Arduino
Tutorial #21 - Arduino Jukebox
Materiale utilizzato – Arduino Uno, NFC Shield, 3 NGC Tag

Per vedere tutti i tutorial: www.marcopucci.it/arduino

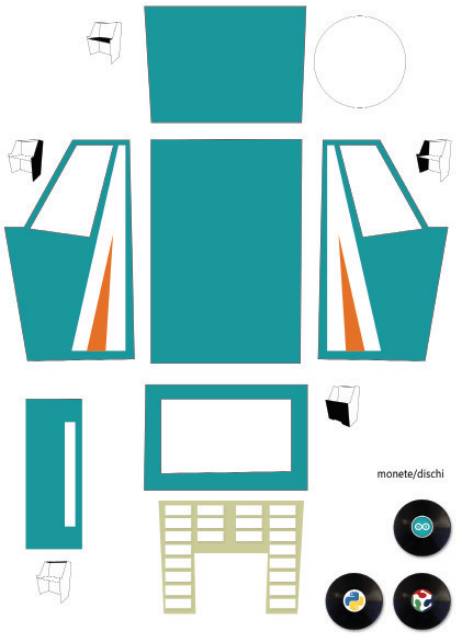
Questo tutorial è stato realizzato in collaborazione con www.robotics-3d.com.
Se volete acquistare il kit per realizzare il tutorial cliccate qui.
Se avete già Arduino potete acquistare i singoli componenti in questi due link:
NFC Shield V2 - clicca qui
NFC Tag - clicca qui

In questo tutorial realizziamo un Jukebox con Arduino.

Di seguito tutte le istruzioni per montare il jukebox e programmarlo.
Come nel precedente tutorial utilizziamo la tecnologia nfc (near field communication) per leggere il valore della moneta inserita. Se questa moneta (con dietro incollato un tag NFC) contiene una messaggio uguale a "rock", Arduino invierà un messaggio a Phtyon che selezionerà dal computer una canzone "rock".

Prima di continuare vi invito a leggere il tutorial precedente
ARDUINO NFC: NEAR FIELD COMMUNICATION dove è spiegato tutto il codice di Arduino.
Per quanto riguarda la programmazione di Python, questa verrà descritta in fondo al tutorial. Se è la prima volta che incontrate questo linguaggio di programmazione vi invito a leggere il tutorial ARDUINO E PYTHON per scoprire come si installa e come si utilizza questo programma open source che permette ad Arduino di gestire operazioni complesse come apertura di video, gestione della musica, pagine web, ecc...

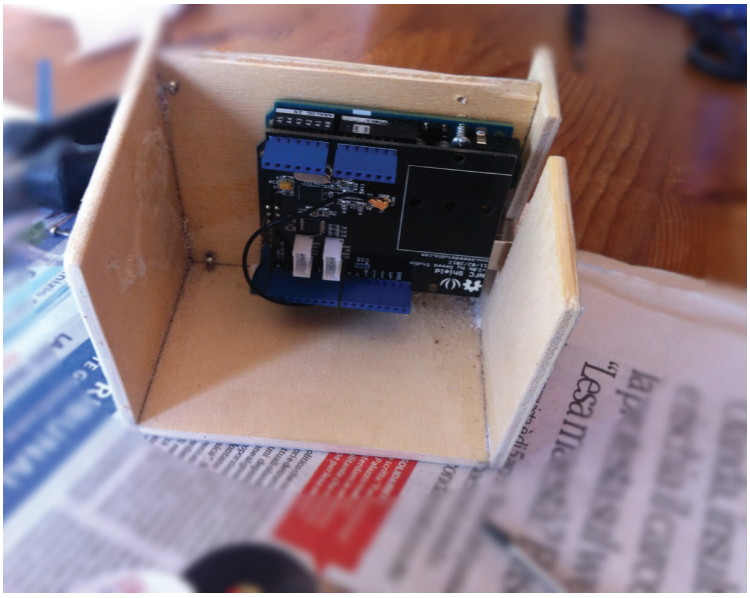
Costruiamo il jukebox



Nell'immagine a fianco, trovate il file in formato A3 che deve essere stampato e incollato su un supporto rigido come legno, cartoncino, poliplat, ecc... scegliete voi il materiale più adatto.
Clicca qui per scaricare il file .jpg da stampare.
Per il modello del jukebox è stato preso come riferimento il sito: http://www.wagner.de.com/blog/how-to-make-a-vintage-jukebox/ dove potete trovare un esempio realizzato con cartoncino.

Siete comunque liberi di creare il vostro jukebox come più vi piace.

Tagliate tutti i pezzi del jukebox e incollateli seguendo le istruzioni presenti nel foglio stampato.



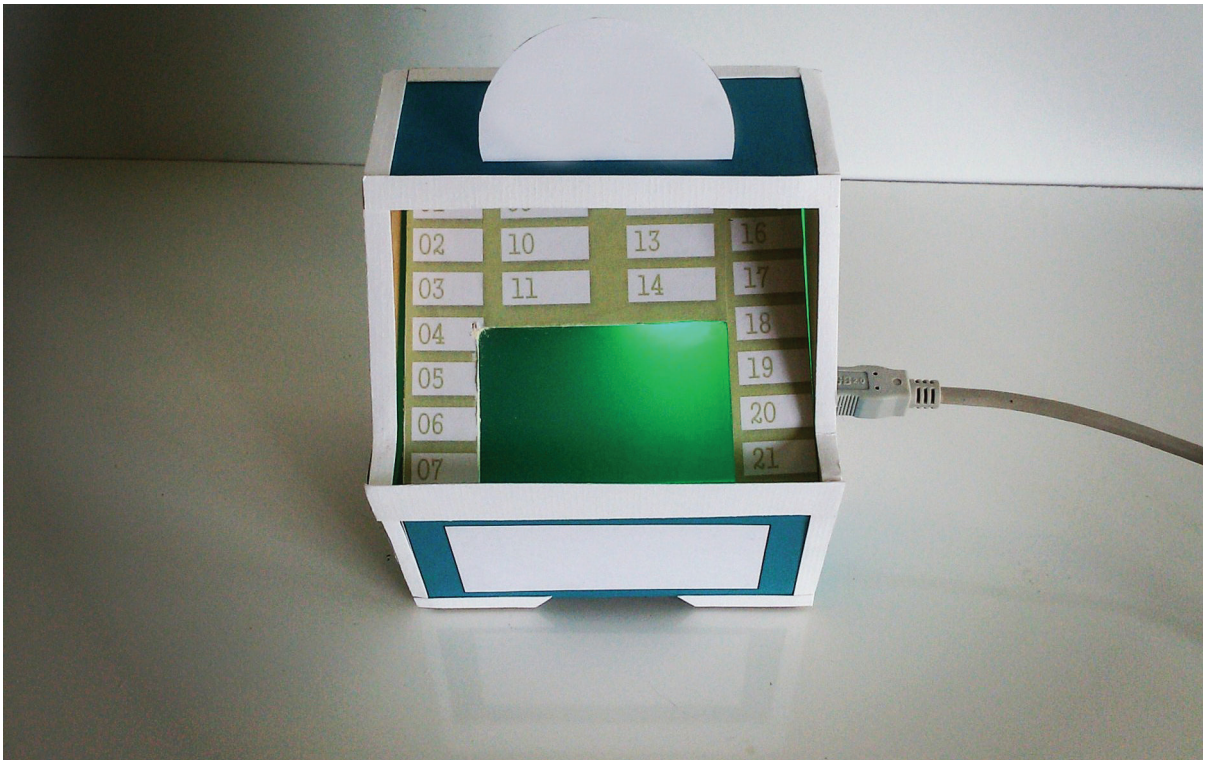
Inseriamo la NFC Shield sopra Arduino e fissiamoli con due viti, sotto il piano intermedio. Creiamo un buco sul lato destro del jukebox per inserire il cavo di programmazione.

Con il cartoncino blu costruite una specie di scivolo per indirizzare le monete inserite verso il basso. Il lettore NFC è posto vicino a questo scivolo di carta.

Con un secondo pezzo di carta allungate lo scivolo per portare la monete fuori dal jukebox.

Completiamo le rifiniture con delle piccole strisce di carta bianca da incollare ai bordi.

Incolliamo infine i tag NFC dietro ai dischi. Se volete potete personalizzare i vostri dischi con i vostri LP preferiti.



Programmiamo Arduino

Clicca qui per scaricare tutti gli sketch del tutorial Jukebox.

Apriamo il file **nfc_scrittura.ino**

Uplodiamo il codice su Arduino. Avviciniamo il primo NFC tag programmandolo con il messaggio "rock". Ripetiamo la stessa operazione con gli altri due tag inserendo le scritte "pop80" e "pop90".

I nostri tag NFC sono pronti per essere letti.

Apriamo il file **1_jukebox.ino**

...questo codice è stato spiegato nel tutorial precedente. Clicca qui per leggerlo.

```
    if ((payloadAsString.substring(1) ==
"rock")&(verifica==0)) {
    Serial.println("rock");
    verifica=1;
    }
    if ((payloadAsString.substring(1) ==
"pop80")&(verifica==0)) {
    Serial.println("pop80");
    verifica=1;
    }
    if ((payloadAsString.substring(1) ==
"pop90")&(verifica==0)) {
    Serial.println("pop90");
    verifica=1;
    }
```

Se il messaggio è uguale a "rock" allora scrivo sul Serial Monitor la scritta "rock" (questo messaggio verrà letto e interpretato da Python che manderà in esecuzione una canzone rock.

Programmiamo Python

Aprite il file controllo **1_playmusic.py** con Python IDLE.

Iniziamo a vedere con questo esempio come è facile gestire un file audio con Python. Installiamo la libreria pygame cliccando qui http://www.pygame.org/download.shtml. Scaricate la versione **pygame-1.9.1.win32-py2.7.msi** (per Python 2.7). Una volta scaricata doppio click e l'installazione è terminata.

PER UTENTI MAC: provate a seguire questa guida per installare la libreria pygame per Python 2.7 - clicca qui

```
import pygame
pygame.mixer.init(frequency=22050, size=-16, chan-
nels=2, buffer=4096)
song = pygame.mixer.Sound("test.ogg")
pygame.mixer.Sound.play(song)
```

Con queste poche righe possiamo far partire la riproduzione di un file audio.

test.ogg è il nome del file audio che viene messo in play

ATTENZIONE: il file audio deve essere in formato .ogg altrimenti non viene letto da Python.
Potete convertirlo con qualsiasi programma audio, qui di seguito una breve descrizione per convertire un file audio .mp3 in .ogg utilizzando VLC.

1 - scarica VLC - clicca qui

2 - Click su **Media / Converti Salva**.

3 - Click su **Aggiungi...** (selezionate il file audio mp3 che volete convertire)

4 - Click su **Converti / Salva**

5 - In **Profilo** selezionate **Audio - Vorbis (OGG)**

6 - In **File di destinazione** - click su **Sfoglia**

7- Inserite il nome del nuovo file, cliccate su **Salva** e poi su **Avvia**.

Torniamo a Python: inseriamo il titolo del file audio .ogg all'interno del codice al posto di "test.ogg" (il file audio deve essere nella stessa cartella dove è presente il file di Python).
Click su **Run - Run Module.**

Se tutto è andato bene, Python dopo qualche secondo fa partire il file audio.

Uniamo ora questo sketch che legge un file audio con quello che riceve dati da Arduino (tutorial precedente).

Aprite il file **2_playmusic_Arduino.py** con Python IDLE.

```
import serial
import time
import pygame
ard = serial.Serial('COM9', 9600, timeout=0)
pygame.mixer.init(frequency=22050, size=-16,
channels=2, buffer=4096)
while True:

  x = ard.readline()
  print ("Message from arduino: ")
  print (x)

  if "rock" in x:
    pygame.mixer.stop()
    song = pygame.mixer.Sound("rock.ogg")
    pygame.mixer.Sound.play(song)
  if "pop80" in x:
    pygame.mixer.stop()
    song = pygame.mixer.Sound("pop80.ogg")
    pygame.mixer.Sound.play(song)
  if "pop90" in x:
    pygame.mixer.stop()
    song = pygame.mixer.Sound("pop90.ogg")
    pygame.mixer.Sound.play(song)
  else:
    print 'nessun dato ricevuto !'

  time.sleep(2)
```

Se da Arduino arriva il messaggio "rock" blocco la canzone che è in fase di riproduzione e mando in play la canzone rock.ogg

La prima versione del jukebox è terminata, a ogni moneta è associata una canzone.
Di seguito vediamo una funzione di Python che ci permette di scegliere in maniera random (a caso) una canzone all'interno della cartella "rock", "pop80" o "pop90" a seconda della moneta inserita.

Jukebox #2 - Python e la funzione random

Convertiamo 4 canzoni rock in formato .ogg chiamandole **rock_0.ogg, rock_1.ogg, rock_2.ogg e rock_3.ogg**.

Aprite il file **3_playmusic_random.py** con Python IDLE.

```
import pygame
import random
numero_canzone= random.randint(0,3)
print numero_canzone
pygame.mixer.init(frequency=22050, size=-16, chan-
nels=2, buffer=4096)
canzoni_rock= ['rock_0.ogg', 'rock_1.ogg', 'rock_2.
ogg', 'rock_3.ogg']


song = pygame.mixer.Sound(canzoni_rock[numero_canzo-
ne])
pygame.mixer.Sound.play(song)
```

funzione che sceglie a caso un numero tra 0 e 3. Questo valore è inserito nella variabile numero_canzone.

Creo una lista di canzoni chiamata canzoni_rock.

Il numero random generato fa partire la canzone 0, 1 ,2 o 3.

Click su **Run - Run Module** per vedere come funziona la scelta casuale.

Per completare la seconda versione del Jukebox dobbiamo unire questo codice con quello relativo alla lettura del messaggio proveniente da Arduino.
Inoltre dobbiamo creare le canzoni pop80_1.ogg .... e le canzoni pop90_1.ogg ....(potete convertite quanti file volete).

Aprite il file controllo **4_playmusic_random_Arduino.py** con Python IDLE.

```
import serial
import time
import pygame
import random

ard = serial.Serial('COM9', 9600, timeout=0)
pygame.mixer.init(frequency=22050, size=-16, channels=2,
buffer=4096)

canzoni_rock= ['rock_0.ogg', 'rock_1.ogg', 'rock_2.ogg',
'rock_3.ogg']

canzoni_pop80= ['pop80_0.ogg', 'pop80_1.ogg', 'pop80_2.
ogg', 'pop80_3.ogg']

canzoni_pop90= ['pop90_0.ogg', 'pop90_1.ogg', 'pop90_2.
ogg', 'pop90_3.ogg']



while True:

  x = ard.readline()
  print ("Message from arduino: ")
  print (x)

  if "rock" in x:
    numero_canzone= random.randint(0,3)
    print numero_canzone
    pygame.mixer.stop()
   song = pygame.mixer.Sound(canzoni_rock_[numero_can-
zone])
   pygame.mixer.Sound.play(song)
  if "pop80" in x:
    numero_canzone= random.randint(0,3)
    print numero_canzone
    pygame.mixer.stop()
   song = pygame.mixer.Sound(canzoni_pop80_[numero_can-
zone])
   pygame.mixer.Sound.play(song)
  if "pop90" in x:
    numero_canzone= random.randint(0,3)
    print numero_canzone
    pygame.mixer.stop()
   song = pygame.mixer.Sound(canzoni_pop90_[numero_can-
zone])
   pygame.mixer.Sound.play(song)
  else:
    print 'nessun dato ricevuto !'

  time.sleep(2)
```

lista di canzoni rock

Se Arduino invia un messaggio con scritto "rock" allora scelgo casualmente un numero random tra 0 e 3 e mando in play la canzone relativa.