

Alma Mater Studiorum - Università di Bologna
Corso di Laurea in Ingegneria Informatica Magistrale

Progetto di Sistemi Digitali
December 2022

Face Recognition - PAM

Leonardo Bambini
Patrick Di Fazio

Indice

1	Introduzione	2
2	Riconoscimento Facciale	3
3	PAM	4
4	Applicazione	5
4.1	Deploy	5
4.2	Utilizzo	5
4.3	Scelte Progettuali: Container	5
5	Testing	6
5.1	Face Recognition	6
5.2	Hardware	6
6	Cablaggio	7
6.1	Bredboard e Collegamenti	7
6.2	Webcam	8
6.3	I2C Oled Display	9
7	Struttura	10
7.1	Case	10
7.2	3D Modeling	12
7.3	Bill of Material (BOM)	13
7.4	Repository e Sorgenti	13
7.5	Risorse e Documentazioni	13
7.5.1	Documentazioni PAM	13
7.5.2	Documentazioni OpenCV	13
8	Conclusioni	14

Elenco delle figure

1	Esempio di riconoscimento facciale	3
2	Architettura di PAM	4
3	Testing	6
4	Circuito sulla breadbord	7
5	Webcam con cavo USB	8
6	Webcam con cavo Flex	8
7	Emulazione Webcam tramite DroidCam	8
8	Stato del sistema in idle	9
9	Pinout Raspberry PI 3	9
10	Pinout Display	9
11	Case	10
12	Cablaggio	10
13	Vista da dietro	11
14	Vista da sotto	11
15	Vista laterale	11
16	Vista frontale	11
17	Assonometria	12
18	Esploso	12

1 Introduzione

I sistemi di autenticazione tradizionali prevedono in gran parte l'utilizzo di una password per dimostrare la propria identità. Un'autenticazione passiva one factor è una modalità piuttosto debole poiché se la password venisse scoperta per colpa dell'utente o per attacchi informatici consentirebbe a un intrusore di assumere l'identità della vittima.

Le password tradizionali hanno anche la necessità di essere frequentemente cambiate e devono avere una complessità tale da scongiurare i vari attacchi possibili, due abitudini che l'utente medio non rispetta favorendo la semplicità e la velocità d'utilizzo dei sistemi a cui ci si vuole autenticare. Un sistema che preveda un'autenticazione mediante riconoscimento facciale permetterebbe di non avere in carico la gestione per l'utente di una password con tutte le problematiche precedentemente discusse.

Il progetto ha come scopo la realizzazione di un sistema di autenticazione facciale basato su **PAM** (*Pluggable Authentication Module*), una “tecnologia” adottata in diversi contesti di sicurezza tra cui il login su Linux, unita al riconoscimento facciale dell'utente.

2 Riconoscimento Facciale

Il riconoscimento facciale non è altro che il risultato di un confronto di due facce, di cui di una si ha la certezza di chi sia. L'utente si autentica se la faccia fornita è simile a quella registrata (con una certa tolleranza).

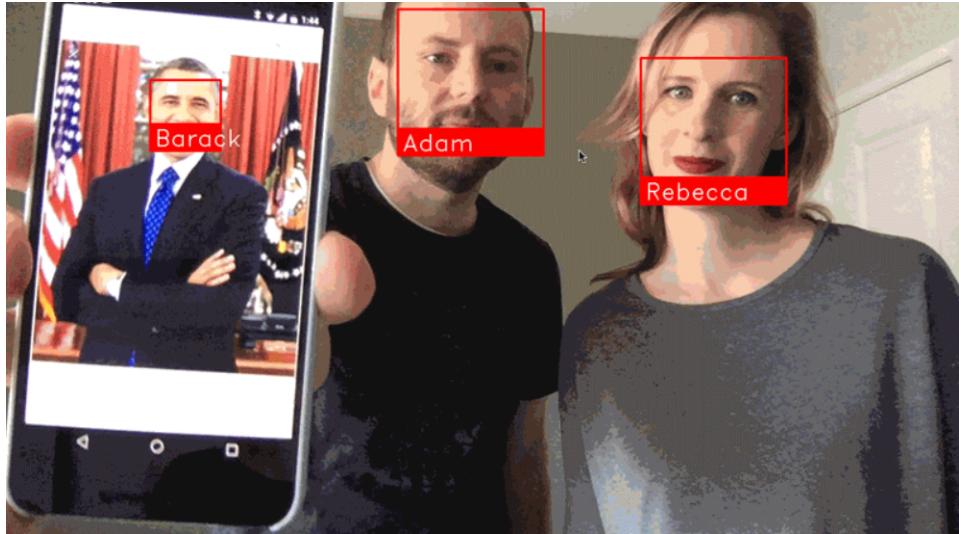


Figura 1: Esempio di riconoscimento facciale

Per il nostro sistema abbiamo sfruttato il seguente progetto: https://github.com/ageitgey/face_recognition Questa libreria si basa su **dlib**, un' altra libreria di Computer Vision con funzionalità specifiche per il riconoscimento facciale

In dlib le facce sono rappresentate da un array a 128 dimensioni. Il confronto di questo array (detto encoding) con quello di una faccia nota (precedentemente registrata) consente di poter stabilire con una certa tolleranza se i due encoding appartengono alla stessa persona

Per poter effettuare il riconoscimento facciale è necessario che l'utente si registri presso il servizio che vuole utilizzare (nel nostro caso PAM) attraverso il deposito dell'encoding della sua faccia, nel nostro progetto questa operazione è fatta mediante lo script *facial_signup_novideo.py*

Successivamente l'autenticazione facciale sarà fatta in maniera automatica al lancio dei comandi, che attivano l'uso della webcam, faranno una serie di foto (con un timeout di 10 secondi) e se almeno una di esse sarà giudicata abbastanza simile a quella che l'utente ha registrato (verifica mediante score di matching e tolleranza della rete) l'autenticazione sarà completata con successo. In caso contrario si ricorrerà alla tradizionale password.

3 PAM

Il progetto si basa su **PAM** (Pluggable Authentication Module), ovvero un sistema di autenticazione a moduli che è alla base dell'autenticazione nei moderni sistemi Linux.

I moduli PAM possono essere usati per qualunque tipo di autenticazione sul sistema. Questo vuol dire che il modulo di accesso biometrico può essere poi utilizzato in modo General Purpose e può essere anche agganciato ad altri applicativi (es: sblocco di un portafoglio elettronico). L'utente tramite la propria faccia, al momento del login e quindi subito dopo l'attivazione della webcam, avrà un encoding generato dal sistema che verrà confrontato con l'encoding della foto salvata sul database.

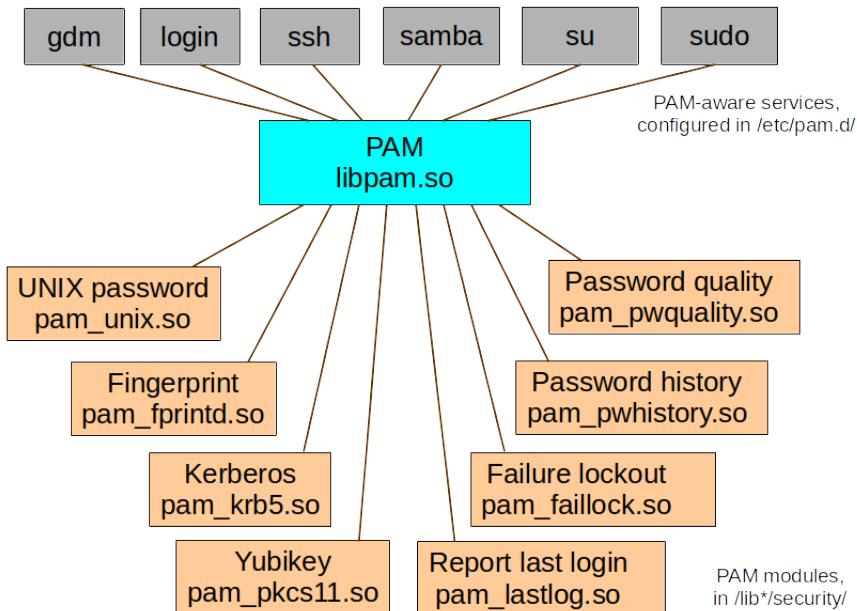


Figura 2: Architettura di PAM

Solitamente i moduli di PAM sono scritti in C e sono a stretto contatto con il kernel. Nell'implementazione del progetto utilizziamo un plugin python per PAM su linux, **libpam-python** che permette di farci agganciare un modulo esterno scritto in python e programmato secondo le nostre esigenze, in modo da usare le librerie di face recognition. Il codice che gestisce l'autenticazione sarà quindi

```

def pam_sm_authenticate(pamh, flags, argv):
    user = pamh.get_user()
    res = login(user)
    if(res):
        return pamh.PAM_SUCCESS
    else:
        return pamh.PAM_AUTH_ERR

```

dove *login(user)* si occuperà del riconoscimento facciale dell'utente.

Come in figura, generiamo un nuovo modulo *pam-python.so*, abbinato al codice python *facial_pam_auth.py* aggiungendo a */etc/pam.d/common-auth* la configurazione

```
auth sufficient pam_python.so facial_pam_auth.py
```

In caso si voglia utilizzare il sistema anche per le fasi di login, si vanno a modificare anche i file */etc/pam.d/login*. Queste modifiche sono applicabili ad ogni ramo della Figura 2, possiamo personalizzare così il sistema a piacimento secondo i requisiti di sicurezza desiderati (ad esempio vogliamo utilizzare lo sblocco facciale solo per il login dell'utente, ma non per accedere ai privilegi di **root**).

4 Applicazione

Il progetto è stato sviluppato in due differenti branch. In uno si segue lo sviluppo del progetto su Raspberry, nel secondo si generalizza il progetto per un sistema che dispone di webcam, come un computer portatile o fisso. Questo ramo non prevede la circuiteria e l'integrazione con i pin General Purpose perché non disponibili. In questo foglio di progetto seguiamo lo sviluppo su Raspberry. Il sistema operativo utilizzato è Debian.

4.1 Deploy

Per riuscire ad eseguire una demo del progetto è necessario disporre di una webcam, installare Docker e avere i permessi di root sul proprio sistema.

L'installazione dell'applicazione può essere eseguita tramite delle immagini prefatte, oppure facendo il "build" manuale dei container. Nella parte di testing, dato che questa fase risultava particolarmente pesante per il Raspberry, sfruttando l'architettura ARM è stato possibile eseguire "cross-compiling" da parte di un computer con la stessa architettura. Per questo scopo è stato utilizzato un MacBook Air con processore M1, basato su Apple Silicon, in grado di creare container eseguibili su ARMv8.

4.2 Utilizzo

Per riuscire ad utilizzare le API a basso livello della webcam e dei pin GPIO, è necessario fare il "run" del container in modalità *privileged*.

```
sudo docker run --privileged -it -d --name PAM debian-pam:v1
```

Per utilizzare il container poi si utilizza

```
sudo docker exec -it PAM /bin/bash
```

Successivamente per utilizzare le funzionalità fornite del progetto, l'utente si registra premendo un bottone, salvando l'immagine della faccia sul database.

Il riconoscimento facciale viene scatenato in seguito a qualunque operazione che richiede un'elevazione di privilegi, come ad esempio:

```
sudo su
```

Oppure se si passa da un utente ad un altro:

```
su user
```

O durante un'operazione di login tramite

```
login user
```

4.3 Scelte Progettuali: Container

Grazie all'uso dei container è possibile testare il sistema in modo "sicuro", ovvero non si va a modificare il comportamento di login di default del sistema che si sta utilizzando. In più utilizzando una sola immagine Docker su una distribuzione prestabilita come Debian, è possibile distribuire la stessa versione di test. Sul registry sono presenti due release: una con architettura **amd64** e una con architettura **aarch64** per eseguire l'applicazione sia su sistemi Linux classici, sia su Raspberry Pi 3/4 ed Apple Silicon.

5 Testing

5.1 Face Recognition

Per ottimizzare le prestazioni della rete che crea gli encoding a partire da una foto, sono stati fatti diversi test in cui abbiamo verificato empiricamente quale sia la tolleranza della rete che permetta di avere il miglior trade off tra precisione nell'identificazione e resistenza in usi non ideali. Il confronto tra l'encoding della faccia nota e l'encoding della faccia che si vuole autenticare produce uno score, che va da 0 a 1 (0 simile, 1 diverso).

La tolleranza, ovvero il valore massimo che può avere uno score per decretare un'autenticazione riuscita, se impostata a un valore molto basso garantisce una maggior precisione nell'identificazione poiché c'è più similarità tra le due facce e quindi minor rischio di falsi positivi. Tuttavia una tolleranza eccessivamente bassa rende il sistema utilizzabile solo in condizioni ideali (buona luminosità, webcam frontale ad altezza faccia, qualità dell'immagine elevata), quindi è necessario fare un trade off con la precisione per rendere il sistema più resistente ai casi d'uso non ideali.

Si elencano di seguito i risultati dei test fatti per determinare la miglior tolleranza:

Persone diverse:	0.76
Stessa persona foto simile	0.23
Stessa persona con faccia un po' girata	0.31
Stessa persona con scarsa illuminazione	0.45

Dai test sopra riportati si evince che un buon compromesso è 0.5

5.2 Hardware

Nella parte di testing hardware i test sono stati eseguiti tramite un setup temporaneo del cablaggio. In questa fase si è testato l'uso e il riconoscimento della webcam, il funzionamento dei led, del bottone e dello schermo. Il collegamento con il Raspberry è remoto e tramite **SSH**.

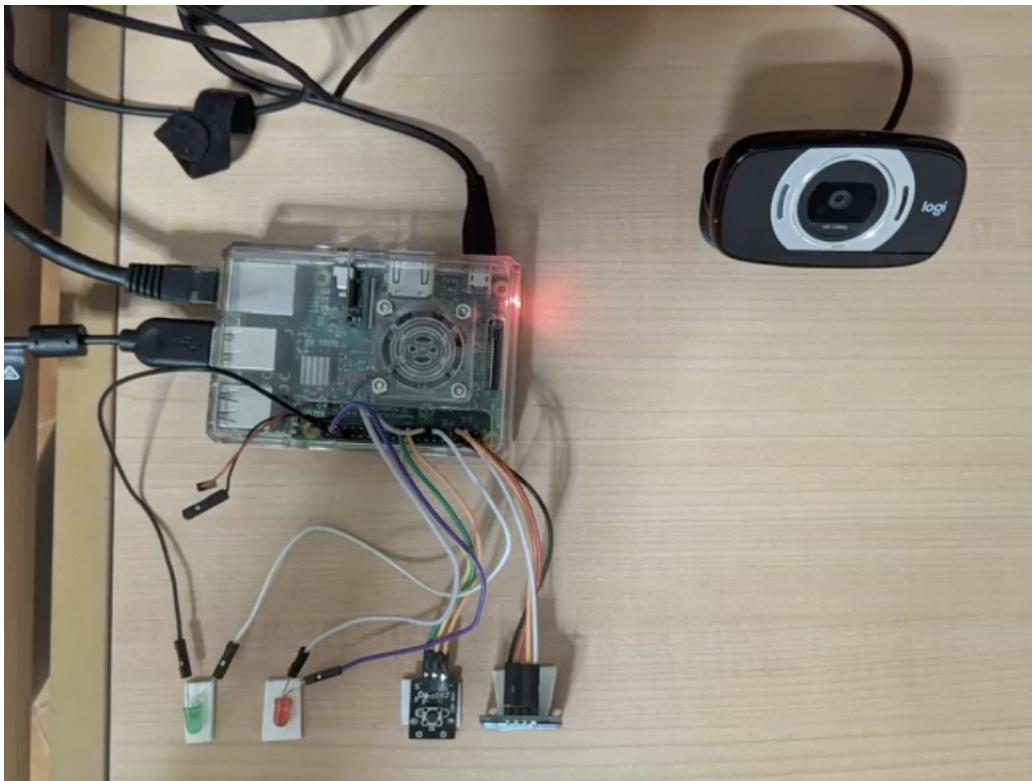


Figura 3: Testing

6 Cablaggio

Oltre alla parte software, è presente anche la parte di hardware e in particolare i circuiti legati al **SoC** (il Raspberry) che stiamo utilizzando. Nella figura sottostante è mostrato il cablaggio e sono evidenziate le connessioni tra le periferiche e la scheda.

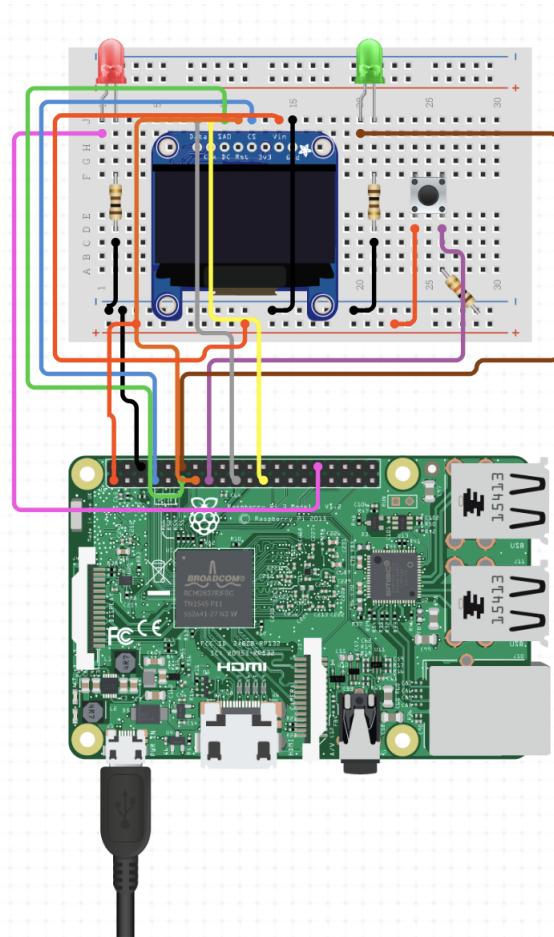


Figura 4: Circuito sulla breadboard

6.1 Bredboard e Collegamenti

I collegamenti di colore rosso hanno tensione 5V e quelli neri sono il collegamento a GND. In questo schema si può notare che lo schermo non è collegato tramite protocollo **I2C** perché possiede più di 2 cavi totali oltre a **VCC** (5 Volt) e **GND** (Ground). Questa periferica però supporta, come il Raspberry, una vasta gamma di protocolli ed in fase di testing per semplicità si è scelto di utilizzare I2C (per ridurre al minimo la quantità di cavi, senza nessuna perdita di prestazioni perché erano pochi gli elementi a schermo da stampare).

Sono poi presenti sulla Bredboard i **LED** rosso e verde, collegati tramite una resistenza da 180 OHM, perché lavorando tra i 10 e i 20 milliampercere si fulminerebbero se collegati direttamente a una tensione di 5V e un amperaggio di 1.5 A rilasciati dal Raspberry. I LED indicano il login avvenuto con successo (verde) o un login errato (rosso) per qualunque operazione di login a terminale o elevazione di privilegi sulla macchina, compreso lo switch tra un utente ed un altro.

Il **bottone** presente alla destra dello schermo, invece ha il compito di scattare una foto tramite la fotocamera esterna o integrata agganciata al Raspberry. Una volta che l'utente amministratore del sistema lacia il comando *signup*, al click del bottone viene scattata la foto e registrata nel database

delle foto utente, per poi essere utilizzata come metodo di paragone al prossimo login.

6.2 Webcam

Grazie alla forte modularità del Raspberry, è possibile utilizzare qualunque tipo di webcam. Disponendo di 4 porte **USB 2.0**, ma anche di un **cavo flessibile**, si può infatti replicare la configurazione del progetto con hardware di diverso tipo, anche in base al budget. Sono disponibili infatti dei moduli webcam ad infrarossi (**IR**) che permetterebbero all' Raspberry di autenticare l'utente anche in assenza di luce. Nel caso non si voglia adottare un cavo, è possibile una implementazione del progetto tramite l' emulazione di una webcam, utilizzando uno streaming video in ingresso, ad esempio utilizzando applicativi terzi come **DroidCam**.



Figura 5: Webcam con cavo USB



Figura 6: Webcam con cavo Flex

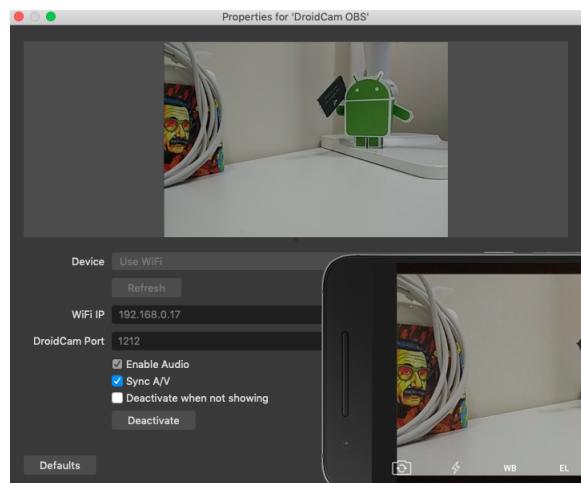


Figura 7: Emulazione Webcam tramite DroidCam

6.3 I2C Oled Display

In fase di testing è stato utilizzato un display oled, nel quale vengono stampate informazioni di sistema quando il Raspberry è in “idle”, ovvero non deve autenticare o registrare nessuno. In questa fase appare a schermo il carico attuale della CPU, della RAM e la capacità del disco. Come primo record appare l’indirizzo IP del Raspberry, che in questo caso è l’indirizzo della rete interna di Docker, essendo l’applicativo all’ interno di un container.

In fase di registrazione lo schermo invece notificherà l’utente riguardo al corretto salvataggio della foto sul sistema.

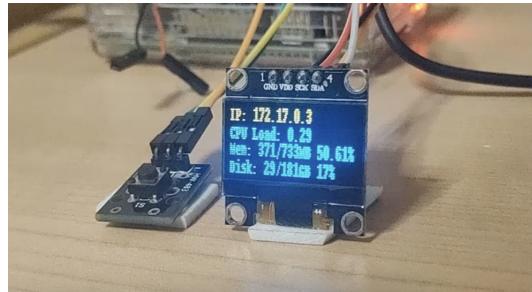


Figura 8: Stato del sistema in idle

Il supporto del protocollo I2C è garantito grazie ad un controller e a dei pin (General Purpose Input Output) che sono presenti sul Raspberry. È possibile collegare **SCK** e **SDA** dello schermo direttamente a questi pin. In particolare il pin GPIO 2 per il **Serial Data** e 3 per il **Serial Clock**.

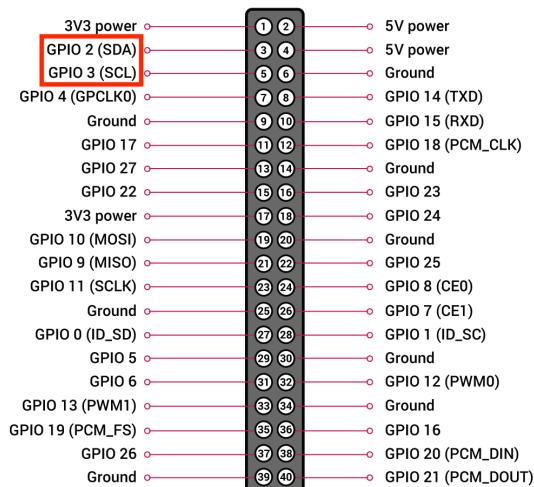


Figura 9: Pinout Raspberry PI 3



Figura 10: Pinout Display

7 Struttura

7.1 Case

Di seguito è presentata la struttura del contenitore (Case) del Raspberry, senza innesto della web-cam. Il modello è stato progettato con il software di modellazione 3D **Rhinoceros** e successivamente stampato in **PLA** nero (materiale plastico) con una stampante 3D.
Il case è diviso in due parti, la parte inferiore contiene il Raspberry e la parte superiore ha l'alloggio per lo schermo e il bottone.



Figura 11: Case

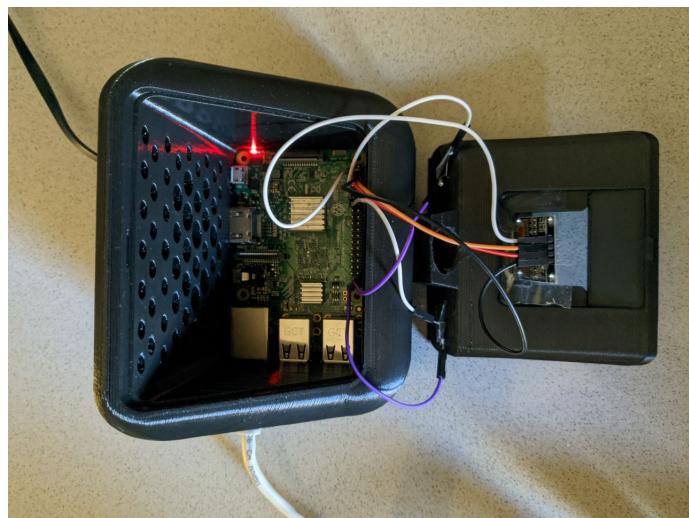


Figura 12: Cablaggio

Il bottone è agganciato ad un ulteriore pezzo stampato in 3D di pochi cm che preme il bottone fisico sottostante inserito in un incastro sulla parte anteriore del case. Lateralmente è possibile vedere l'estruzione effettuata nel case per le porte USB e l'ingresso del cavo RJ-45. Sul lato posteriore invece è presente l'estruzione per la porta micro usb. Nelle immagini sottostanti

si può vedere la struttura del case da varie angolazioni. Da notare sono i fori in Figura 14, i quali permettono una maggiore dissipazione perché posti nella zona del processore del Raspberry. Invece in Figura 13 i fori permettono di utilizzare meno plastica e rendono la struttura più leggera e comunque favoriscono uno scambio d'aria con l'esterno.



Figura 13: Vista da dietro



Figura 14: Vista da sotto



Figura 15: Vista laterale



Figura 16: Vista frontale

7.2 3D Modeling

La modellazione 3D vede come base un rettangolo della grandezza del Raspberry, nel quale sono applicati fori per avvitare sulla struttura la scheda. Nelle figure sottostanti è presente l'assonometria e l'esploso del progetto.

Si può notare la divisione tra parte superiore e inferiore, che si collegano tramite un incastro a scorrimento.

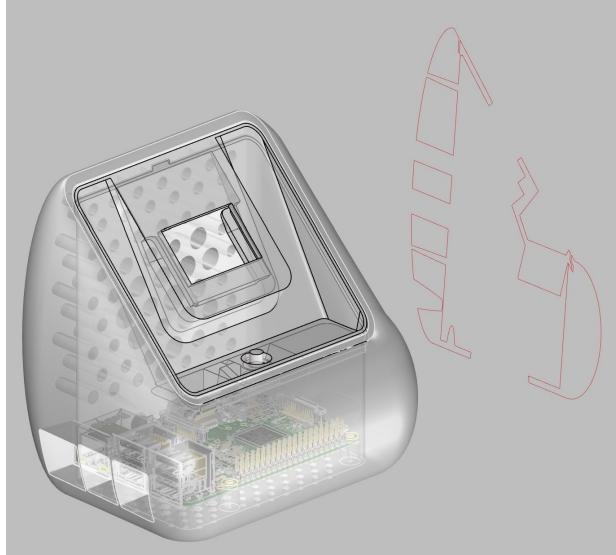


Figura 17: Assonometria

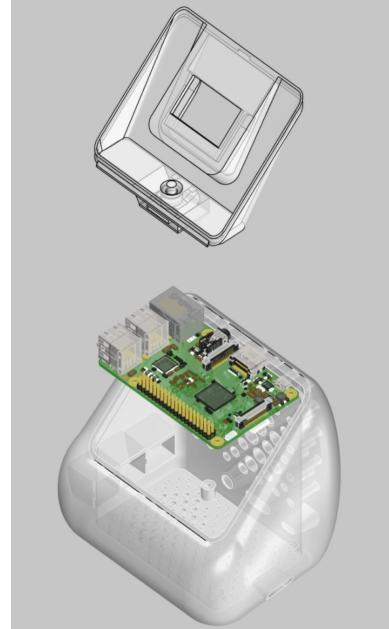


Figura 18: Esploso

Una volta finita la progettazione, viene esportato il progetto in un file **STL** (che si può trovare nella repository online). Il file viene poi caricato su un software di “slicing”, nel nostro caso è stato utilizzato **Cura**.

Dopo aver impostato i parametri fondamentali di stampa come la densità (infill) e la velocità di stampa, viene prodotto un ulteriore file **GCODE** che avrà il compito di comunicare alla stampante 3D i vari step di stampa del file.

Il materiale plastico della stampa è stato scelto per la riduzione di costi, ma la struttura può essere stampata anche in materiali più gommosi e resistenti al calore come il **TPU** o più rigidi come il **PETG**.

Considerando parametri come il nozzle di 0.4mm, velocità di stampa a 80 mm/s e infill al 15 %, il tempo totale di stampa è di circa di 15 ore. La stampante utilizzata è una **Artillery Genius** con piatto di stampa di 220*220*250mm.

7.3 Bill of Material (BOM)

La lista totale delle componenti del progetto è riportata di seguito

- Raspberry PI 3
- Cavo Ethernet - RJ 45
- Cavo di alimentazione 5V - Micro USB
- Webcam
 - Webcam USB
 - Modulo Camera
- Schermo I2C Oled 128x64
- Bottone con resistore - HW483
- 2 Led
- 11 Cavi per Arduino

7.4 Repository e Sorgenti

Tutto il progetto è consultabile, compresi i sorgenti

- **Repository:** <https://github.com/FlippaFloppa/Linux-Face-Login>
- **Container Registry:** <https://hub.docker.com/r/blessedrebus/debian-pam>

7.5 Risorse e Documentazioni

Sono state utilizzate le seguenti risorse:

7.5.1 Documentazioni PAM

- <https://github.com/devinaconley/pam-facial-auth>
- <https://github.com/beatgammit/simple-pam>
- <https://ben.akrin.com/2-factor-authentication-writing-pam-modules-for-ubuntu/>
- <https://wiki.archlinux.org/title/PAM>

7.5.2 Documentazioni OpenCV

- <https://realpython.com/face-recognition-with-python/>
- <https://pysource.com/2021/08/16/face-recognition-in-real-time-with-opencv-and-python/>

8 Conclusioni

L'autenticazione facciale è stata implementata con successo all'interno di Linux. Producendo container sia per **ARM** che per **CISC** si consente quindi l'utilizzo dell'applicativo su qualunque sistema disposto di una webcam o che può ricevere un flusso video. L'autenticazione sarà veloce e senza l'onere della gestione di una password.

Il sistema presenta delle vulnerabilità che potranno essere risolte in una versione futura andando a lavorare sugli script di registrazione e autenticazione, tenendo comunque a mente che una foto non è il mezzo più sicuro per autenticarsi. I dispositivi, come i telefoni, che implementano l'autenticazione facciale usano strumenti più precisi come gli infrarossi o i sensori dell'iride che garantiscono una maggiore precisione e sicurezza. Tuttavia il sistema appena descritto, seppur più debole si dispone ad accettare "moduli" aggiuntivi, come delle API di accesso a sensori di alcuni moduli hardware (ad esempio impronte digitali o utilizzo del *lidar*).

Sulla base di questo progetto si possono costruire varie estensioni come un sistema di Identity Federation basato su riconoscimento facciale, implementabile anche via web.