle azioni e per creare un "compagno" intelligente che si occupi della gestione della propria casa. Nei lavori futuri è necessario definire una connessione wireless tra la base hardware e quella software per elevare la libertà di utilizzo del dispositivo. A tal proposito, si potrebbe sviluppare un'applicazione per sistemi Android e IoS, in modo da ordinare le azioni da compiere a distanza attraverso il microfono dello Smartphone.