Mente e Cervello 2

Release 20191120

11 dic 2019

Raspberry PI

CAPITOLO 1

Introduzione a Raspberry PI

Raspberry Pi (da ora in poi mi riferirò ad esso chiamandolo semplicemente *Raspberry*, oppure con la sigla RPI) è un microcomputer delle dimensioni di una carta di credito. Progettato dalla *Raspberry PI Foundation*, la sua prima release è avvenuta nel 2012 e rappresenta uno dei migliori esempi di applicazione dei modelli di business promossi da Hardware libero e Software libero insieme.

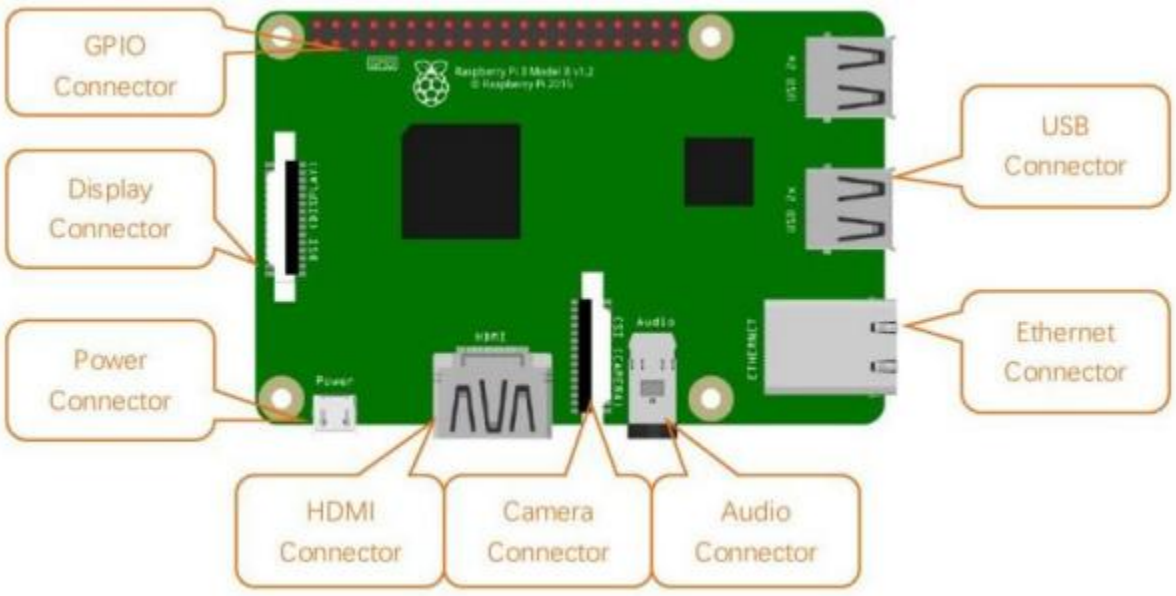
Suggerimento: Il suo sito ufficiale è https://www.raspberrypi.org/ in cui è possibile trovare una miriade di informazioni, documentazione, progetti, tutorial e perfino acquistare e farsi spedire un kit per gli esperimenti completo di RPI e di tutto ciò che serve!

Viene ampiamente utilizzato per l'implementazione di migliaia di progetti che includono workstation desktop a basso costo, media center, smart home, robot, server *tascabili*, sensoristica, prototyping, ecc.

Può eseguire un sistema operativo libero, basato su Linux o anche una versione specifica di Windows 10, chiamata Windows 10 IoT.

Contiene diverse interfacce hardware di utilizzo comune:

- USB.
- ethernet (rete cablata).
- HDMI.
- fotocamera.
- audio (jack da 3.5mm, quello delle cuffie).
- display
- GPIO (un'interfaccia generica a cui collegare qualsiasi sensore).
- Wifi e Bluetooth onboard



Finora, Raspberry Pi si è sviluppato fino alla quarta generazione. Le modifiche nelle versioni sono accompagnate da aumenti e aggiornamenti dell'hardware. Fortunatamente dalla versione 2 in avanti le interfacce e la GPIO si sono uniformati, in modo che qualsiasi progetto possa essere eseguito più o meno allo stesso modo in qualunque delle ultime versioni.

A scuola abbiamo una serie di *RPI versione 3, modello B+*, modello a cui si riferisce la figura precedente. La nostra trattazione da qui in avanti si rivolgerà esplicitamente a questo tipo di RPI. Sappiate comunque che con modifiche minime (e spesso nessuna) quello che vediamo può essere replicato in molte altre versioni di RPI.

CAPITOLO 2

Installazione

Per installare un qualsiasi sistema operativo su RPI abbiamo bisogno delle seguenti cose:

- un RPI :)
- un sistema operativo :))
- Una card MicroSD (farà da Hard Disk al raspberry)
- un adattatore USB oppure SD per scrivere sulla MicroSD (dipende se il vostro PC ha un lettore SD o solo porte USB)
- un cavo MicroUSB per alimentare il raspberry

A proposito di alimentazione del raspberry, ecco una tabella riassuntiva di quanto questi piccoli dispositivi necessitano a livello energetico

Prodotto	Corrente	Assorbimento Massimo	Consumo tipico
RPI 3 Model B	2.5A	1.2A	400mA
RPI 3 Model B+	2.5A	1.2A	500mA
RPI 4 Model B	3.0A	1.2A	600mA
RPI Zero W	1.2A	(limitato)	150mA

Se vogliamo fare proprio i precisoni, fra il materiale necessario bisogna elencare anche:

- Un monitor con ingresso HDMI (oppure VGA con adattatore HDMI)
- un mouse e una tastiera con uscite USB
- Un cavo ethernet per il collegamento alla rete (senza WIFI)

La procedura che segue mostrerà come installare **Raspbian**, il sistema operativo basato su Debian Linux e ottimizzato per raspberry.

A partire dal link: https://www.raspberrypi.org/downloads/raspbian/ saranno proposte 3 opzioni di download:

1. with desktop and recommended software
2. with desktop

3. lite

Io consiglio sempre di scaricare e installare la seconda (quella con il solo desktop), perché avere l'interfaccia grafica è un bel vantaggio per molti (la versione lite ha solo interfaccia testuale) ma il software *raccomandato* consiste in parecchi GB di software potenzialmente inutile ai nostri scopi.

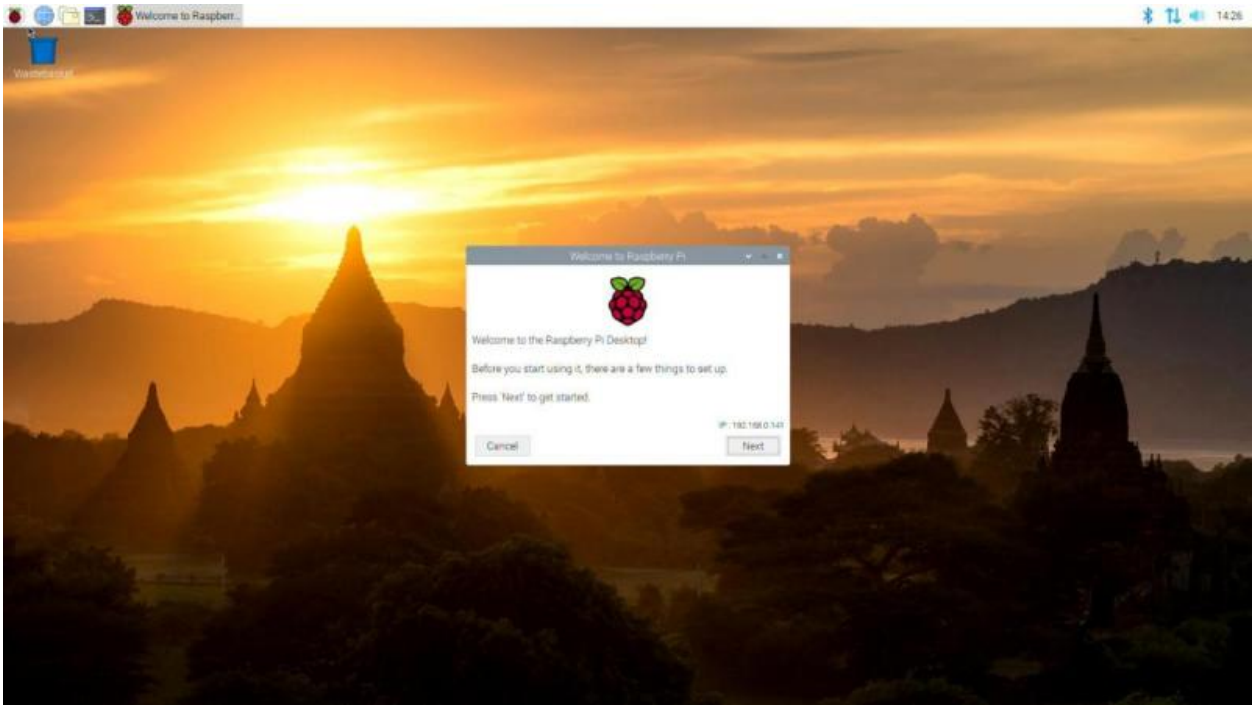
Scaricate lo ZIP ed **estraetelo**.

Adesso è necessario *copiare* il file **.img** nella MicroSD. Per farlo si può utilizzare un tool come *BalenaEtcher* (non ridete). Scaricatelo dal seguente link: https://www.balena.io/etcher/.

Fatta la copia del sistema operativo sulla scheda MicroSD, inseritela nel raspberry, collegate tutto ciò che serve (mouse, tastiera, monitor, rete) e per ultimo l'alimentazione.

Avvertimento: il raspberry non ha un tasto di accensione!

Il collegamento alla corrente deve essere quindi l'ultima cosa da fare quando si prepara un Raspberry per l'accensione.



Ecco fatto!
Buon Raspberry a tutti!

Nota: Il sistema operativo Raspbian propone di default il seguente utente.

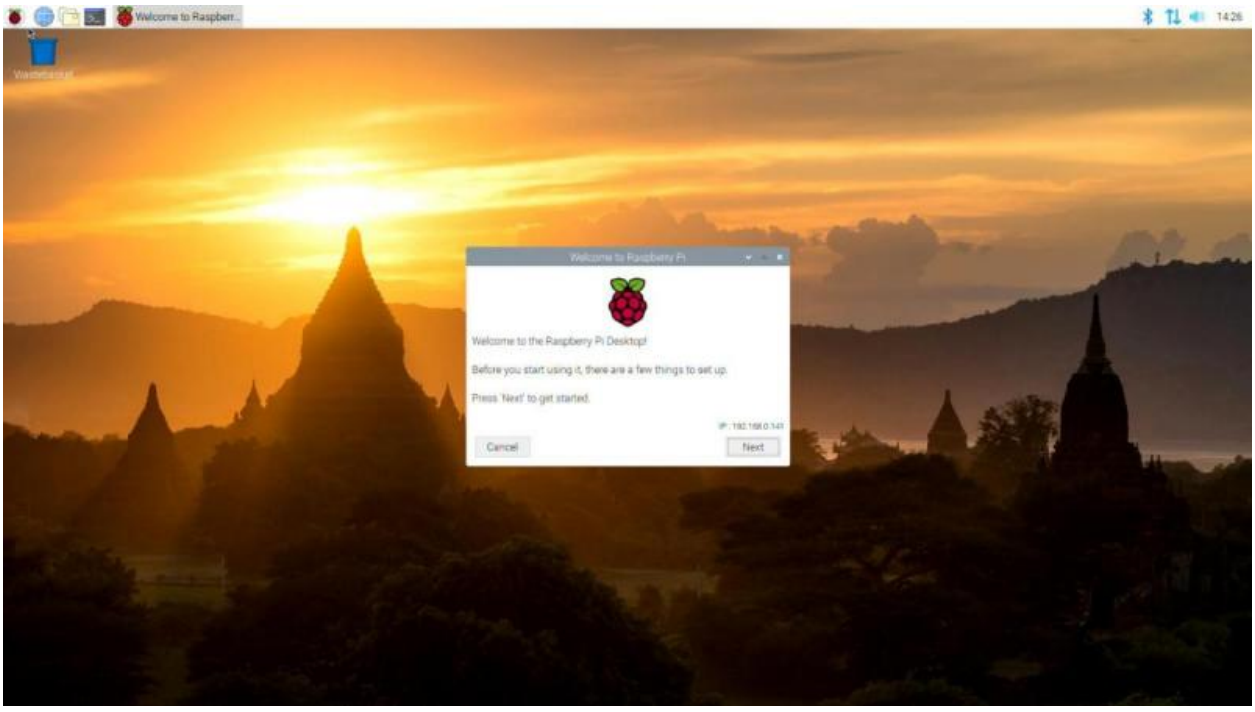
USER: pi
PASS: raspberry

Nella versione desktop comunque, l'utente ha l'accesso automatico abilitato (senza digitare la password), quindi nella realtà all'inizio questa informazione non serve. Per qualsiasi operazione *amministrativa* però, sarà richiesta l'autenticazione!

CAPITOLO 3

Operazioni di base

Ripartiamo dal Raspberry come lo abbiamo lasciato un attimo fa: acceso, pronto (caricamento iniziale finito), con interfaccia grafica



Ricordo inoltre le credenziali dell'account di default.

USER: pi
PASS: raspberry

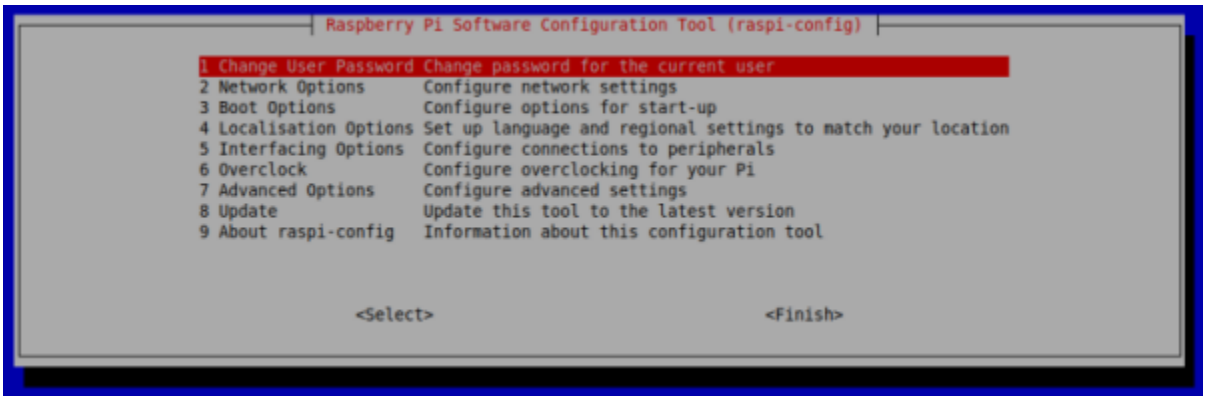
Il tool che ci viene presentato davanti serve per la configurazione iniziale. Mi raccomando di **non** cambiare la password dell'utente *pi* a meno che non siate assolutamente sicuri di ricordarvela. In caso siate senza interfaccia grafica, oppure vogliate procedere con la buona vecchia riga di comando, chiudete quella finestra e leggetevi il prossimo capitolo.

3.1 Configurazione Raspberry

Il sistema operativo Rasbian fornisce un tool a linea di testo da cui accedere a tutte le configurazioni di base del Raspberry. Per accedere ad esso si dovrà utilizzare il comando sudo.

```
$ sudo raspi-config
```

A quel punto si avrà accesso ad una interfaccia *grafica-testuale* con cui completare la configurazione:



Le opzioni sono in ordine e si può procedere tramite esse a configurare il Raspberry. Alcune opzioni sono chiarissime e non necessitano di spiegazioni; ad esempio quella di cambiare password :)

Mi dedico a quelle che di solito servono:

Network Options Le opzioni che trovate qui servono per impostare la rete. Sono cose che potete fare tranquillamente anche tramite l'interfaccia grafica, se la utilizzate.

L'opzione importante da settare all'inizio riguarda però l'hostname, il nome del vostro Raspberry! Tramite quello sarà più facile identificarlo nella rete! Per applicare questa impostazione serve di riavviare il Raspberry. Potete farlo anche alla fine di tutta la configurazione ;)

Localisation Options Queste opzioni servono per la localizzazione: tastiera italiana, lingua dei programmi, data, ora e loro formati, WiFi country code, etc... Sono molto importanti da impostare all'inizio in modo da non aver più problemi successivamente. Passate ognuna delle impostazioni alla ricerca delle opzioni italiane; per quanto riguarda il *locale* scegliete l'opzione **it_IT.UTF-8**.

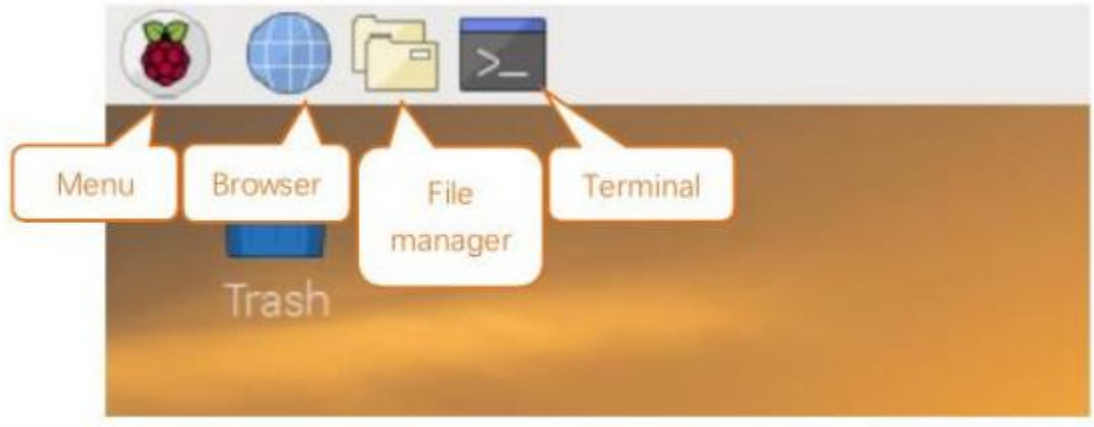
Interfacing Options In questo menù ci sono le opzioni per attivare o disattivare alcune periferiche o alcuni strumenti software per l'interazione con il Raspberry. Tra queste opzioni di interfaccia troviamo la camera, i bus I2C, 1-wire e SPI, la seriale, la remote GPIO, i software SSH e VNC. Attivate o disattivate ciò che vi serve (se non sapete che vi serve... disattivatelo). Per applicare le opzioni serve il riavvio.

Advanced Options Qui ci sono impostazioni abbastanza complicate, studiare... per credere!

Terminate le impostazioni è importantissimo **procedere ad un riavvio** per essere sicuro di aver applicato tutte le modifiche necessarie!

3.2 Prime operazioni

Riavviato il sistema operativo, diamo un occhio più da vicino al desktop del Raspberry



Poiché considero siate utenti di fascia medio-alta davanti ad un Sistema Operativo, immagino non abbiate problemi ad individuare le applicazioni più comuni che di certo utilizzeremo su Raspberry.

Le elenco:

- **Browser** (*Chromium*, la versione opensource di Google Chrome)
- **File manager** (*PCManFM*, un software per navigare tra i file del Raspberry)
- **Terminale** (*Terminal*, per accedere alla riga di comando)
- **Text Editor grafico** (*Mousepad*, un programma tipo il *Blocco Note* di Windows)
- **Text Editor testuale** (nel terminale: *nano*)
- **Editor Python** (*IDLE*, oppure *Thonny*... Facile!)

Considero inoltre che siate in grado di muovermi nel menù principale alla ricerca di una applicazione e sappiate interagire con la *system tray* per operazioni tipo controllare l'ora, abilitare/disabilitare bluetooth o wifi, montare e smontare una penna USB, etc...

Se volete prendere confidenza con queste operazioni, questo è il momento di farlo, **prima** di andare avanti e inesorabilmente iniziare a fingere di aver capito...

Nota: Fra le tante cose *nuove* su cui documentarsi, vale secondo me la pena di spendere 5 minuti e curiosare su Internet a proposito della **organizzazione del file system su linux**.

Provate a cercare quella frase in rete e a leggere qualcosa a proposito.

5 minuti, dai...

3.3 Il terminale Linux

Adesso vogliamo dedicare un pò di tempo a prendere confidenza con il terminale Linux. Utilizzare l'interfaccia testuale può essere molto vantaggioso in diversi casi:

- Tutti i sistemi Linux hanno la stessa interfaccia testuale, ma le interfacce grafiche sono potenzialmente tutte diverse

- La connessione remota ad un dispositivo in modalità testuale è veloce, sicura e facile da stabilire
- L'interfaccia testuale è molto potente. Pensate all'interfaccia grafica del vostro Sistema Operativo preferito:
  - Come si fa a controllare l'IP della macchina?
  - Come si fa a cercare un file all'interno di tutto il computer?
  - Come si fa a disintallare un programma? Arrestare un servizio?

Tutte queste operazioni costano un unico comando, una riga di codice con l'interfaccia testuale. E richiedono un secondo o poco più per l'esecuzione.

Adesso che ho attirato la vostra attenzione sull'utilizzo della linea di testo, vediamo alcuni semplici comandi organizzati per utilizzo:

Muoversi fra i file

Comando	Descrizione
ls	Elenca i file nella directory corrente (list)
cd	Cambia Directory.
pwd	Directory corrente

Manipolazione del testo

Comando	Descrizione
cat	Concatena i file e ne manda il contenuto nello standard output
less	Visualizza il contenuto di un file
nano	Editor testuale

Gestione di file e directory

Comando	Descrizione
mkdir	Crea una directory, una cartella
touch	Crea un file
cp	Copia un file o una directory
mv	Sposta un file o una directory
rm	Rimuove un file o una directory

Sistema

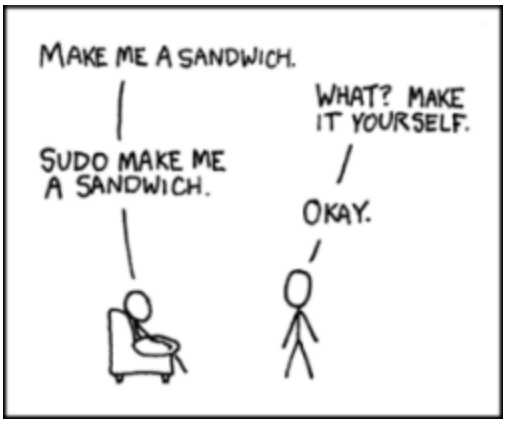
Comando	Descrizione
shutdown	Inizia la procedura di spegnimento
reboot	Riavvia il sistema

Utilities

Comando	Descrizione
history	Elenca la cronologia dei comandi digitati
man	Apre il manuale richiesto

Nota: Il comando **sudo** permette di eseguire qualsiasi comando con privilegi amministrativi.

Basta precedere *sudo* a qualsiasi comando per fare come se fosse l'amministratore del sistema ad eseguirlo.



3.4 Gestione software

L" **Advanced Packaging Tool** (conosciuto con l'acronimo APT) è il gestore standard di pacchetti software della distribuzione Debian e di tutte le sue derivate. In particolare vale la pena di ricordare Ubuntu e Raspberry come derivate di punta.

Questo sistema di gestione dei pacchetti è in grado di cercare, scaricare, installare qualsiasi software disponibile nei repository indicati nei file di configurazione per renderli disponibile all'istante!

Avvertimento: Poiché il comando APT si occupa di operazioni amministrative, deve essere sempre preceduto dal comando sudo.

Vediamo via via le opzioni di APT più importanti:

```
$ sudo apt update
```

Aggiorna l'elenco del software presente nel repository. In questo modo APT saprà qual è l'ultima versione del software disponibile online.

```
$ sudo apt upgrade
```

Sincronizza il software di sistema con quello presente nel repository. Praticamente permette di aggiornare tutto il software all'ultima versione disponibile.

```
$ sudo apt search package
```

Cerca il termine «package» fra i pacchetti software disponibili nel repository. Funziona anche senza sudo.

```
$ sudo apt install package
```

Scarica «package» e lo installa nel sistema, rendendolo disponibile all'utente.

```
$ sudo apt remove package
```

Rimuove «package» dal sistema.

CAPITOLO 4

Collegamento da remoto

Una volta che hai sistemato il raspberry e hai effettuato l'installazione del sistema operativo e dato un'occhiata al tutto, potrebbe tornarti utile **non** scollegare ogni volta mouse, tastiera, monitor e tutto il laboratorio, ma semplicemente collegare il raspberry alla rete e all'alimentazione e collegartici **da remoto!**

I metodi più utilizzati per la connessione remota al raspberry sono sostanzialmente 3: elencherò il nome comune della tecnologia, il software (server) che va installato su raspberry per renderlo disponibile e il client da utilizzare su Windows 10 per la connessione remota, visto che a scuola abbiamo PC con Windows 10. Se avete dispositivi con sistemi operativi Mac o Linux, documentatevi su Internet sui client per la corrispondente tecnologia.

Protocollo	Tipologia	Server (su RPI)	Client (su Win10)
RDP	Grafica	Xrdp	Remote Desktop
VNC	Grafica	vnc	VNC Viewer
SSH	Testuale	sshd	Putty

Avvertimento: Qualsiasi metodo sceglierai, ricordati che avrai bisogno di conoscere il **nome** e/o l'"**indirizzo IP** del tuo raspberry!

Cerca di capire **prima** come sia possibile ottenere (e magari modificare) queste informazioni!

4.1 RDP

Remote Desktop Protocol è un protocollo di rete proprietario sviluppato da Microsoft, che permette la connessione remota da un computer a un altro in maniera grafica. Il protocollo di default utilizza la porta TCP e UDP 3389.

I client RDP esistono per la maggior parte delle versioni di Microsoft Windows, Linux, Unix, macOS, Android, iOS. I server RDP ufficiali esistono per i sistemi operativi Windows nonostante ne esistano anche per i sistemi Unix-Like.

Suggerimento: Su RPI

Installa il servizio xrdp:

```
$ sudo apt install xrdp
```

Fatto questo, riavvia.

Suggerimento: Su Windows

Non devi fare nulla! Ti basta cercare il software *Connessione a Desktop Remoto*

4.2 VNC

Virtual Network Computing è un protocollo per applicazioni software di controllo remoto, utilizzato per amministrare il proprio computer a distanza. Può essere utilizzato anche per controllare in remoto server che non posseggono né monitor né tastiera.

Il protocollo di comunicazione usato a livello di trasporto è il TCP sulla porta di default 5900, oppure tramite interfaccia HTTP sulla porta 5800/tcp.

Suggerimento: Su RPI

Il server VNC è disponibile di default su Raspbian, ma va abilitato tramite **raspi-config**: Interfacing Options –> VNC –> Enable

Fatto questo, riavvia.

Suggerimento: Su Windows

Un client VNC gratuito è il VNC Viewer di RealVNC: https://www.realvnc.com/en/connect/download/viewer/windows/

Scaricalo, installalo su Windows e provalo.

4.3 SSH

Secure Shell è un protocollo che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica. È il protocollo che ha sostituito l'analogo, ma insicuro, Telnet, perché basato su una comunicazione **non** cifrata.

A livello server utilizza la porta 22, sia tramite TCP che UDP.

Suggerimento: Su RPI

Il server SSH è disponibile di default su Raspbian, ma va abilitato tramite **raspi-config**: Interfacing Options –> SSH –> Enable

Fatto questo, riavvia.

Suggerimento: Su Windows

Ti basta scaricare Putty e usarlo senza neanche installarlo!

Il sito ufficiale è: https://www.putty.org/

CAPITOLO 5

Multimedia

Quello che si vuole intendere in questa sezione come *Multimedia* è la possibilità di utilizzare con il nostro Raspberry alcune periferiche che poi torneranno utili nell'utilizzo dell'Intelligenza Artificiale.

Queste periferiche nello specifico sono:

- Una Webcam USB (utile per il riconoscimento facciale)
- Un microfono (per parlare, nella nostra dotazione è integrato nella webcam)
- le casse audio (per ascoltare musica e... la AI che parla!!!)

5.1 Webcam

Avvertimento: Per poter utilizzare la webcam (ovvero accedere al file della periferica che la rappresenta) bisogna far parte del gruppo **video**. L'utente *pi* è automaticamente parte di questo gruppo. Se vuoi verificare digita:

```
$ groups
```

Verranno elencati i gruppi di cui l'utente corrente fa parte. Se per qualche motivo *video* non fosse fra questi, bisogna aggiungere il proprio utente al gruppo con il comando:

```
$ sudo usermod -a -G video NOMEUTENTE
```

Per poter utilizzare la webcam, basta collegarla al Raspberry e installare il software necessario al suo utilizzo:

```
$ sudo apt install fswebcam
```

Questione di 1 minuto...

Per fare una prima prova e farsi una foto con la webcam, basta digitare:

```
$ fswebcam prova.jpg
```

Sorridete, poi aprite il file manager, andate a guardare la foto e controllate il risultato :)

Per modificare la risoluzione, si può agire con il parametro **-r**. Basta non esagerare. Ad esempio:

```
$ fswebcam -r 800x600 prova2.jpg
```

Se non vi piace la barra sotto (il *banner*), basta eliminarlo con l'opzione *–no-banner*

```
$ fswebcam -r 800x600 --no-banner prova3.jpg
```

Il risultato:

Più o meno tutto qui!

5.2 Casse audio

Queste sono davvero semplici da provare! Collegate le casse al Raspberry (jack e USB per l'alimentazione) e provate a sentire un video di Youtube! Se non funziona, forse il problema è che la periferica collegata al jack non è stata riconosciuta. Per assicurarci della cosa, proviamo con:

```
$ sudo raspi-config

Advanced Options ---> Audio ---> Jack
```

E questo è tutto!

5.3 Microfono integrato

Una volta positivamente testate le casse, si potrà provare anche con il microfono: basterà registrare qualche secondo di conversazione e poi provare a riascoltarla!

Per fare ciò, procediamo con le utility installate tramite pulseaudio. Per registrare:

```
$ arecord prova.wav
```

Dite una frase tipo «che bello questo corso!!!», aspettate un paio di secondi e poi premete la combinazione di tasti **CTRL + C** per interrompere la registrazione.

Per riascoltare le nostre soavi parole, digitate un bel:

```
$ aplay prova.wav
```

E anche questa è fatta!!!

CAPITOLO 6

Introduzione a Mycroft

Mycroft è una intelligenza artificiale opensource scritta in Python, che può essere una alternativa a prodotti commerciali come Amazon Alexa, Google Assistant, Apple Siri o Microsoft Cortana.

Suggerimento: Il sito ufficiale di Mycroft è https://mycroft.ai. Lì è possibile trovare tutte le informazioni, la documentazione e il codice relativo all'intelligenza artificiale, oltre alla possibilità di acquistare uno smart speaker :)

In un momento «storico» così complicato, in cui le intelligenze artificiale stanno prendendo piede e allo stesso tempo le persone cominciano a realizzare l'importanza della privacy online e di mantenere la sicurezza dei propri dati, ecco che avere una intelligenza «aperta» da contrapporre alle soluzioni commerciali «chiuse» può essere la soluzione alla questione.

CAPITOLO 7

Installazione

Suggerimento: Prima di procedere, assicuriamoci che il sistema operativo sia aggiornatissimo e pronto per le operazioni. Procediamo con un bel:

```
$ sudo apt update
$ sudo apt upgrade
```

In caso di reale aggiornamento, sempre per la maggior sicurezza, procediamo ad un riavvio.

Per installare Mycroft sul Raspberry, procediamo con i seguenti comandi nel terminale:

```
$ cd
$ git clone https://github.com/MycroftAI/mycroft-core.git
$ cd mycroft-core
$ bash dev_setup.sh
```

L'operazione «collettiva» che si va ad eseguire con il comando *dev_setup.sh* installa tutte le dipendenze del Raspberry, configura il sistema e prepara Mycroft all'esecuzione.

Ci vengono all'inizio proposte una serie di domande sull'installazione. Potete rispondere Sì a tutte, **tranne che a quelle sull'aggiornamento automatico e sulla compilazione di Mimic**: se abilitiamo l'aggiornamento automatico, ogni giovedì perdiamo 10 minuti. Se compiliamo Mimic, l'installazione durerà quasi 2 ore; senza Mimic ce la caviamo in 20 minuti...

Nota: Da quello che vedo, nel Raspberry è già installato di default il software **git** che viene utilizzato per scaricare il codice Mycroft. Nel caso il secondo comando fallisca per l'assenza di git, basta sistemare il tutto con un bel

```
$ sudo apt install git
```

L'installazione è tutta qui :)

CAPITOLO 8

Utilizzo

Se avete ancora il terminale aperto sulla cartella *mycroft-core* potete dare subito un *ls* e ricontrollare tutti i file presenti. Se non siete lì vi basta riaprire il terminale e digitare i seguenti comandi:

```
$ cd
$ cd mycroft-core
$ ls
```

Dovreste vedere la seguente situazione:

```
$ ls
bin doc mycroft scripts test coveralls.yml ACKNOWLEDGEMENTS.md CODE_OF_CONDUCT.md
LICENSE.md MANIFEST.in README.md dev_setup.sh requirements.txt setup.py start-mycroft.
↪sh
stop-mycroft.sh test-requirements.txt venv-activate.sh
```

Se provate ad eseguire start-mycroft.sh senza parametri dovreste vedere la finestra seguente:

```
$ ./start-mycroft.sh

usage: start-mycroft.sh [command] [params]

Services:
all                       runs core services: bus, audio, skills, voice
debug                     runs core services, then starts the CLI

Services:
audio                     the audio playback service
bus                       the messagebus service
skills                    the skill service
voice                     voice capture service
wifi                      wifi setup service
enclosure                 mark_1 enclosure service

Tools:
```

(continues on next page)

(continua dalla pagina precedente)

```
cli                         the Command Line Interface
unittest                    run mycroft-core unit tests

Utils:
skill_container <skill>  container for running a single skill
audiotest                   attempt simple audio validation
audioaccuracytest           more complex audio validation
sdkdoc                      generate sdk documentation

Examples:
start-mycroft.sh all
start-mycroft.sh cli
start-mycroft.sh unittest
```

A noi praticamente ne servono solo tre:

$ ./start-mycroft.sh audiotest serve per verificare il funzionamento del servizio audio. Buono per la prima volta... Sostanzialmente è un test di registrazione della propria voce e di riascolto, per testare microfono e casse. Se funziona tutto si può andare avanti.

$ ./start-mycroft.sh debug Questa opzione esegue tutto quello che serve per Mycroft più lancia la console di debug per la verifica del funzionamento iniziale e interattivo (ovvero, durante utilizzo). Useremo sempre questa opzione finché non siamo sicuri al 100% che tutto sia ok e potremo lanciare Mycroft *senza* interfaccia

$ ./start-mycroft.sh all Come debug, ma senza l'interfaccia per il debug, in modo che possa essere utilizzato solo *dialogando* ma senza interagire fisicamente con il terminale

Per fermare tutto abbiamo bisogno del secondo script: **stop-mycroft.sh**. Questo script non presenta alcuna opzione particolare: controlla tutti i servizi Mycroft e li ferma nel giusto ordine. Richiede una decina di secondi per essere eseguito:

```
$ ./stop-mycroft.sh
```

8.1 Pairing Mycroft

Il Pairing è il processo di sincronizzazione dell'intelligenza artificiale appena installata con una interfaccia web, che le permette una interazione con un sito esterno da cui caricare impostazioni, scaricare skills, etc...

Nota: L'applicazione web con cui si fa la sincronizzazione si chiama **selene** (https://github.com/MycroftAI/selene-backend), ma la sua installazione e configurazione va al di là del nostro corso. Per i nostri scopi utilizzeremo l'installazione ufficiale di Mycroft, disponibile al sito https://home.mycroft.ai.

Per effettuare il pairing, procedere con l'esecuzione di Mycroft e dei suoi servizi con in più l'opzione di debug:

```
$ ./start-mycroft.sh debug
```

Il caricamento iniziale è abbastanza oneroso in termini di tempo, soprattutto durante la primissima esecuzione dove vengono scaricate tutte le skills dal repository Mycroft. Dopo un pò Mycroft dovrebbe parlare o comunque scrivere qualcosa del tipo:

```
I am connected to the internet and need to be paired. Your 6-digit
Registration Code is X1Y2Z3
```

Dove le 6 cifre che ho scritto sono state scelte a caso. Quelle 6 cifre che il vostro device avrà scritto o pronunciato però... saranno fondamentali per registrare Mycroft online e terminare la prima configurazione.

Accedete al sito https://home.mycroft.ai, registratevi e selezionate «Add a device» dove inserirete le 6 cifre selezionate dalla vostra installazione di Mycroft. A quel punto nel sito ci sono poche e semplici opzioni da scegliere, tramite le quali terminare la prima parte di configurazione.

L'unica cosa che raccomando è quella di selezionare l'opzione **American Male** come voce :)

Una volta fatto il **pairing** è possibile iniziare ad utilizzare le skills di base.

8.2 Skills di base

Le skills predefinite installate con Mycroft sono quelle che ho elencato qui sotto.

Dedicate una decina di minuti a testarle tutte per verificarne il funzionamento. Se ci sono problemi con la «comprensione» fra voi e Mycroft (ovviamente per colpa del microfono, non del vostro inglese) provate a digitare le domande nell'interfaccia di debug.

Le skills preinstallate su Mycroft sono:

Alarms Hey Mycroft, set an alarm for two hours Hey Mycroft, set an alarm for 3pm

Audio record Hey Mycroft, record

Configuration Hey Mycroft, configuration update

DuckDuckGo Hey Mycroft, what is Frankenstein? Hey Mycroft, who is Kathryn Johnson?

Hello World Hey Mycroft, how are you?

IP address Hey Mycroft, what is your IP address?

Jokes Hey Mycroft, tell me a joke!

Naptime Hey Mycroft, go to sleep

News from NPR Hey Mycroft, news

Pairing Hey Mycroft, pair my Device

Personal Hey Mycroft, what are you

Reminders Hey Mycroft, remind me to turn off the oven in 5 minutes

Support information Create a support ticket You're not working! Send me debug info

Version checker Hey Mycroft, check version

Volume Hey Mycroft, increase volume Hey Mycroft, decrease volume

Weather Hey Mycroft, what is the weather?

Wikipedia Hey Mycroft, tell me about artificial intelligence Hey Mycroft, tell me about Grace Hopper

CAPITOLO 9

Personalizzazioni

9.1 mycroft.conf

La configurazione di Mycroft è memorizzata nel file **mycroft.conf**. In realtà i file di configurazione sono 4 e vengono applicati a cascata.

I livelli di configurazione sono:

1. **DEFAULT**: sono le impostazioni predefinite. Si trovano nella cartella mycroft-core, percorso `mycroft/configuration/mycroft.conf`
2. **REMOTE**: sono le impostazioni scaricate dal sito home.mycroft.ai o comunque dall'installazione con sui si é fatto il pairing.
3. **SYSTEM**: sono le impostazioni del dispositivo; si trovano nel file `/etc/mycroft/mycroft.conf`
4. **USER**: sono le impostazioni dell'utente locale (l'utente pi su Raspberry) e si trovano nel file `$HOME/.mycroft/mycroft.conf`

Come vedete sono praticamente 4 file mycroft.conf. Come funziona dunque? Come dicevo le impostazioni vengono applicate a cascata a partire dal primo file.

Proviamo a spiegare con un esempio: tra le impostazioni di DEFAULT troviamo il fuso orario «AmericaNew York» che viene dunque inizialmente applicato. Sul sito home.mycroft.ai non vengono impostate informazioni sul fuso orario, in modo tale che rimangono attive le impostazioni di default. A livello di sistema viene impostato il fuso orario «ItalyRome» che sovrascrive quello precedente, collocando correttamente il dispositivo. Se ci fossero nuove impostazioni nel file utente, queste andrebbero a sovrascrivere quelle impostate precedentemente.

Sostanzialmente se si ha a che fare con un Raspberry, le impostazioni da controllare (per rendere le cose più facili) sono solo 2: quelle sul sito **home.mycroft.ai** e quelle nel file utente, ricordandosi che le ultime vincono su tutte! Il file di sistema è meglio non utilizzarlo, in modo da non rendere più complicate le impostazioni.

Il file `mycroft.conf` è un file JSON standard (https://it.wikipedia.org/wiki/JavaScript_Object_Notation). Il consiglio principale è quello di cercare di mantenerlo il più breve possibile, in modo tale che sia più facile controllare le impostazioni.

Vi lascio un file di esempio con le impostazioni che può servire di impostare:

```
{
    // Language used for speech-to-text and text-to-speech.
    "lang": "en-us",

    // Measurement units, either 'metric' or 'english'
    // Override: REMOTE
    "system_unit": "metric",

    // Time format, either 'half' (e.g. "11:37 pm") or 'full' (e.g. "23:37")
    // Override: REMOTE
    "time_format": "full",

    // Date format, either 'MDY' (e.g. "11-29-1978") or 'DMY' (e.g. "29-11-1978")
    // Override: REMOTE
    "date_format": "DMY",

    // Play a beep when system begins to listen?
    "confirm_listening": true,

    // Location where the system resides
    // Override: REMOTE
    "location": {
        "city": {
            "code": "Jesi",
            "name": "Jesi",
            "state": {
                "code": "TM",
                "name": "The Marches",
                "country": {
                    "code": "IT",
                    "name": "Italy"
                }
            }
        },
        // DA SISTEMARE
        "coordinate": {
            "latitude": 38.971669,
            "longitude": -95.23525
        },
        "timezone": {
            "code": "America/Chicago",
            "name": "Central Standard Time",
            "dstOffset": 3600000,
            "offset": -21600000
        }
    },

    // Speech to Text parameters
    // Override: REMOTE
    "stt": {
        // Engine.  Options: "mycroft", "google", "google_cloud", "wit", "ibm", "kaldi
↪", "bing",
        //                   "houndify", "deepspeech_server", "govivace", "mycroft_
↪deepspeech"
        "module": "mycroft"
    },
```

(continues on next page)

(continua dalla pagina precedente)

```
    // Text to Speech parameters
    // Override: REMOTE
    "tts": {
        // Engine.  Options: "mimic", "mimic2", "google", "marytts", "fatts", "espeak
↪", "spdsay", "watson", "bing", "responsive_voice"
        "module": "mimic2",
    },

}
```

A proposito... per testare «Mimic2», dall'interfaccia WEB, selezionare «American Male»!

9.2 Italianizzare Mycroft

Questa cosa che proviamo qui è altamente sperimentale e non siamo per niente sicuri che funzionerà... è una delle cose belle di questo corso :)

Per utilizzare Mycroft in un'altra lingua, cioè in Italiano, dobbiamo andare a modificare 6 impostazioni:

1. **le impostazioni del linguaggio** nel file mycroft.conf
2. la **Wake Word** per renderla adatta al proprio linguaggio
3. il motore **STT** (Speech To Text), scegliendone uno che supporti il proprio linguaggio
4. il motore **TTS** (Text To Speech), come sopra
5. le **Skills** che devono supportare il nuovo linguaggio
6. il supporto della lingua scelta nel modulo **Lingua Franca**

La cosa un pò più complicata è quella di personalizzare la wake word in maniera generica. L'attuale documentazione di fornisce questi suggerimenti: https://mycroft-ai.gitbook.io/docs/using-mycroft-ai/customizations/mycroft-conf#changing-your-wake-word

Per quanto riguarda Skills e Lingua Franca, tutto dipende dal fatto che la lingua che desideriamo sia stata implementata oppure no. Per questi due aspetti, nel caso dell'italiano, la risposta è sì!!

Le impostazioni del linguaggio e i motori STT e TTS possono essere modificati tramite il file di configurazione. Lascio qui sotto un esempio:

```
{
    "lang": "it-it",

    "stt": {
        "module": "mycroft",
        "mycroft": {
            "lang": "it-it"
        }
    },

    "tts": {
        "module": "google",
        "google": {
            "lang": "it"
        }
    }
}
```

CAPITOLO 10

OpenCV

Liberamente ispirandomi a Wikipedia, posso dirvi che OpenCV (acronimo in lingua inglese di Open Source Computer Vision Library) è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale.

È una libreria software libera originariamente sviluppato da Intel e poi presa in carico dalla comunità opensource. Il linguaggio di programmazione principalmente utilizzato per sviluppare con questa libreria è il C++, ma è possibile interfacciarsi ad essa anche attraverso il C, il Python e Java.

10.1 Installazione su RPI3 (Raspbian Buster)

L'installazione della libreria su Raspbian Buster è notevolmente semplificata dalla presenza di alcuni pacchetti precompilati nei repository.

L'installazione descritta di seguito è relativo alla versione 3.4.x nonostante sia già disponibile, anche su Raspbian, la versione 4.1.x. Purtroppo nei nostri test di laboratorio abbiamo avuto qualche problemino con essa...

Procedete come di seguito. Ricordate inoltre che qualsiasi operazione di installazione del sistema inizia con l'aggiornamento dello stesso...

```
$ sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt install libxvidcore-dev libx264-dev
$ sudo apt install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
$ sudo apt install libhdf5-dev libhdf5-serial-dev libhdf5-103
$ sudo apt install libatlas-base-dev gfortran
$ sudo apt install libfontconfig1-dev libcairo2-dev
$ sudo apt install libgdk-pixbuf2.0-dev libpango1.0-dev
$ sudo apt install libgtk2.0-dev libgtk-3-dev
$ sudo apt install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
$ sudo apt install python3-dev
```

Poi, finalmente...

```
$ sudo pip3 install opencv-contrib-python==3.4.3.18
```

10.2 TEST

Al termine dell'installazione si può procedere con il primo test di funzionamento, che verifica semplicemente se l'installazione è andata a buon fine. Il test prova semplicemente a caricare il modulo `cv2` all'interno dell'interprete `python3`.

```
$ python3
Python 3.7.3 (default, Apr  3 2019, 05:39:12)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>>
```

10.3 LAB 00 - Controlliamo un pò di versioni

Prima di procedere, assicuriamoci, ancora, di avere installato correttamente le versioni di Opencv e di Python. Ancora , direte voi... beh la prudenza non è mai troppa. Probabilmente troverete già installato il programma Thonny sul vostro Raspberry. Thonny è un semplice ed «abbastanza» completo IDE (Integrated Development Environment) per sviluppare applicazioni in Python. Quindi, caricate su Thonny il testo sotto indicato ed eseguitelo... Se non dovesse funzionare potete sempre utilizzare la shell di comandi (terminale).

```
import cv2 as cv
import sys
print("Versione Python")
print(sys.version)
print ("versione OpenCv")
print(cv.__version__)
```

10.4 LAB 01 - Carichiamo un'immagine

Cominciamo a prendere un pò di confidenza con le immagini. Carichiamone una e proviamo a comprendere il codice sotto riportato.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
#legge un file in formato .jpg mostrato poi in due finestra
#una di questa mostra l'immagine in scala di grigi

img = cv2.imread('watch.jpg',cv2.IMREAD_GRAYSCALE)
img1 = cv2.imread('watch.jpg')
cv2.imshow('imageGray',img)
cv2.imshow('imageColor',img1)
#wait for a pressed key
cv2.waitKey(0)
cv2.destroyAllWindows()
```

10.5 LAB 02 - Ancora sulle immagini

Ancora sulle immagini. Questa volta invece di usare una immagine trasformata in scala di grigio, usiamo l'immagine a colori. Notate il diverso tasto di controllo per uscire dal programma e la possibilità di salvare l'immagine sotto un altro nome.

```
import cv2 as cv
import sys
print("Versione Python")
print(sys.version)
print ("versione OpenCv")
print(cv.__version__)

#import numpy library to do some draws
import numpy as np
# Load an color image in grayscale
img = cv.imread('cam.jpg',cv.IMREAD_COLOR)
print ("immagine caricata")
cv.imshow('image', img)
#wait for a pressed key
k=cv.waitKey(0)
if k == 27:         # wait for ESC key to exit
        cv.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
        cv.imwrite('camsalvata.png',img)
        cv.destroyAllWindows()
```

10.6 LAB 03 - Disegniamo qualcosa. Forme, parole, linee... liberate la vostra fantasia e create nuove immagini

```
import numpy as np
import cv2

img = cv2.imread('watch.jpg',cv2.IMREAD_COLOR)
cv2.line(img,(0,0),(200,300),(255,255,255),5)
cv2.rectangle(img,(15,25),(200,250),(0,0,255),5)
cv2.circle(img,(100,63), 55, (0,255,0), -1)

pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
# OpenCV documentation had this code, which reshapes the array to a 1 x 2. I did not
# find this necessary, but you may:
#pts = pts.reshape((-1,1,2))
cv2.polylines(img, [pts], True, (0,255,255), 3)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Ciao Mondo!',(0,50), font, 1, (200,255,155), 2, cv2.LINE_AA)
#now show the modified image
cv2.imshow('image',img)
#wait for a pressed key
cv2.waitKey(0)
cv2.destroyAllWindows()
```

10.7 LAB 04 - Finalmente un pò di video

```
import numpy as np
import cv2
#a little of videos
#reading video files
cap = cv2.VideoCapture(0)

while(True):
        ret, frame = cap.read() #il modo più semplice per leggere un file video

        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
                break

cap.release()
cv2.destroyAllWindows()
```

10.8 LAB 05 - Apriamo qualche finestra

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
while(True):
        # Capture frame-by-frame
        ret, frame = cap.read()
        # Our operations on the frame come here
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        #color = cv.cvtColor(frame, cv.COLOR_BRG)
        # Display the resulting frame
        cv.imshow('frame',gray)
        cv.imshow('frame1',frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
                break
# When everything done, release the capture
cap.release()
cv.destroyAllWindows()
```

10.9 LAB 06 - Visi: riconosciamoli

```
import cv2
import sys

#cascPath = sys.argv[1]
#faceCascade = cv2.CascadeClassifier(cascPath)
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
video_capture = cv2.VideoCapture(0)

while True:
        # Capture frame-by-frame
        ret, frame = video_capture.read()
```

(continues on next page)

(continua dalla pagina precedente)

```
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        faces = faceCascade.detectMultiScale(
                gray,
                scaleFactor=1.1,
                minNeighbors=5,
                minSize=(30, 30),
                #flags=cv2.cv.CV_HAAR_SCALE_IMAGE
        )

        # Draw a rectangle around the faces
        for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        # Display the resulting frame
        cv2.imshow('Video', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
                break

# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

10.10 LAB 07 - Ogni viso ha i suoi occhi

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while 1:
        ret, img = cap.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,255),1)
                roi_gray = gray[y:y+h, x:x+w]
                roi_color = img[y:y+h, x:x+w]

                eyes = eye_cascade.detectMultiScale(roi_gray)
                for (ex,ey,ew,eh) in eyes:
                        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)

        cv2.imshow('img',img)
        k = cv2.waitKey(30) & 0xff
        if k == 27: #esc 27 ascii
                break
```

(continues on next page)

(continua dalla pagina precedente)

```
cap.release()
cv2.destroyAllWindows()
```

10.11 LAB 08 - Alla ricerca del colore

Ponete un oggetto di colore verde davanti alla vostra webcam e vediamo cosa succede. Una pallina sarebbe l'oggetto perfetto.

```
import cv2
import numpy as np
import math

# creo l'oggetto per l'acquisizione del video inserendo 0 il video verra acquisito␣
↪dalla telcamera,
# inserendo il nome di un file video (posto nella directory del programma) verra␣
↪aperto quello.
cap = cv2.VideoCapture(0)

# Controllo che la telecamera sia disponibile
if (cap.isOpened()== False):
  print("IMPOSSIBILE ACQUISIRE IL VIDEO!")

# Eseguo finche il video e disponibile

while(cap.isOpened()):
  # leggo frame per frame
  ret, frame = cap.read()
  if ret == True:

        #converto in formato hsv
        frame1= cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        #applico una sfumatuta per ridurre i disturbi
        frame2= cv2.GaussianBlur(frame1,(5,5),0)

        #definisco i margini di colore da filtrare
        chiaro=np.array([50,80,80])
        scuro=np.array([80,200,200])

        #filtro l'immagine secondo i colori definiti
        frame3=cv2.inRange(frame2, chiaro,scuro)

        #applico una sfumatuta per ridurre i disturbi
        _,frame4= cv2.threshold(frame3,127,255,0)

        #calcolo l'area della superficie colorata e ottengo di conseguenza il centro
        moments = cv2.moments(frame4)
        area = moments['m00']

        radius =int( (math.sqrt((area/3.14)))/10)
        centroid_x, centroid_y = None, None

        if area != 0:
                c_x = int(moments['m10']/area)
```

(continues on next page)

(continua dalla pagina precedente)

```
                c_y = int(moments['m01']/area)
                print("x: ", c_x,"y: ",c_y)

        # se e stato trovato un oggetto corrispondente ai citeri di ricerca(colore),
        # utilizzo le sue coordinate sullo schermo per tracciare un cerchio attorno␣
↪ad esso

        if c_x != None and c_y != None:

                # disegno il cerchio
                cv2.circle(frame, (c_x,c_y), radius, (0,0,255),2)

        #disegno la griglia sullo schermo
        cv2.line(frame,(320,0),(320,480),[255,0,0],1)
        cv2.line(frame,(0,240),(640,240),[255,0,0],1)
        cv2.rectangle(frame,(310,230),(330,250),[0,0,255],1)


        # proietto il video acquisito in una finestra
        cv2.imshow('Frame',frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
                break

  # esco dal loop
  else:
        break

# chiudo il file video o lo stream della telecamera
cap.release()

# Chiudo la finestra creata
cv2.destroyAllWindows()
```

CAPITOLO 11

Skills

Creiamo una skill per Mycroft

11.1 ColorSkill

In questa skill chiederemo a Mycroft il suo colore preferito.

11.2 DoubleSkill

In questa skill diremo a Mycroft un numero e lui calcoler? il suo doppio.