Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Progetto con Raspberry

Studente/i

Allegra Nicol
Ambrosio Luca

Codice progetto:

Paganini

Niccolò Paganini è stato un violinista, compositore e chitarrista italiano, fra i più importanti esponenti della musica romantica. Celebre fu la sua frase detta al Re che richiedeva un bis del concerto appena svolto: “Paganini non ripete”.

Corso di laurea

Ingegneria Informatica

Modulo

Telamatica

Anno

2017/2018

13/05/2018

Data

Indice generale

Sommario

Descrizione progetto

1 Desk notifier

1.1 Idea iniziale

La prima idea che ci era venuta in mente da realizzare con l'utilizzo del raspberry era creare un desk notifier che indicasse il numero di notifiche dei vari social network o caselle di posta. (Figura 1)

.

Figura 1: Desk Notifier

Per implementare questa versione erano richiesti diversi componenti hardware e conoscenze elettroniche.
In quanto il tempo era abbastanza ridotto abbiamo optato e scelto una versione alternativa impiegando meno risorse ma arrivando allo stesso scopo, anche aggiungendo delle funzionalità che un semplice contatore non era in grado di svolgere.

1.2 Idea finale

Prendendo idee e spunti dall'epoca attuale e dalle diverse tecnologie già in uso, abbiamo deciso di utilizzare il raspberry per creare un notificatore vocale, cioè per ogni notifica sui social o posta elettronica ti avvisa dell'arrivo e ne legge il contenuto (che tipo di notifica o il messaggio della mail).

Il nostro progetto terminale vede come protagonisti le sole notifiche di Facebook, leggendone il contenuto, e le nuove mail in arrivo dalla casella di posta di un account Gmail.

1.2.1 Hardware utilizzato

Per l'implementazione non è stato necessario un grande quantitative di hardware.
-Raspberry Pi (qualsiasi versione)
-Micro sd 8Gb
-Cavo alimentazione
-Casse audio con jack 3.5mm

Figura 2: Raspberry

1.2.2 Software utilizzato

Come software abbiamo utilizzato il Sistema operativo Raspian, che era il più utilizzato per il raspberry ed era ottimo per ciò che volevamo implementare.
Inoltre il linguaggio che abbiamo utilizzato è stato Python quindi abbiamo dovuto scaricare le diverse librerie per il suo utilizzo.
Altre librerie e software sono stati necessari per altre interazioni.
Di seguito i comandi amministratore per le librerie sopracitate:

sudo apt-get install python-dev

sudo apt-get install python-pip

sudo pip install feedparser

sudo easy_install -U distribute

sudo apt-get install python-rpi.gpio

sudo pip install urllib2

1.2.3 Descrizione interazione con le API

Come già anticipato, dati anche i limiti di tempo, abbiamo deciso di concentrarci sull'utilizzo di Gmail e Facebook, non escludendo la possibilità di poter lavorare anche su altre caselle di posta e su altri social network.

L'implementazione (vedi capitolo 2) vede soprattutto la comunicazione con le API dei fruitori del servizio:
-Attraverso il nostro raspberry inviamo delle richieste,
-Le API ci inviano le informazioni che abbiamo richiesto,
-Il raspberry le elabora e le trasforma in audio.
(Figura 3)

Questo progetto si basa soprattutto sul Networking, necessitiamo della connessione alla rete in quanto il nostro raspberry chiede in continuazione determinate informazioni che viaggiano costantemente sul canale; l'aggiornamento delle notifiche avviene dunque in real-time con ovviamente qualche ritardo dovuto ai molti fattori di disturbo.

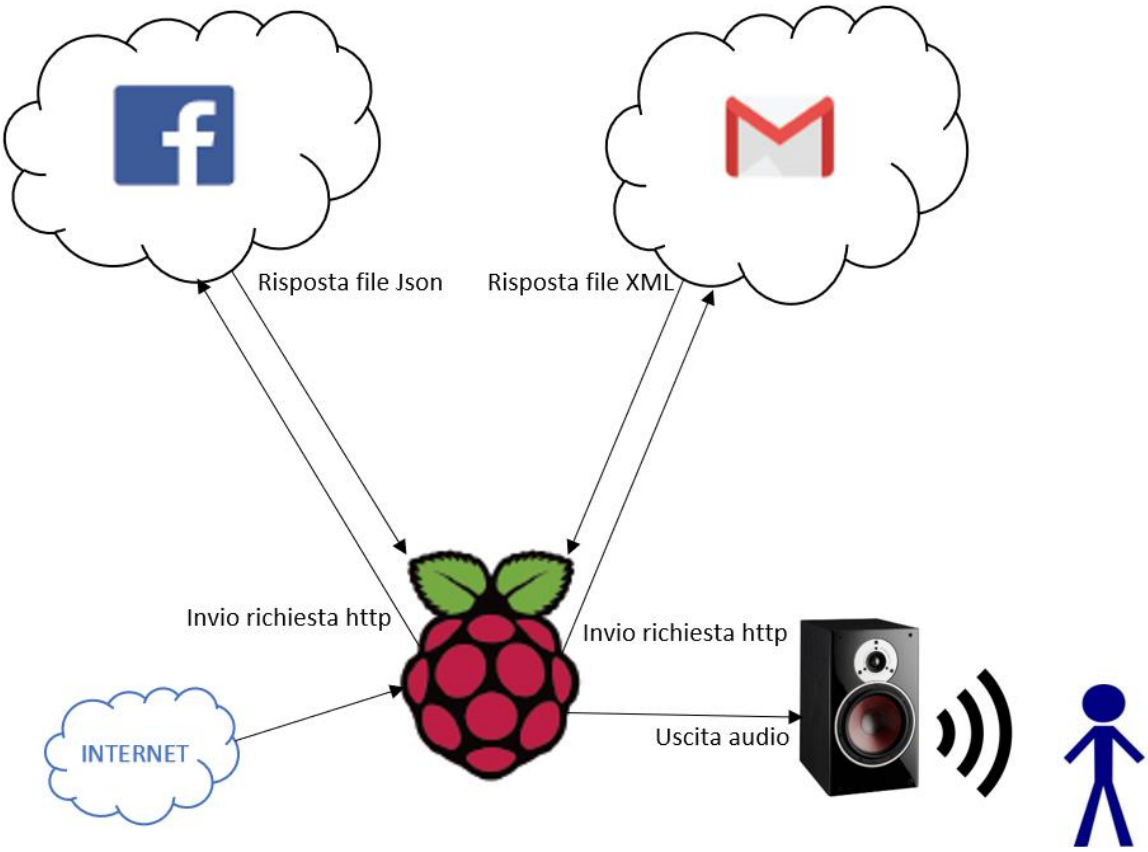


Figura 3: Schema Client-Server

Implementazione

2 Dettagli implementazione

2.1 Gmail

2.1.1 Connessione alle API con script Python

Di seguito si trova la script che abbiamo scritto e utilizzato per connetterci e richiedere informazioni alle API di Gmail.

```
import RPi.GPIO as GPIO, feedparser, time
import os
DEBUG = 1

USERNAME = "lucanicoltlm"     # just the part before the @ sign, add yours here
PASSWORD = "Ciao1234"     # password of your email
prec=0
MAIL_CHECK_FREQ = 1      # check mail every 60 seconds

while True:

    newmails = feedparser.parse("https://" + USERNAME + ":" + PASSWORD +"@mail.google.com/gmail/feed/atom")["feed"]["fullcount"]

    if (prec<newmails):
        bash='flite -voice slt -t "Have you "'+ newmails +'" new mail from Gmail"'
        os.system(bash)
        prec=newmails;

    time.sleep(MAIL_CHECK_FREQ)
```

Come è possibile intuire nelle variabili USERNAME e PASSWORD si devono inserire le credenziali di accesso al proprio account Gmail.

La variabile MAIL_CHECK_FREQ viene utilizzata nella funzione time.sleep e sta quindi ad indicare con quale frequenza il nostro raspberry chiede informazioni all' API, nel nostro esempio ogni secondo.

All'interno del while True, fondamentale è la variabile newmails dove vengono salvato il numero di mail presenti nella casella di posta in arrivo.

Questo numero di ottiene attraverso il parse del file Xml ritornato dall'interrogazione dell'API, accedendo al ramo fullcount che, come vedremo in seguito, contiene esattamente il valore che ci interessa.

L'if è necessario per far "parlare" il raspberry solo nel caso ci siano effettivamente nuove mail nella casella di posta.

Il comando utilizzato nella variabile bash lo discuteremo in seguito (Capitolo 2.3)

2.1.2 File Xml di risposta


```
▼<feed xmlns="http://purl.org/atom/ns#" version="0.3">
   <title>Gmail - Inbox for lucanicoltlm@gmail.com</title>
   <tagline>New messages in your Gmail Inbox</tagline>
   <fullcount>1</fullcount>
   <link rel="alternate" href="https://mail.google.com/mail" type="text/html"/>
   <modified>2018-05-12T16:01:09Z</modified>
  ▼<entry>
     <title>Prova</title>
     <summary>Ciao a tutti</summary>
     <link rel="alternate" href="https://mail.google.com/mail?account_id=lucanicoltlm@gmail.com&message_id=163551574e6ad551&view=conv&extsrc=atom" type="text/html"
     <modified>2018-05-12T16:00:59Z</modified>
     <issued>2018-05-12T16:00:59Z</issued>
     <id>tag:gmail.google.com,2004:1600274678019708241</id>
    ▼<author>
       <name>Nicol Allegra</name>
       <email>niclol.allegra@gmail.com</email>
     </author>
   </entry>
 </feed>
```


Questo è il file Xml che ci rende disponibile l'API come risposta alla nostra richiesta, oltre al ramo fullcount che è quello che utilizziamo noi, si possono notare molte altre informazioni che potrebbero essere utili e utilizzabili per sviluppi futuri come ad esempio:

-Title che rappresenta l'oggetto
-Summary che contiene il testo, quindi il corpo della mail
-Name contiene il nome del mittente
-Email indirizzo del mittente

2.2 Facebook

2.2.1 Connessioni alle API con script Python

```
import urllib2
import json
import time
import os
precv=0
precf=0

def get_page_data(page_id,access_token):
    api_endpoint = "https://graph.facebook.com/v3.0/"
    fb_graph_url = api_endpoint+page_id+"?fields=id,name,fan_count,unread_notif_count,notifications{title},link&access_token="+access_token
    try:
        api_request = urllib2.Request(fb_graph_url)
        api_response = urllib2.urlopen(api_request)

        try:
            return json.loads(api_response.read())
        except (ValueError, KeyError, TypeError):
            return "JSON error"

    except IOError, e:
        if hasattr(e, 'code'):
            return e.code
        elif hasattr(e, 'reason'):
            return e.reason

while 1:
    page_id = "825164851011192" # username or id
    token = "EAACEdEose0cBAME9ZCWsp2yNpAD2i5uJ3EpFSWz3fdGtCLYg7BsPtzKVEiyeJZCrrQ6v7D2bqRFW2l4wWJSy4gJaKdZB6wiZACq2OAm67ALzfwOnCVZB75fkdf9uhOqZ
    page_data = get_page_data(page_id,token)
    fan=page_data['fan_count']
    notif=page_data['unread_notif_count']
    if(precf<fan  or precv<notif):
        bash='flite -voice slt -t "Page name: '+ page_data['name'] + '"'
        os.system(bash)
        c=page_data['notifications']['data'][0]['title'].encode('utf-8').strip()
        bash='flite -voice slt -t "New notification : '+ c+ '"'
        os.system(bash)
        bash='flite -voice slt -t "Followers : '+ str(page_data['fan_count'] )+ '"'
        os.system(bash)
        bash='flite -voice slt -t "Notifications : '+ str(page_data['unread_notif_count']) + '"'
        os.system(bash)
        precf=fan
        precv=notif
    time.sleep(0.5)
```

Questo è lo script finale che ci permette di interrogare l'API di Facebook e di interpretare il file Json di risposta.

Per testare il funzionamento abbiamo preferito creare un nuovo account a cui abbiamo associato la pagina "Luca Ambrosio" per poter eseguire i vari test in totale libertà.

La funzione get_page_data è il corpo di tutto lo script in quanto va a richiedere al tool di esplorazione le informazioni che ci interessano, importante come la riga di codice che viene scritta nella variabile fb_graph_url dev'essere uguale alla richiesta che viene fatta nel sito dell'API di Facebook, come vedremo in seguito.

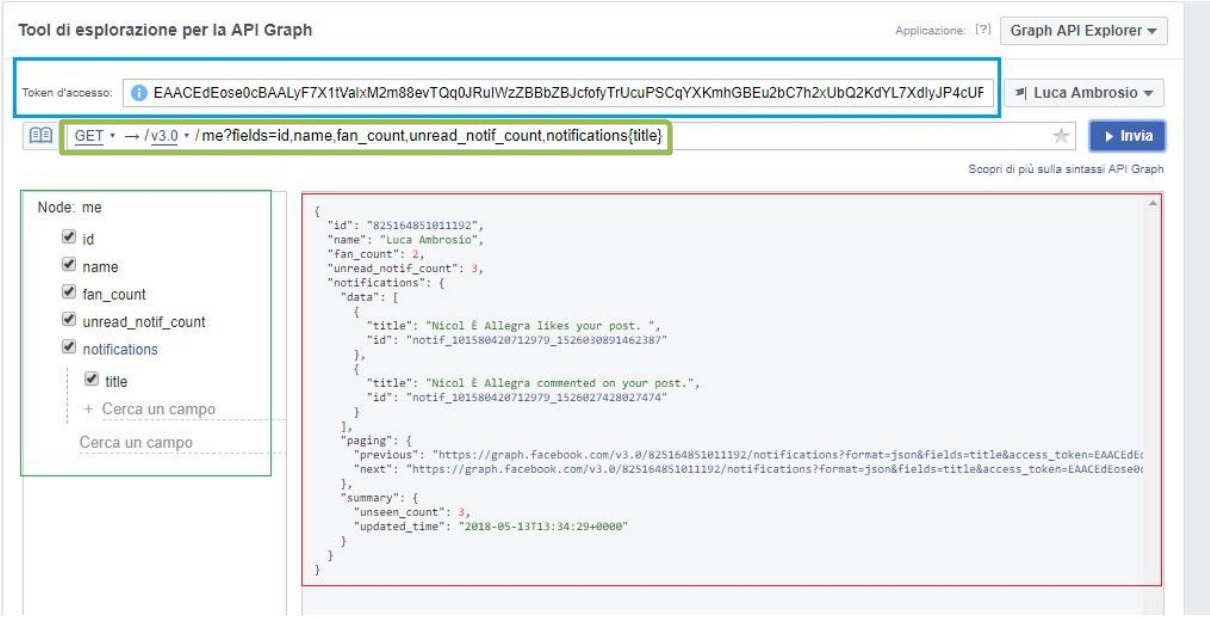
All'interno del while 1 entriamo più nel dettaglio dell'implementazione del nostro progetto, in quanto andiamo a estrapolare dal file Json le nostre informazioni che andiamo a elaborare.

Anche in questo caso abbiamo un if che esegue il controllo sulle variabili che contengono il numero di notifiche, per far "parlare" il raspberry solo nel caso in cui avvenga un aggiornamento.

Una riga di codice molto significativa è quella che andiamo a salvare all'interno della variabile c; questo ramificazione è necessaria in quanto in notifications è un "vettore" che contiene diversi 'title', che sono il corpo della notifica e per accedervi singolarmente è necessario quel tipo di metodo.

Anche in questo script facciamo uso del comando flite per permettere la sintetizzazione vocale.

2.2.2 File Json di risposta



Ora possiamo capire più nel dettaglio come e perché lo script python è stato sviluppato.

Per prima cosa siamo andati sul sito che Facebook mette a disposizione per gli sviluppatori (https://developers.facebook.com/tools/explorer), entrando con il nostro account abbiamo selezionato la pagina da cui volevamo estrapolare le diverse notifiche.

Dentro il riquadro GET-> bisogna inserire i campi di cui si vogliono richiedere le informazioni, nel nostro esempio abbiamo deciso di richiedere:
-l'id, necessario per accedere all'API e indispensabile nello script
-name, nome della pagina
-fan_count, numero di followers
-unread_notif_count, contatore di notifiche non ancora visualizzate
-notifications(title), che ti restituisce il corpo della notifica

Come abbiamo già anticipato quello che viene inserito all'interno di questo riquadro dev'essere identico a quello che viene salvato nella variabile fb_graph_url; da notare come i campi scelti vengano visualizzati nella parte sinistra del sito (vedi riquadri verdi)

Una volta premuto il tasto 'Invia' viene creato un Token d'accesso (riquadro azzurro), anche questo indispensabile e riportato all'interno dello script, e cosa più importante il file Json (riquadro rosso) su cui si dovrà andare a lavorare.

Come si può notare dall'esempio riporta tutte le informazioni che gli abbiamo richiesto, sta a noi saperle gestire.
All'interno di title sono presenti le notifiche che vengono inserite con metodo FIFO, dunque la notifica più recente è sempre la prima, per questo per far dire la notifica appena arrivata bisogna accedere nella posizione 0.

2.3 Comando Flite

Per permettere di creare e sintetizzare l'audio abbiamo utilizzato il comando di bash 'flite' per questo abbiamo dovuto anche studiare il modo di utilizzare un comando bash all'interno di uno script in Python.

Andando più nel dettaglio prendendo come esempio la seguente riga di codice

```
bash='flite -voice slt -t "You have "'+ newmails +'" mails from Gmail"'
```

salviamo in una variabile il comando come lo dovremmo scrivere in bash, questo comando è formato dalla parola chiave flite e dai suoi parametri:
-voice slt, utilizzato per "cambiare" la voce computerizzata
-t per indicare che deve "dire" quello che sta scritto all'interno delle virgolette, quindi prendendolo come formato testuale.

Importantissimo è l'utilizzo delle virgolette in quanto anche tutto il comando dev'essere racchiuso tra virgolette (vedi esempio sopracitato),
inoltre bisogna fare attenzione anche quando vogliamo concatenare delle variabili che fanno parte dello script.

Infine per utilizzare realmente in bash il nostro comando dobbiamo importare la libreria
Os e da questa libreria utilizzare il metodo os.system() passandogli come parametro la nostra variabile in cui abbiamo salvato il comando.

Nell'esempio precedente sarà: os.system(bash).

Infine facendo eseguire lo script il nostro audio dirà: " You have TWO mails from Gmail"

2.4 Problemi riscontrati

I problemi durante lo sviluppo non sono stati pochi, per prima cosa è stato impegnativo capire come richiedere interrogare la API e soprattutto interpretare la risposta che ci veniva fornita.
Ad esempio con il file di risposta Xml (fatto da Gmail) non siamo stati in grado di accedere ai rami più indentati perché bisogna per l'appunto attraversa più livelli e di conseguenza non ci è stato possibile estrapolare le informazioni per quanto riguarda il testo della mail ricevute.

Anche capire come utilizzare il sito di Facebook per creare il file Json che abbiamo utilizzato non è stato semplice ma alla fine siamo riusciti a interpretarlo e a farci fornire le informazioni che desideravamo.

L'ostacolo più grande da superare è stato l'utilizzo di python in quanto è un linguaggio che non abbiamo mai visto né utilizzato e che presenta molte insidie per quanto riguarda la programmazione.

In ultimo abbiamo dovuto cercare e scegliere il comando più adatto per sintetizzare l'audio e infine trovado flite abbiamo studiato le piccole sfacettature.

Test e demo

3 Presentazione

Abbiamo fatto diversi test per entrambi gli script, c'erano sempre alcuni problemi ma siamo riusciti a risolverli e a ottenere una versione finale completamente funzionante.

La demo è stata presentata, ottenendo ottimi risultati e valutazioni positive, il giorno 11 Maggio 2018 davanti alla classe.

Possiamo pensare a migliorie quali aggiungere la lettura del testo dalle mail e inoltre è possibile implementare ulteriori interfacciamenti con le API.

Un altro possibile accorgimento è quello di creare un unico script che congiunga i due facendo quindi partire in automatico e in parallelo le domande alle API.

Linkografia

[1] http://www.instructables.com/id/Raspberry-Pi-Desk-Notifier/

[2] http://www.festvox.org/flite/doc/flite_7.html