

Fingerprint detection, storing and display on bare-metal

*Membri Team:*

**Iraci Davide**

**Montemaggiore Andrea**



***Relatore: Prof. Peri Daniele – Sistemi Embedded***

Sommario

1. Introduzione
   1. Scopo del progetto
2. Hardware in uso
   1. Raspberry Pi 4B
   2. FTDI FT233RL
   3. Fingerprint Module FZ2904
3. Protocolli e metodi di comunicazione utilizzati

3.2 I2C

1. Configurazione dell’ambiente di sviluppo
   1. pijFORTHos
   2. Picocom
2. Sviluppo dell’applicazione
   1. Configurazione dell’UART
   2. Comunicazione con modulo FZ2904
   3. Display e HDMI

6. Possibili migliorie

## Introduzione

Il corso di Laurea Magistrale in Ingegneria Informatica presso l’Università degli Studi di Palermo è articolato in due anni composti da oltre 10 materie. Sistemi Embedded è una materia del primo anno del corso di laurea, l’esame della materia prevede lo sviluppo di un’applicazione embedded che deve essere consegnata al Professore insieme alla documentazione della suddetta, oltre a ciò i candidati devono svolgere una prova orale sugli argomenti del corso.

Argomenti principali del corso:

• Embedded systems architectures

• ISA level programming

• Machine language and assembly (ARM)

• BCM2835/7 Architecture

• STM32-F446 architecture

• Forth programming

• Buses (I2C, SPI, UART e USART)

### 1.1 Scopo del progetto

L’applicazione da sviluppare può essere sviluppata su qualunque hardware a scelta dello studente o del team purché rispetti il requisito che sia sviluppata su un ambiente specifico, ovvero pijFORTHos. Il documento è composto da una parte iniziale dove vengono specificati e analizzati i componenti hardware utilizzati, oltre a ciò verranno anche descritti i software, l’ambiente di sviluppo e i protocolli di comunicazione adoperati.

Lo scopo del progetto è quello di permettere il riconoscimento delle impronte da più utenti, il loro salvataggio in memoria in caso di nuova registrazione. Oltre a ciò il team ha introdotto una interfaccia video con l’utente, permettendo la visualizzazione dell’impronta, opportunamente offuscata con una schermata di caricamento in fase di attesa. Infine sarà permesso all’utente che utilizza il sensore, tramite un menu mostrato a video, decidere se effettuare il login, registrare una nuova impronta, o semplicemente mostrare a video una qualunque impronta scannerizzata in quel momento.

Per via dell’hardware utilizzato il team ha dovuto provvedere anche all’abilitazione di un’altra UART oltre a quella già utilizzata in fase di comunicazione con l’hardware scelto, anche questa parte è stata discussa nel documento.

## Hardware

### 2.1 Raspberry Pi 4B

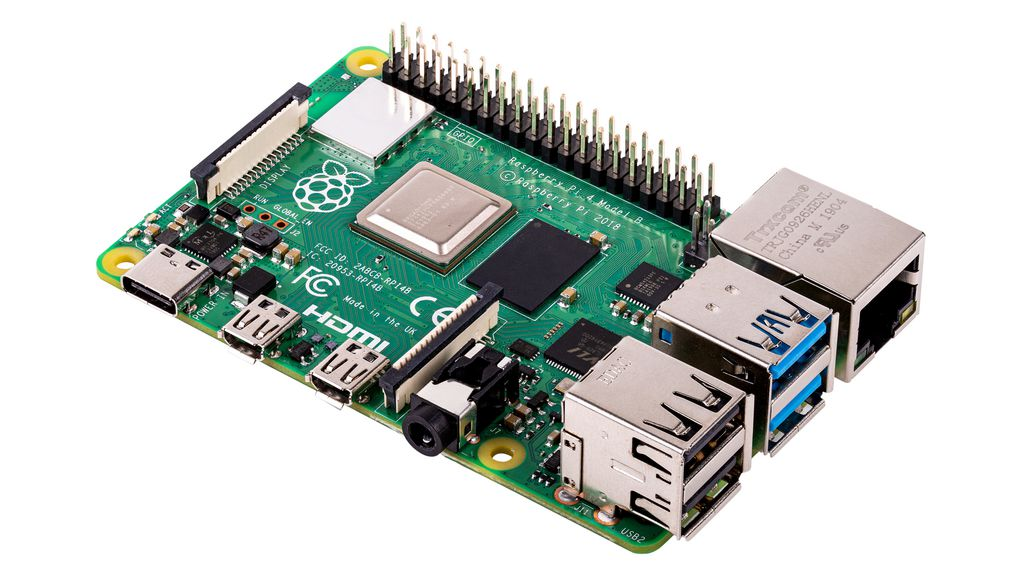
Il team dopo un’approfondita analisi ha scelto come macchina target il Raspberry Pi 4b.

Il Raspberry Pi è un single-board computer sviluppato nel Regno Unito dalla Raspberry Pi Foundation. La presentazione al pubblico è avvenuta il 29 febbraio 2012.

La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux o RISC OS. È stato concepito un sistema operativo appositamente dedicato, chiamato Raspbian.

Il Raspberry Pi 4 Model B, immesso sul mercato nel giugno 2019, monta un Broadcom BCM2711 quad-core a 1.5 Ghz e offre molteplici novità rispetto ai predecessori, di seguito è inclusa la lista delle specifiche:

* Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
* 1GB, 2GB or 4GB LPDDR4-3200 SDRAM (il modello in uso ne possiede 2)
* 2.4 GHz e 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
* Gigabit Ethernet
* N.2 porte micro HDMI con supporto al 4K
* N.2 porte USB 3.0 + N.2 porte USB 2.0

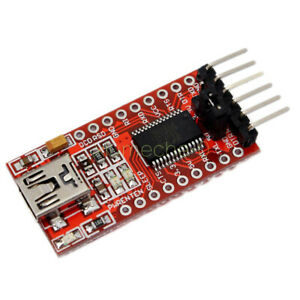


*Figura 1. Raspberry Pi 4b*

### 2.2 FTDI 232RL

La "USB to Serial Adapter with FT232RL" è un utilissimo strumento che permette di collegare i PC con qualsiasi sistema a microcontrollore attraverso la porta USB. Utilizza il famosissimo integrato FT232RL della FTDI, dotato di buffer in ricezione da 128 byte e buffer in trasmissione da 256 byte che garantiscono robustezza in trasmissioni ad alta velocità fino a 3Mbaud/s. Oltre ai segnali TX e RX, sono presenti anche le linee CTS, RTS e le altree linee di handshaking. UART sta per Universal Asynchronous Receiver-Transmitter ed è un dispositivo che trasforma flussi di bit da fa formato seriale asincrono a parallelo e viceversa.

Come già detto presentano diverse linee di handshaking e anche servizi per il controllo del flusso e degli errori.

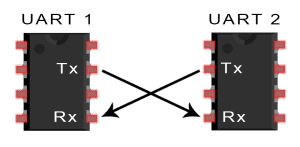


*Figura 2. FTDI FT232RL*

### 

Per configurare un dispositivo all’utilizzo della porta UART è necessario collegare il pin RXD del dispositivo in uso al pin TXD del FT232RL e viceversa.

Quando un byte viene trasmesso esso è preceduto e seguito da uno start e stop bit.



*Figura 3. Configurazione del cablaggio tra UART*

Inoltre è da specificare che le due UART devono avere lo stesso BAUD rate, ovvero una specifica frequenza alla quale leggono i bit inviati. Il Baud rate ha un minimo di 9600 bps fino ad un massimo di 115200 bps.

### 2.3 Fingerprint Module FZ2904

Il modulo in questione permette il riconoscimento, salvataggio ed eventuale cancellazione delle impronte digitali di una o più individui. Il sensore in questione è molto valido in quanto permette il riconoscimento di un’impronta in un range di 360° quando posta sul lettore. Quando attivo il sensore accende un led rosso, che può essere programmato e settato anche per altri scopi. Inoltre il sensore è in grado automaticamente di settarsi in base alle condizioni climatiche in quanto è in grado di riconoscere impronte che presentano segni di variazioni climatiche (es impronte bagnate per via di pioggia, impronte secche per via del caldo o del freddo). Inoltre, a disposizione dei programmatori o più in generale dell’utilizzatore vi è la possibilità di impostare un livello di sicurezza crescente che va da 1 a 5. Il processo di riconoscimento include due parte: l’enrolling, ovvero eseguire la scannerizzazione e il matching, ovvero riconoscere l’impronta, quest’ultima fase può essere impostata per 1:1 o 1:N, cioè se corrisponde esattamente a una impronta o di fare il matching tra tutte quelle disponibili. Quando un utente si registra per la prima volta, quindi effettua l’enrolling di una nuova impronta deve scannerizzare due volte l’impronta, il sensore ricava due immagini e da esse genera un template che immagazzina nei 72kb di memoria disponibili. Similmente quando un utente effettua il matching il sensore genera un template da confrontare con gli altri in memoria, come già detto 1:1 significa fare il match con uno specifico template e 1:N invece con tutta la libreria di template salvate.

In entrambi i casi il sistema darà come risultato il risultato, successo o fail.

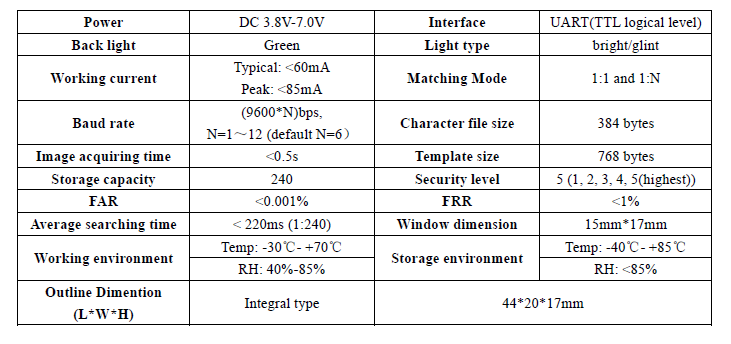


*Figura 4. Fingerprint Module FZ2904*

Il modulo si può connettere tramite interfaccia UART a un qualunque PC o come nel nostro caso al Raspberry Pi 4b. La tecnologia al giorno d’oggi è usata in qualunque ambito, dagli Smartphone per il riconoscimento del proprietario, all’entrata delle palestre per il riconoscimento degli abbonati o ancora all’apertura delle portiere dei veicoli e per altri infiniti usi e applicazioni. Il modulo è fornito di 6 cavi da connettere tramite UART, sia per lo sviluppo che per l’uso normale tuttavia è necessario collegarne 4: GND, TXD, RXD, Power Input.

Il sensore presenta svariate caratteristiche oltre a quelle già elencate, è possibile ottenere una bitmap di un template di un’impronta, funzione implementata in fase di sviluppo; inoltre è possibile modificare il baud rate a piacimento, anche questa funzione è stata implementata.

Infine il sensore presenta una risposta con un pacchetto di ACK in caso sia di successo che di errore, in alcuni casi specificando anche di che tipo di errore si tratta.



*Figura 5. Parametri principali del sensore*

## Protocolli e metodi di trasmissione utilizzati

### 3.1 I2C

I2C è un famoso protocollo seriale diventato popolare per numerosi vantaggi. Principalmente è da sottolineare come bastano due linee, dunque due pin, per la comunicazione. E’ possibile individuare due parti in una comunicazione: il master e lo slave. Nel progetto in questione identifichiamo come master il Raspberry, gli slave invece sono molteplici e ognuno di essi ha un indirizzo chiamato slave address specifico per distinguersi.

## 3.2 UART

La UART o Universal Asynchronous Receiver-Transmitter, è un dispositivo hardware che converte flussi di bit di dati da un formato parallelo a un formato seriale asincrono o viceversa.Sono richiesti almeno due segnali digitali: *trasmissione* (TX), segnale di output; *ricezione* (RX), segnale di input. Il pin TX in un dispositivo è collegato al pin RX nell'altro.   
Ogni famiglia di microprocessori ha la sua UART/USART dedicata. Nei [microcontrollori](https://it.wikipedia.org/wiki/Microcontrollore) che la posseggono questa funzione è interna agli stessi.

## Configurazione dell’ambiente di sviluppo

### 4.1 pijFORTHos

pijFORTHos è un interprete FORTH bare-metal per il Raspberry Pi, è basato su Jonesforth-ARM. Per poter usufruire dell’interprete in questione è prima necessario installare Raspbian, un sistema creato ad hoc. Successivamente è necessario sostituire il kernel di Raspbian con quello di pijFORTHos, compiuto quest’ultimo passaggio il sistema è attivo e funzionante.

### 4.2 Picocom

Picocom è un software di emulazione da terminale per sistemi Unix-based. Per avviare picocom è necessario digitare un comando nel terminale Linux:

* picocom --b 115200 /dev/ttyUSB0 --imap delbs -s "ascii-xfr -sv -l100 -c10"

Digitato questo comando, se il dispositivo è funzionante e l’installazione di pijFORTHos è avvenuta correttamente allora l’interprete FORTH sarà attivo.

### 4.3 Minicom

Minicom è uno strumento simile a Picocom che tramite la combinazione CTRL+A+S permette l’invio di file che verranno interpretati da pijFORTHos.

## Sviluppo dell’applicazione

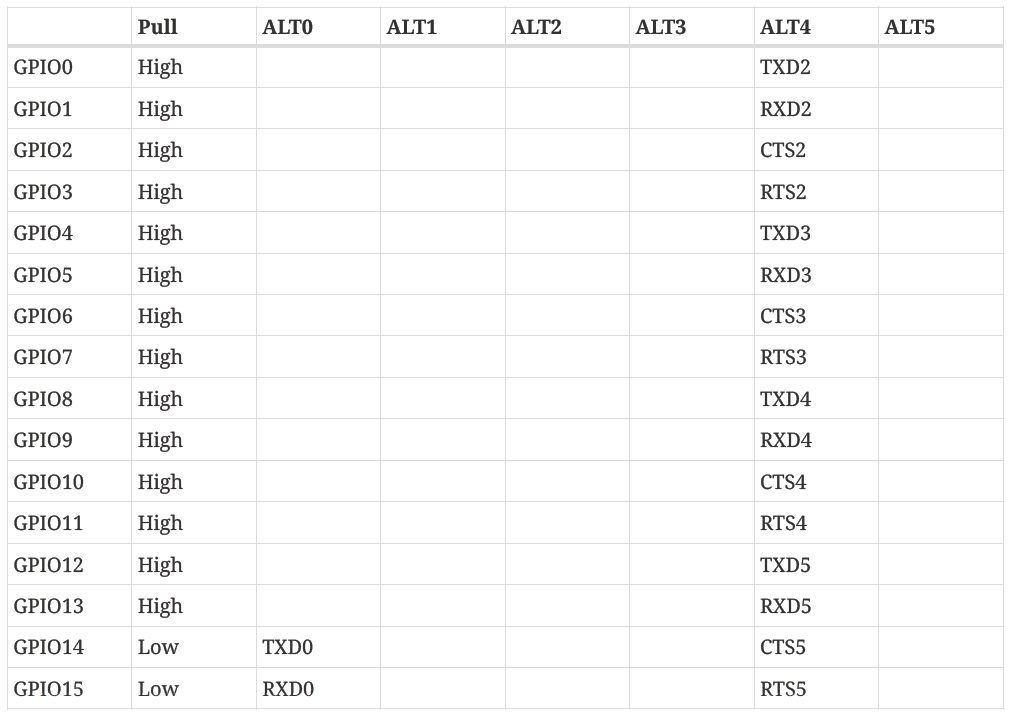
Di seguito vi è spiegato lo sviluppo dell’applicazione che consente la scannerizzazione di impronte digitali con visualizzazione a video del template dell’impronta, inoltre è stato inserito sempre a video un menù interattivo che permette la registrazione, cancellazione e visualizzazione di un impronta. L’applicazione è composta da 4 file:

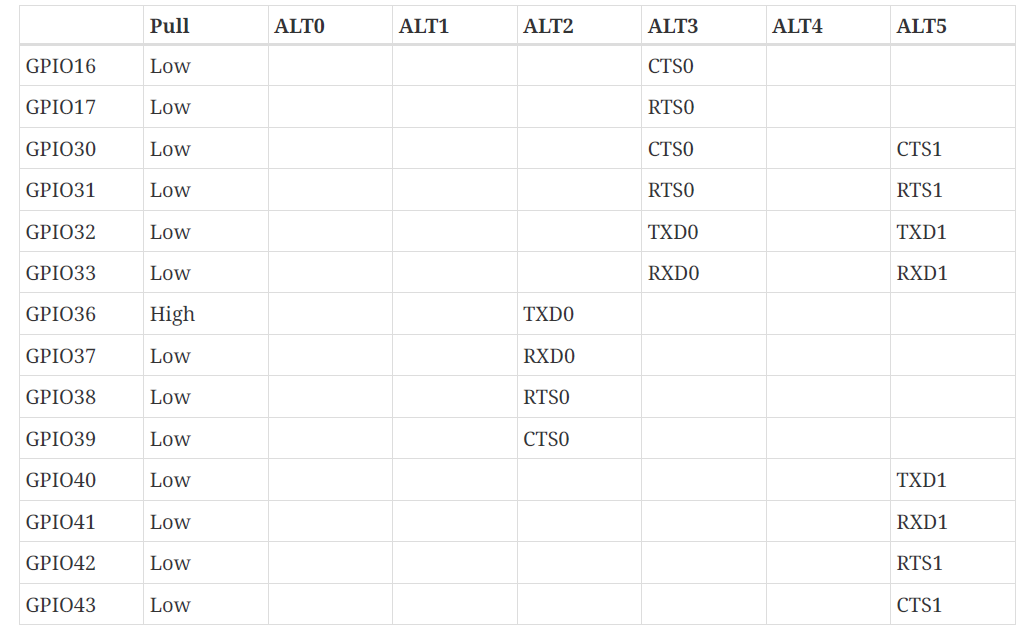
* se-ans.f
* UART.f
* Fingerprint.f
* HDMI.f

### 5.1 Configurazione dell’UART (UART.f)

Il sistema, una volta pronto per poter sviluppare, ha solamente una UART con i suoi corrispettivi pin (GPIO 14-15) abilitati. Il modulo FZ2904 per poter essere utilizzato tramite Raspberry necessita dunque di un collegamento tramite UART, è necessario abilitare un’altra UART dal Raspberry.

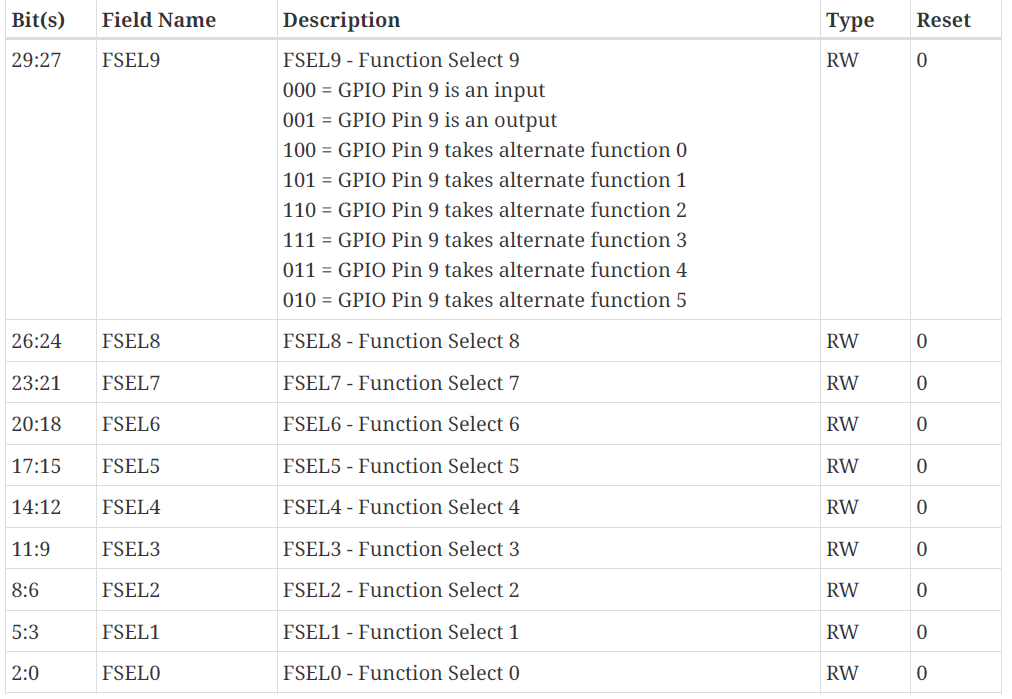
Documentandosi è possibile vedere come il Raspberry possieda 6 UART: una mini e 5 classiche, quindi si è arrivati a scegliere quale pin utilizzare.





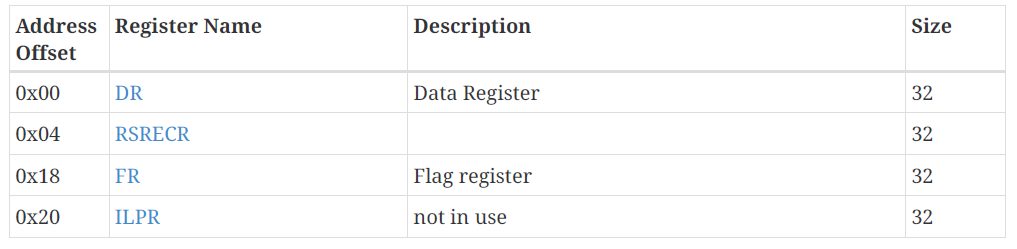
*Figura 6. Tabella delle alternate functions*

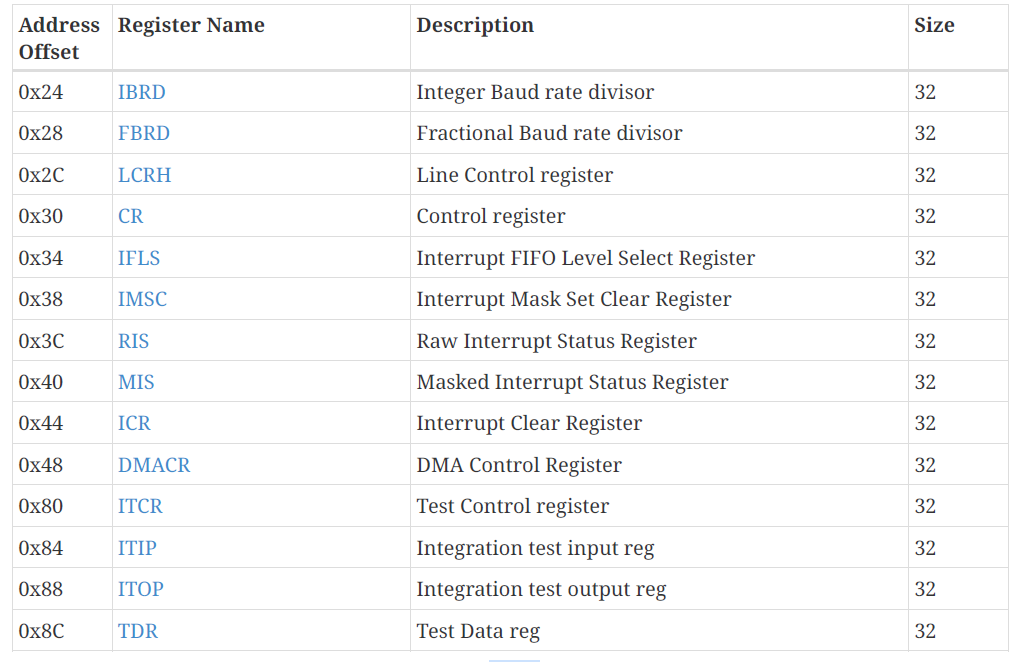
Consultando la tabella qui rappresentata il team ha scelto di utilizzare i GPIO0 e GPIO1 che corrispondono alla coppia di PIN 27-28. Per poter abilitare l’Alternate Function 4 di tali pin è necessario settare dei bit nell’indirizzo di GPFSEL0



*Figura 7. Tabella GPFSEL0*

GPFSEL0 gestisce le alternate function dei GPIO00 fino a 09, consultando tale tabella si è deciso di impostare i primi 6 bit, precisamente dal bit 5 al bit 0 con il seguente ordine “011 011”, ogni trio di bit abilita l’alternate function 4 dei pin 27 e 28 e quindi dei GPIO00 e GIPIO01. Tuttavia per abilitare l’UART non è solamente necessario abilitare l’alternate function nei pin corrispondenti, vi sono anche molteplici registri da dover utilizzare, alcuni per controllo e altri per permettere lettura e scrittura.





*Figura 8. Tabella dei registri UART*

Il team ha dunque definito una HAL inserendo tutti i registri della UART, mettendo nei rispettivi indirizzi i bit necessari per il corretto funzionamento. Successivamente si è anche impostato il baud rate, il clock e sono state create principalmente cinque word: INIT, SERIAL\_READ, READ, WRITE, SERIAL\_WRITE.

La prima word, INIT, è quella da avviare dopo aver inserito tutte le altre definizioni e word nell’interprete FORTH, in quanto tramite alcune maschere abilita le alternate function e i registri necessari per la comunicazione.

SERIAL\_READ e READ sono due word necessarie per la lettura tramite la UART, è stato inserito un timeout per evitare che se non vi fossero dati da poter leggere il sistema andasse in loop. SERIAL\_WRITE e WRITE sono due word che permettono la scrittura sulla UART per poter inviare i pacchetti al sensore.

Allo scopo di rendere il programma il più versatile possibile gli indirizzi della UART sono spostati dei rispettivi OFFSET a partire da PERI\_BASE, una costante contenente l’indirizzo base del Raspberry. Inoltre è stata creata una WORD che permette di modicare il baudrate della UART, rendendo così l’abilitazione di quest’ultima e la modifica del baud rate utile per tutti gli scopi, sia per sensori che richiedono alte velocità sia per quelli che per vari motivi richiedono una bassa velocità di trasmissione.

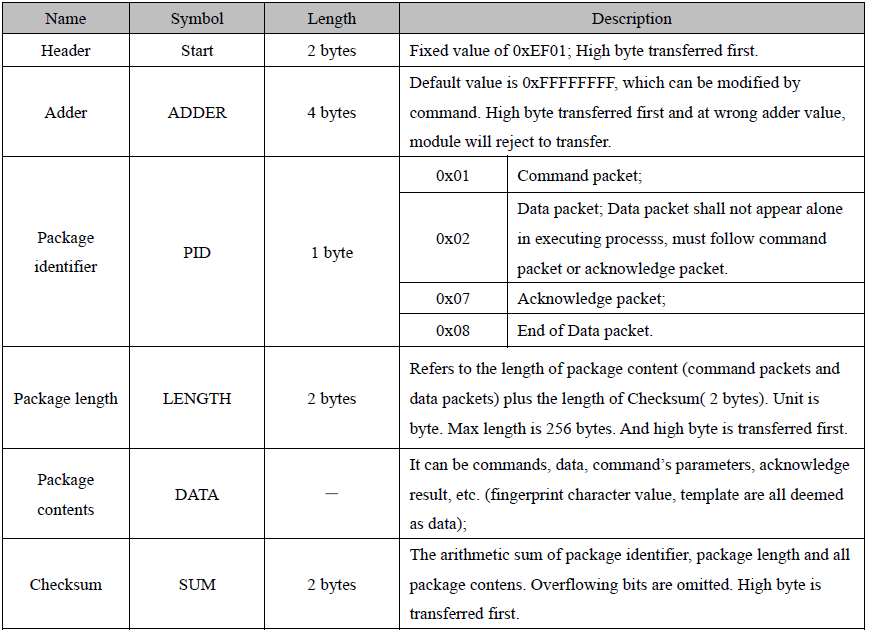
### 5.2 Configurazione del modulo FZ2904 (Fingerprint.f)

Una volta configurata e abilitata la UART, primo passo del progetto, si può passare al ricevere e trasmettere dati dal modulo FZ2904. Il modulo quando riceve un comando lo riceve attraverso un pacchetto ben formato, qui di seguito è rappresentata la composizione:



*Figura 9. Formato dei pacchetti ricevuti e inviati dal sensore*

Già dal formato del pacchetto è possibile notare che è necessario a fine pacchetto calcolare un valore di checksum, quest’ultimo è calcolato sommando tutti i byte ad eccezione di HEADER e ADDER.



*Figura 9. Dettaglio dei componenti del pacchetto.*

I primi byte da inviare per qualunque tipo di comunicazione saranno 0xEF01FFFFFF corrispondenti all’HEADER e all’ADDER, i byte successivi variano da istruzione a istruzione e anche la lunghezza del pacchetto può variare, infine come detto va calcolata la checksum.

Il team dopo un’attenta analisi ha deciso che il modo migliore per gestire i pacchetti da inviare e la loro composizione è quella di creare un array con già inseriti sia l’HEADER che l’ADDER, ad essi vengono aggiunti per i vari comandi (LED\_ON, LED\_OFF, etc) i byte necessari e infine calcolata la checksum che viene anch’essa inserita nell’array.

Per poter sfruttare gli ARRAY è stato necessario creare READ\_ARRAY, READ\_ARRAY\_N, WRITE\_ARRAY, WRITE\_ARRAY\_N. Le prime due word servono rispettivamente per: la prima per poter leggere tutto l’array e l’altra per poter leggere solamente una parte di esso data una lunghezza e posizione in input, lo stesso concetto è applicato per le altre due word che permettono invece la scrittura sull’array.

Quando il modulo riceve un comando, esso risponderà con un pacchetto di ACKNOWLEDGE sia che esso sia corretto sia che non sia corretto o vi siano problemi nell’esecuzione. Dunque per poter identificare i pacchetti è necessario leggere i byte tramite la UART che vengono salvati in un array chiamato PACKTOREC, tramite questo array possiamo gestire gli errori in fase di enrolling, matching, etc, sezione utile per la fase successiva.

Il sistema prevede l’uso di una variabile di appoggio chiamata “TAKE\_COUNTER” che all’avvio, o ogni volta che il sensore registra una nuova impronta, aggiorna il numero di impronte registrate permettendo così l’inserimento di nuove impronte in posizioni non occupate da altre impronte nella memoria del sensore.

Inoltre è possibile effettuare l’upload dal sensore al Raspberry di un immagine, o meglio del template dell’impronta scannerizzata sul momento in formato bitmap. Il sensore invia 288 pacchetti da 139 byte l’uno con 128 byte di immagine e il resto di header del pacchetto, ogni 4 bit si ottiene la misura di un pixel. All’immagine inviata è assente l’header bitmap che il team ha aggiunto in un array dove viene inserita la bitmap a 8 bit, dove in fase di compressione da parte del sensore un byte è diventato 4 bit, che viene opportunamente trasformata a 32 bit potendo così essere poi inviata alla sorgente video, trattato nel prossimo capitolo.

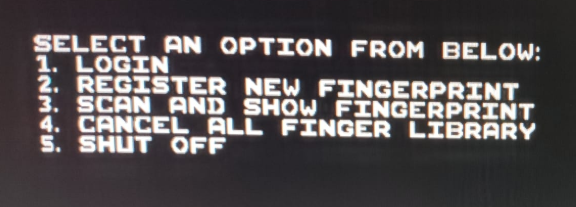
### 5.3 Display e HDMI (HDMI.f)

Come già anticipato precedentemente, a video viene mostrata un'immagine bitmap opportunamente manipolata e adattata già nel file fingerprint.f trattato nel capitolo precedente. L’immagine viene mostrata centrata, tuttavia mentre essa viene caricata sul Raspberry ed elaborata per poter essere mostrata a video vi è una progress bar di loading, mentre la barra si carica vengono eseguite le operazioni di caricamento sul Raspberry, operazione di blurring e invio allo schermo, che una volta riempita mostrerà a video l’immagine.



*Figura 10. Esempio di template*

Sulla bitmap mostrata a video è stata eseguita un'operazione di blurring per motivi di privacy, il team dopo un’analisi ha optato per una maschera di 7x7 pixel con calcolo del valore medio da sostituire per ogni pixel dell’immagine. Per motivi computazionali e di difficoltà si è optato per evitare il padding di colonne e righe, ovvero il problema per il quale non si può effettuare il blurring di elementi di righe e colonne con posizione tale da avere elementi mancanti nella maschera 7x7 , dunque il blurring parte dalla 7° riga e 7° colonna.



*Figura 11. Menu mostrato a video*

A video è stato introdotto un menù interattivo che permette di effettuare le seguenti operazioni:

1. Login
2. Registrazione fingerprint
3. Scannerizzare e mostrare univocamente un impronta
4. Cancellare tutte le impronte
5. Spegnere il dispositivo

Le operazioni 2 e 4 richiedono prima l’autorizzazione dell’admin, dunque il riconoscimento della sua impronta per poter procedere, in particolare l’operazione n°4 non elimina l’admin dal sistema, quest’ultimo è identificato univocamente con l’ID 00. Per quanto concerne l’admin è possibile modificare l’impronta associata all’ID tramite un comando non mostrato a video ovvero INSERT\_ADMIN .

Le varie scelte vengono eseguite tramite comandi da terminale come CH\_1, CH\_2, etc.

L’opzione n°5 spegne lo schermo, per poter riutilizzare il sensore è necessario digitare “START” da terminale.

Per tutte le operazioni, come già detto, è stata creata una GUI per guidare l’utente in tutte le fasi di utilizzo, anche tutti gli errori possibili che il modulo fingerprint può inviare sono opportunamente stati inseriti nella schermata a video, le conferme di operazione correttamente eseguita sono colorate in verde, gli errori invece in rosso. Gli errori inseriti a video sono vari, ad esempio vi è un “No Match” in fase di Login o riconoscimento Admin, un “No Finger On Sensor” quando non viene inserito il dito dopo aver scelto l’operazione da effettuare, fino al più generico “Error!” per tutti gli errori che non sarebbero comunque gestibili dall’utente (errore di pacchetto, eventuali danni interni alla memoria del sensore, etc).

Per quanto concerne il lato di programmazione il tutto è stato gestito tramite l’accensione di pixel con determinati colori, in particolare il verde per la conferma e il rosso per i messaggi di errore. La gestione degli errori avviene attraverso SWITCH CASE.

A video sono state mappate tutte le lettere maiuscole A-Z, i numeri da 0-9 e alcuni caratteri speciali come : “ **!** “; “ **,** “ ; etc.

## Possibili migliorie

Il sistema possiede già un interfaccia utente che comprende tutte le possibili interazioni che si possono effettuare con il modulo, tuttavia per poter navigare nel menù si deve ancora utilizzare il terminale. In una possibile seconda versione si potrebbe utilizzare un keypad per navigare nel menù e in caso, associare ad ogni impronta un pin favorendo così la ormai nota autenticazione a due fattori.



*Figura 12. Possibile keypad da abbinare*

E’ da sottolineare che con poche modifiche il sistema può essere collegato a qualunque sistema che necessita una sicurezza prima di poter essere avviato, ad esempio l’apertura di una porta in un edificio, l’accensione di un computer, etc.