Immagine che contiene testo, segnale, grafica vettoriale

Descrizione generata automaticamente

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Documentazione**  Unlocker IoT   |  |  | | --- | --- | | Riferimento |  | | Versione | 1.0 | | Data | 15/05/2022 | | Destinatario | Prof. P. Ritrovato  Prof.ssa L. Fotia | | Presentato da | Montefusco Alberto  Spina Gennaro  Oskar Szuba | |

Sommario

[Sommario 2](#_Toc530825397)

[Responsabilità 4](#_Toc530825396)

[Introduzione 5](#_Toc530825398)

1.1 Scopo del Sistema 5

1.2 Panoramica ............................................................................................................................................ 9

Sistema Proposto ............................................................................................................................................ 9

2.1 Requisiti funzionali ............................................................................................................................. 10

2.2 Requisiti non funzionali .................................................................................................................... 10

Strumenti Software e Hardware utilizzati................................................................................................. 20

3.1 Dispositivi Hardware ........................................................................................................................ 12

3.2 Micropython vs C .............................................................................................................................. 12

3.3 GUI ...................................................................................................................................................... 12

3.4 Mock-up .............................................................................................................................................. 12

Responsabilità

|  |  |
| --- | --- |
| **Artefatto** | **Autori** |
|  |  |
|  |  |

1. Introduzione
   1. Scopo del Sistema

La realizzazione di Unlocker IoT ha come obiettivo quello di facilitare l’utente nell’autenticazione digitale ma anche di assicurare la sicurezza attraverso la sua funzione di password manager. Con questo sistema l’utente non dovrà fare altro che collegarlo ad un dispositivo (Computer, Smartphone, …) tramite Bluetooth e, una volta che si è autenticato tramite il lettore di impronte digitali di Unlocker IoT, le credenziali scelte saranno inviate automaticamente nei campi che l’utente ha selezionato per la verifica. Quindi, tra gli obiettivi che il sistema propone di assicurare abbiamo: la sicurezza, l’efficienza, la portabilità e la versatilità.

* 1. Panoramica

Nei seguenti capitoli andremo ad analizzare i requisiti funzionali e non funzionali che il sistema deve necessariamente assicurare (Capitolo 2). Inoltre, verranno descritti gli strumenti Software utilizzati per la realizzazione dell’interfaccia grafica che permette il setup del sistema IoT da parte dell’utente, i dispositivi Hardware aggiuntivi integrati nella scheda ESP32 ed i linguaggi utilizzati per la programmazione IoT (Capitolo 3).

1. Sistema Proposto

Il sistema che proponiamo di realizzare è un dispositivo IoT che permette ad un utente di autenticarsi in un sito web oppure in un’applicazione per smartphone in pochi e semplici passi, inoltre, l’utente non dovrà ricordare tutte le password e gli username che possiede poiché Unlocker IoT funge anche da password manager: al suo interno saranno memorizzate un insieme di credenziali (cifrate) che l’utente potrà reperire tramite semplici steps.

Spieghiamo ora il funzionamento del sistema proposto. L’utente potrà accendere Unloacker premendo un piccolo bottone posto sul lato. All’accensione, l’utente visualizzerà su un piccolo schermo OLED una lista di nomi (i siti web o le applicazioni a cui è registrato con username e/o password). L’utente sceglierà un nome dalla lista e avrà accesso a quelle credenziali solo dopo essersi verificato tramite il lettore di impronte digitali posto sul sistema.

Una volta che la verifica avrà avuto esito positivo, l’utente collegherà il sistema ad un dispositivo (Computer o Smartphone) tramite il Bluetooth: il collegamento avverrà solo quando l’utente avrà inserito un codice di sicurezza sul suo Computer o Smartphone; nei prossimi accoppiamenti il codice non sarà richiesto fin quando l’utente non cancellerà l’autenticazione Bluetooth sul dispositivo accoppiato.

Successivamente, l’utente, dopo aver selezionato un campo per l’inserimento dell’username o della password, cliccherà sul primo bottone, posto di fianco lo schermo OLED e automaticamente (tramite una simulazione della tastiera mediante Bluetooth) il sistema invierà la stringa decifrata nel campo selezionato. Per l’inserimento della password la procedura è analoga ma l’invio avverrà solo quando l’utente pigerà il secondo bottone posto sotto al primo, di fianco lo schermo OLED.

Dopo che le credenziali sono state inviate, il sistema automaticamente mostrerà nuovamente la lista di nomi e, per accedere alle credenziali, l’utente dovrà nuovamente verificarsi tramite fingerprint.

Nel prossimo capitolo spiegheremo come avverrà il setup del sistema IoT tramite GUI realizzata in Java.

* 1. Requisiti funzionali

|  |  |  |
| --- | --- | --- |
| **Identificativo** | **Priorità** | **Descrizione** |
| **RF[1]** |  |  |
| **RF[2]** |  |  |
| **RF[3]** |  |  |
| **RF[4]** |  |  |
| **RF[5]** |  |  |
| **RF[6]** |  |  |
| **RF[7]** |  |  |
| **RF[8]** |  |  |
| **RF[9]** |  |  |
| **RF[10]** |  |  |
| **RF[11]** |  |  |
| **RF[12]** |  |  |
| **RF[13]** |  |  |

* 1. Requisiti non funzionali

|  |  |  |
| --- | --- | --- |
| **Identificativo** | **Priorità** | **Descrizione** |
| **RNF[1]** | 5 | Il sistema deve garantire la sicurezza delle credenziali memorizzate |
| **RNF[2]** | 5 | Il sistema deve collegarsi ad un dispositivo e inviare le credenziali entro pochi secondi |
| **RNF[3]** | 5 | Il sistema deve essere compatibile con tutti i dispositivi che supportino una connessione Bluetooth e una tastiera virtuale |

1. Strumenti Software e Hardware utilizzati
   1. Dispositivi hardware

Il sistema dovrà essere composto dai seguenti componenti hardware:

* **Scheda ESP32 TTGO**, per la gestione del sistema, dei sensori esterni e dell’interfacciamento con i dispositivi su cui sarà necessario inserire le credenziali di accesso (Computer, Smartphone ecc…). La scheda è dotata di un display OLED integrato da 0,96’’, due pulsanti programmabili e uno per il reboot della scheda, un connettore di alimentazione per batterie e una porta type-C per il collegamento con i dispositivi e l’alimentazione della scheda stessa.

[Immagine che contiene elettronico, circuito

Descrizione generata automaticamente](https://www.amazon.it/gp/product/B07VNG9D52/ref=ppx_yo_dt_b_asin_image_o03_s00?ie=UTF8&psc=1)

* **Capacitive fingerprint sensor**, attraverso il quale sarà possibile effettuare l’autenticazione dell’utente. Il sensore garantirà l’unicità dell’utente a cui apparterrà il dispositivo stesso.

[Immagine che contiene testo

Descrizione generata automaticamente](https://www.amazon.it/Capacitive-Fingerprint-Raspberry-360-degree-Recognition/dp/B07W84DCT7/ref=sr_1_1?crid=1ZKMZ8VO325YJ&keywords=capacitive+fingerprint+sensor&qid=1649605922&sprefix=capacitive+fingerprint+sensor%2Caps%2C114&sr=8-1)

* **Batteria LiPo** 1 cella da 500 mAh 3,7 V permetterà di avere un dispositivo portatile e indipendente.

[Immagine che contiene testo

Descrizione generata automaticamente](https://www.amazon.it/gp/product/B082NZGW3V/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1)

* 1. Micropython vs C

La scheda ESP32 TTGO può essere programmata utilizzando due linguaggi di programmazione differenti: **micropython** e **C**. Ognuno dei due presenta dei vantaggi e delle limitazioni che andremo ad analizzare nel seguente paragrafo.

Nel caso in cui verrà utilizzato **micropython**, per la programmazione del dispositivo IoT, saranno integrate le seguenti librerie:

* [**MicroPythonBLEHID**](https://github.com/Heerkog/MicroPythonBLEHID)([**LINK**](https://github.com/Heerkog/MicroPythonBLEHID)): simulazione di una tastiera bluetooth su diversi dispositivi;
* [**AdafruitFingerprint**](https://docs.circuitpython.org/projects/fingerprint/en/latest/_modules/adafruit_fingerprint.html)([**LINK**](https://docs.circuitpython.org/projects/fingerprint/en/latest/_modules/adafruit_fingerprint.html)): gestione del sensore di impronte digitali;
* **[Microbit\_ssd1306](https://github.com/micropython/micropython/blob/master/drivers/display/ssd1306.py)** ([**LINK**](https://github.com/fizban99/microbit_ssd1306)): gestione di immagini e testo su uno schermo OLED;

Altrimenti, le librerie adottate, per la programmazione del dispositivo mediante il linguaggio **C**, saranno le seguenti:

* **ESP32-BLE-Keyboard** ([**LINK**](https://github.com/T-vK/ESP32-BLE-Keyboard)): simulazione di una tastiera bluetooth su diversi dispositivi;
* **DFRobot-FingerPrint-Sensor** ([**LINK**](https://github.com/akarsh98/DFRobot-FingerPrint-Sensor)): gestione del sensore di impronte digitali;
* **TFT\_eSPY** (**libreria di Arduino IDE**): gestione delle scritte su uno schermo OLED;
* **JPEGDecoder** (**libreria di Arduino IDE**): gestione di immagini su uno schermo OLED.

|  |  |  |
| --- | --- | --- |
|  | **Micropython**  (MicroPythonBLEHID) | **C**  (ESP32-BLE-Keyboard) |
| **Pro** |  | * Compatibilità con tutti i dispositivi più diffusi sul mercato; * Possibilità di simulare ogni tasto della tastiera, compresi quelli dedicati al controllo del volume |
| **Contro** | * Incompatibilità con dispositivi IOS, MacOS, IpadOS; * Impossibilità nel simulare la pressione dei tasti speciali (option, control, WIN, command, etc…) |  |

|  |  |  |
| --- | --- | --- |
|  | **Micropython**  (AdafruitFingerprint) | **C**  (DFRobot-FingerPrint-Sensor) |
| **Pro** | * Gestione di immagini e testo con la stessa libreria | * Le immagini possono essere a colori * Non è necessario un ulteriore step di conversione delle immagini in testo |
| **Contro** | * Le immagini devono essere prima convertire in testo con un’apposita app esterna * Le immagini possono essere solo monocromatiche | * Sono accettate solo immagini con estensioni Jpeg * È necessario il supporto di un ulteriore libreria esterna per la gestione dei testi |

Dalle analisi effettuate e dalle varie ricerche su varie documentazioni ricercate in Internet possiamo notare che il linguaggio C ha vari vantaggi rispetto ad una programmazione in micropython, in quanto: con il linguaggio C avremo maggiore compatibilità con i vari dispositivi presenti sul mercato e maggior efficienza dal punto di vista delle prestazioni del codice. D’altro canto, scrivere un codice in C ha una difficoltà maggiore rispetto ad un codice scritto in micropython.

Essendo l’obiettivo del progetto realizzare un dispositivo IoT sicuro, efficiente e versatile, la nostra scelta è incline nell’utilizzare, come linguaggio di programmazione per ESP32 TTGO, il C.

* 1. GUI

L’interfaccia grafica da realizzare sarà un’applicazione desktop che permetterà all’utente di effettuare il setup del dispositivo IoT, in particolare, l’utente potrà:

1. aggiungere nuove credenziali;
2. eliminare le credenziali;
3. modificare le credenziali già esistenti;
4. configurare il sensore di impronte digitali aggiungendo o rimuovendo un’impronta.

Per accedere all’applicazione, l’utente dovrà inserire un codice di sicurezza che sarà collegato con il dispositivo IoT.

La GUI sarà realizzata interamente in Java (sia back-end che front-end), in particolare per il front-end utilizzeremo come tecnologie JavaFX e CSS.

* 1. Mock-up

Di seguito sono mostrati dei mock-up di alcune funzionalità dell’interfaccia grafica e un prototipo finito del sistema IoT in questione (la scocca del dispositivo sarà in polimero realizzato tramite stampa 3D).

Immagine che contiene testo, elettronico

Descrizione generata automaticamente

Fig. 1 - Sistema IoT

Immagine che contiene testo, elettronico, schermo, screenshot

Descrizione generata automaticamente

Fig. 2 - Autenticazione

Immagine che contiene testo, screenshot, elettronico, schermo

Descrizione generata automaticamente

Fig. 3 - Dashboard

Immagine che contiene testo, screenshot, elettronico, schermo

Descrizione generata automaticamente

Fig. 4 – Dashboard (con side menu)

Immagine che contiene testo, monitor, screenshot, schermo

Descrizione generata automaticamente

Fig. 5 - Aggiungi credenziali

Immagine che contiene testo, screenshot, monitor, schermo

Descrizione generata automaticamente

Fig. 6 - Modifica credenziali