#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif#include <SPI.h>

#include "FishinoSdFat.h"

SdFat SD;

#ifdef SDCS

#define SD\_CS\_PIN SDCS

#else

#define SD\_CS\_PIN SS

#endif

Sensore cassetta delle lettere

# Indice

1 Indice 2

2 Introduzione 3

2.1 Informazioni sul progetto 3

2.2 Abstract 3

2.3 Scopo 3

3 Analisi 4

3.1 Analisi del dominio 4

3.2 Analisi e specifica dei requisiti 4

3.3 Use case 7

3.4 Pianificazione 7

3.5 Analisi dei mezzi 9

3.5.1 Software 9

3.5.2 Hardware 9

4 Progettazione 9

4.1 Design dell’architettura del sistema 9

4.2 Design dei dati e database 9

4.3 Design delle interfacce 12

4.4 Design procedurale 12

5 Implementazione 13

6 Test 13

6.1 Protocollo di test 13

6.2 Risultati test 15

6.3 Mancanze/limitazioni conosciute 15

7 Consuntivo 15

8 Conclusioni 17

8.1 Sviluppi futuri 17

8.2 Considerazioni personali 17

9 Bibliografia 18

9.1 Bibliografia per articoli di riviste: 18

9.2 Bibliografia per libri 18

9.3 Sitografia 18

10 Glossario 19

11 Indice delle figure 19

12 Allegati 19

# Introduzione

## Informazioni sul progetto

* Allievi: Alessandro Perri, Alex Ierardi e Diego Muniz
* Supervisore: Geo Petrini
* Classe: 3BB SAM Trevano, sezione informatica
* Data di inizio progetto: 27.01.23
* Data di consegna: 05.05.2023

## Abstract

**Situazione iniziale**

**Approccio**

**Risultati**

## Scopo

Lo scopo del progetto (scopi didattici/scopi operativi). Dovrebbe descrivere il mandato, ma non vanno ricopiate le informazioni del quaderno dei compiti (che va invece allegato).

# Analisi

## Analisi del dominio

Per questo progetto è stato richiesto di sviluppare un sistema per il rilevamento della posta e dei pacchi in una cassetta delle lettere. Non appena il sistema rileverà una lettera, all’utente verrà inviata una notifica che indica che la posta è arrivata.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **Req-01** | |
| **Nome** | Realizzare codice per la trasmissione dei dati |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-02** | |
| **Nome** | Realizzare codice per la raccolta dei dati |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-03** | |
| **Nome** | Elaborare protocollo di trasmissione |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-03** | |
| **Nome** | Progettare Hardware per la cassetta delle lettere |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-04** | |
| **Nome** | Realizzare Hardware per la cassetta delle lettere |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-05** | |
| **Nome** | Salvataggio all’interno di un DB adeguato |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

|  |  |
| --- | --- |
| **Req-06** | |
| **Nome** | Realizzare codice per la raccolta e trasmissione dei dati |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **Req-012\_1** |  |
| **Req-012\_2** |  |
| **Req-012\_3** |  |

**Spiegazione elementi tabella dei requisiti**

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: l'importanza con la quale deve essere svolto un requisito.

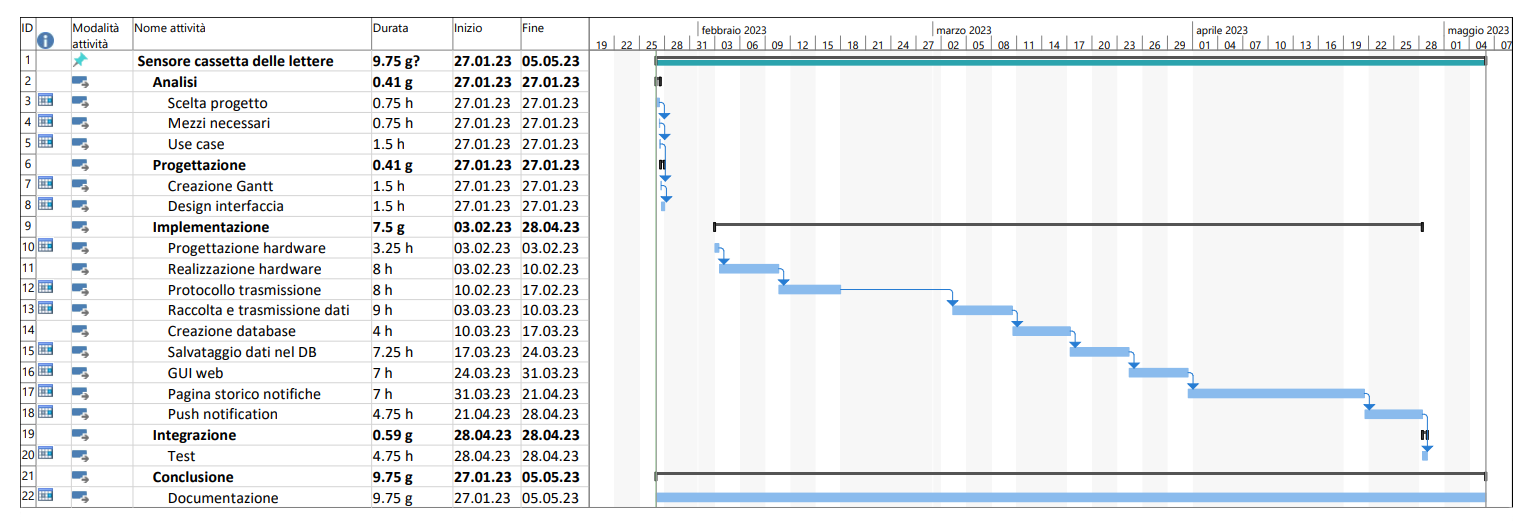
**Versione**: indica la versione del requisito.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

## Pianificazione

Di seguito c’è la pianificazione preventiva, nel Gantt preventivo è stimato il tempo necessario per completare le task del progetto.

## Analisi dei mezzi

Per realizzare il progetto è stato utilizzato

### Software

Per realizzare questo progetto sono stati utilizzati i seguenti software:

* OS Computer - Windows 10: Per la creazione di tutto il progetto.

### Hardware

* HDD e SSD per memorizzare le VM

Un computer con:

* CPU Intel Core i7-7700
* RAM 16 GB

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

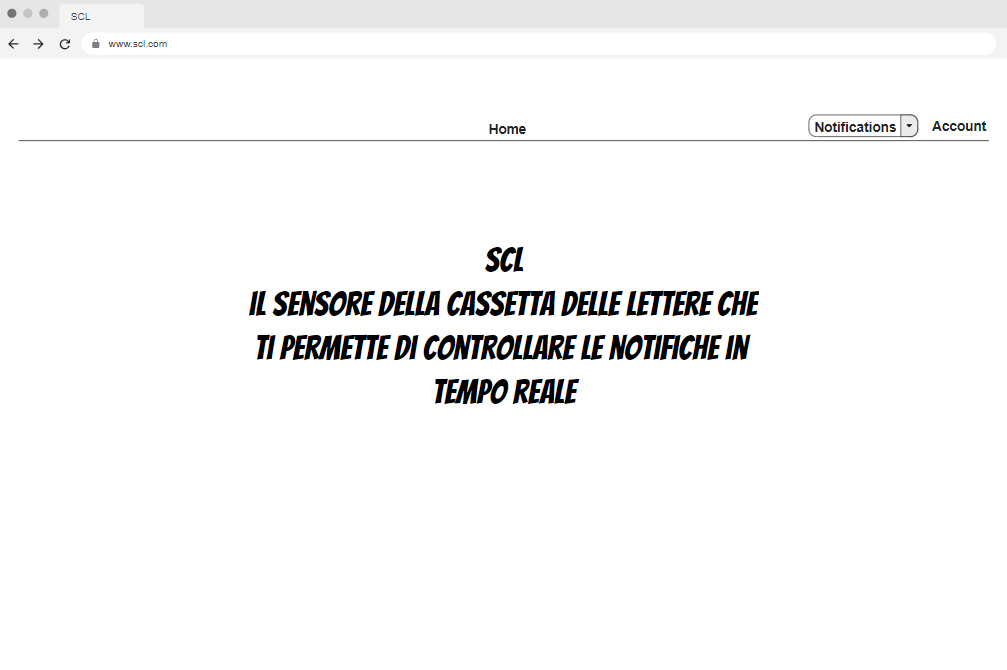
Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

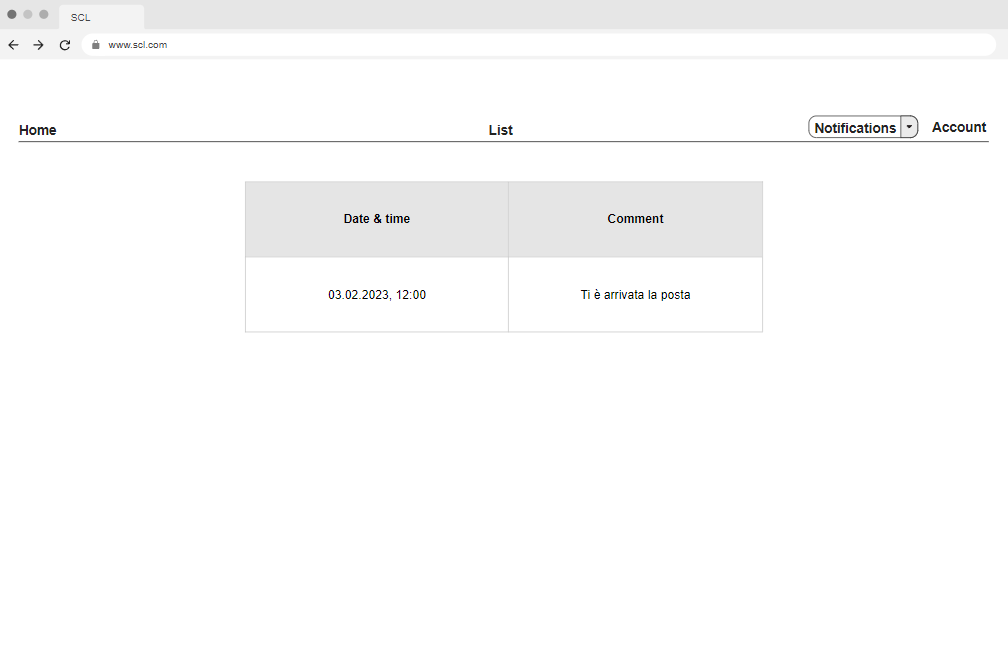
Schema E-R, schema logico e descrizione.

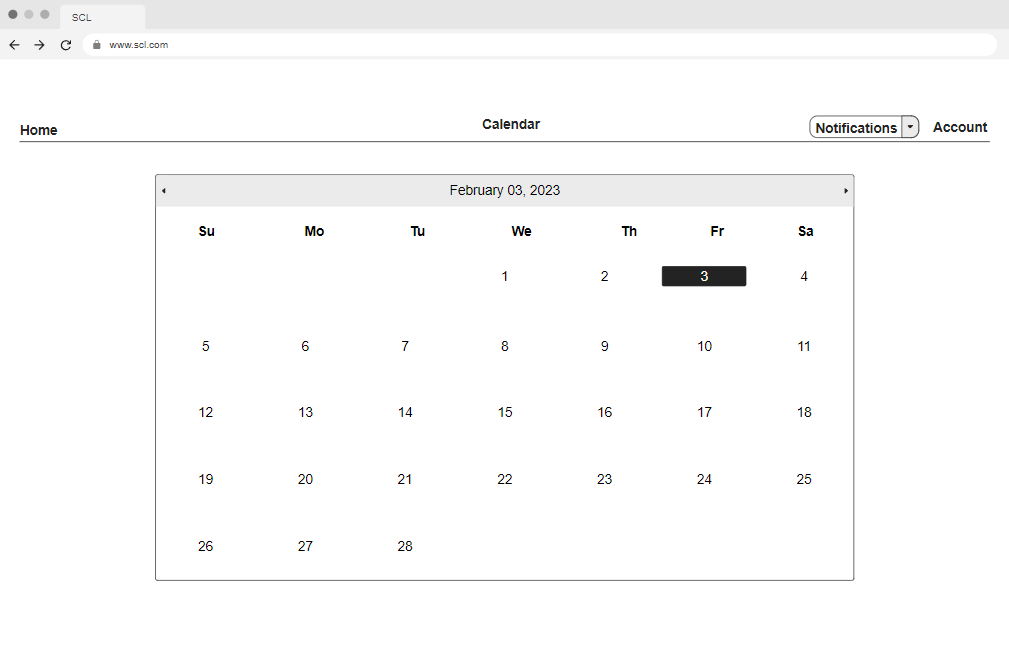
Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

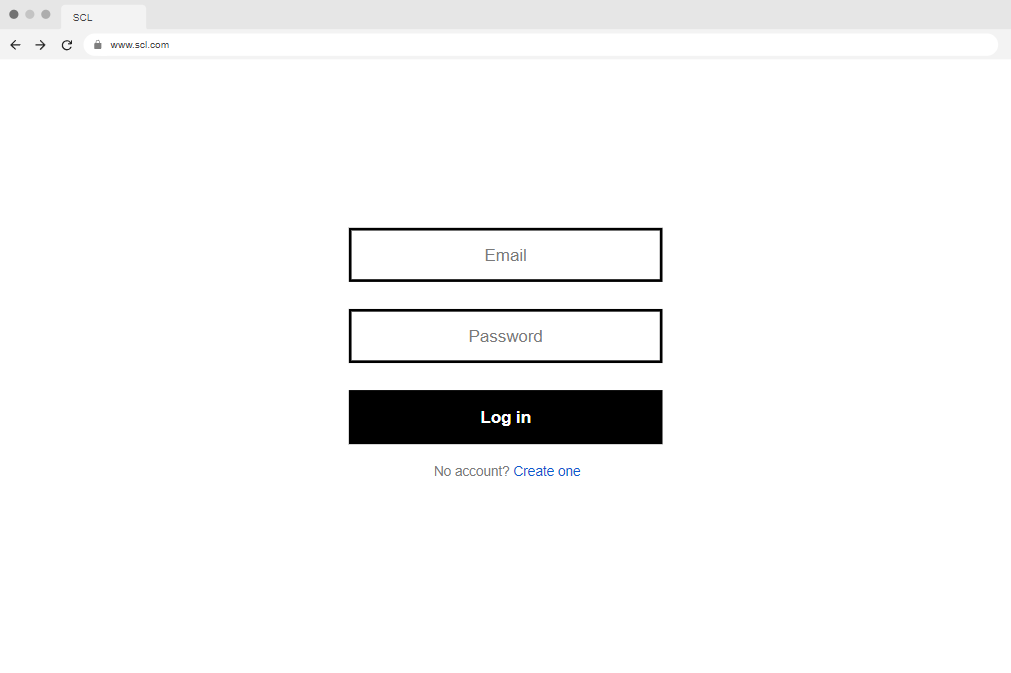
## Design delle interfacce

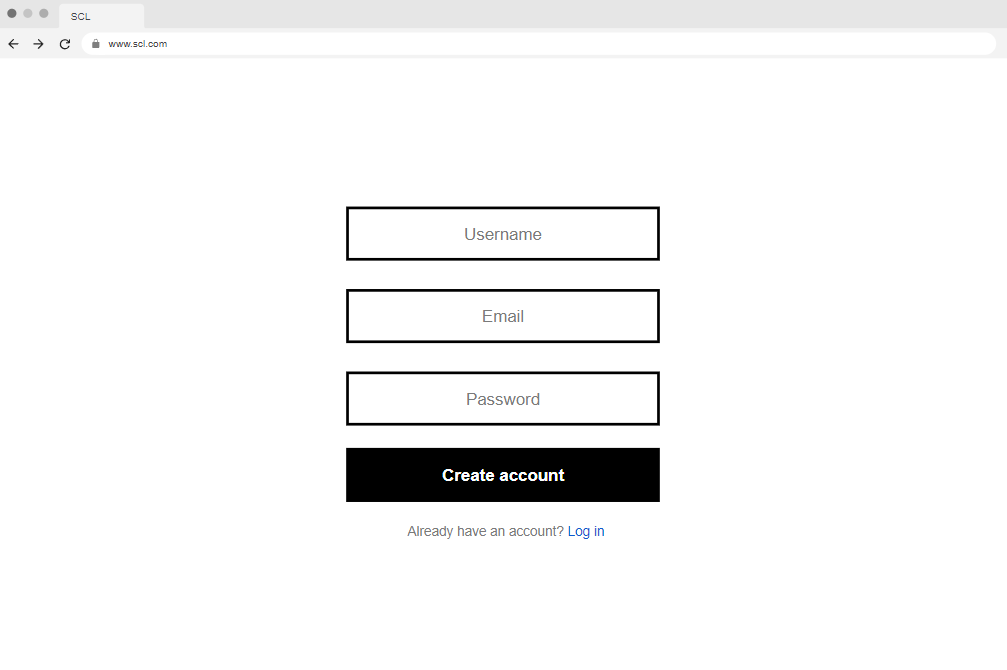
Il design del sito è strutturato nel seguente modo:

**Interfaccia home page**

**Interfaccia notifiche in lista**

**Interfaccia notifiche nel calendario**

**Interfaccia di login**

**Interfaccia creazione account**

# Implementazione

## Scrittura all’interno del SD

### Import e direttive

Il codice inizia importando le librerie "SPI.h" e "FishinoSdFat.h". Viene poi istanziata una classe "SdFat" chiamata SD. La riga successiva contiene una direttiva di precompilazione che definisce la costante "SD\_CS\_PIN" o a SDCS (Se presente) o a SS. La libreria "FishinoSdFat.h" è una versione personalizzata della libreria "SdFat.h" per poter funzionare con la scheda Fishino. Il codice in questo modo è in grado di gestire la lettura e la scrittura di dati sulla scheda SD montata sul Fishino.

### Scrittura del file

Il codice inizia con la dichiarazione di una variabile di tipo File chiamata myFile. Successivamente, nel setup, viene inizializzata la comunicazione seriale e viene eseguita la procedura di inizializzazione della scheda SD. Se la procedura non ha successo, viene stampato un messaggio di errore. Viene quindi creato un nuovo file di testo chiamato "config.txt" tramite la funzione SD.open(). Successivamente, il file viene rimosso (se esiste già) e quindi aperto nuovamente. Se l'apertura del file ha successo, viene costruita una stringa contenente una serie di valori separati da virgola. Questi dati di configurazione comprendono un nome di una rete Wi-Fi (MOD340), una password (passM340), un identificatore univoco (bad6f4d7-3c35-4d34-92c0-f18839a5372a), un URL per una API (Application Programming Interface) di registrazione dati, un indirizzo IP per un server remoto (164.128.168.41). Infine, il file viene chiuso e viene stampato un messaggio di successo o di errore a seconda del risultato dell'operazione.

### Stampa del contenuto nel file

Questo codice apre il file “config.txt” presente nel SD e se l’apertura è riuscita legge il contenuto del file e lo stampa sul terminale

## Connessione al Wi-Fi e invio dati

### Import e variabili globali

Il codice configura prima alcune variabili e librerie necessarie per l'utilizzo della scheda microSD e della connessione Wi-Fi tramite il modulo Fishino. In particolare, si importano le librerie "Fishino.h", "FishinoSdFat.h" e "SPI.h". Si definisce poi l'oggetto "client", di tipo "FishinoClient", e si usa successivamente per l'interazione con il server web. Vengono dichiarate tutte le variabili globali che serviranno all’interno del codice.

### Stampa informazioni file nel SD

All’interno di questa funzione viene effettuata l’inizializzazione della scheda SD e viene controllato che l’operazione sia stata eseguita con successo, altrimenti viene richiamata la funzione “SD.initErrorHalt()” che blocca l’esecuzione del programma. Per ogni file aperto viene salvato l'indice della directory corrente in cui si trova, viene poi stampato il nome del file sulla console seriale e viene incrementata la variabile "n". Infine, il file viene chiuso.

### Lettura del file nel SDL

Questa funzione legge il contenuto del file chiamato “config.txt” dalla scheda SD fina ad un massimo di 500 caratteri e restituisce una stringa con il contenuto del file.

### divisione dei contenuti prese

Questa funzione prende la stringa di ritorno del metodo “readSDCardFile” e divide i valori separati da virgola in elementi singoli e li inserisce dentro l’array di stringhe globale strs.

### Connessione al Wi-Fi

Questa funzione si occupa di connettere la scheda Fishino ad una rete Wi-Fi utilizzando lo ssid e la password forniti come argomenti. Nella prima riga della funzione viene invocato il metodo "begin" della libreria Fishino, passando lo ssid e la password come argomenti, e viene utilizzato un loop while per attendere il completamento della connessione WiFi. Successivamente, viene chiamato il metodo "staStartDHCP" per inizializzare il client DHCP, il quale permette di ottenere un indirizzo IP in modo dinamico dalla rete WiFi. Infine, viene eseguita una serie di print sulla console seriale, mostrando l'avanzamento della connessione WiFi, finché lo status della connessione non diventa "STATION\_GOT\_IP" (ovvero quando l'IP viene assegnato alla scheda Fishino). Una volta completata la connessione, viene stampato un messaggio di "OK" sulla console seriale.

## Invio dei dati

### Creazione json da inviare

Questo codice invia dati ad un server tramite una connessione client. Il formato dei dati inviati è un oggetto JSON che contiene un valore “api\_key” che deve essere qullo generato all’interno del sito. All’interno del ciclo quando il carattere trovato è ”x” lo rimpiazza con un carattere della chiave, abbiamo utilizzato questo metodo perché il concat delle stringhe non funzionava come doveva.

### Inoltre dei dati

Questo codice è una funzione che esegue una richiesta POST ad un server HTTP. Utilizza la libreria Ethernet per stabilire una connessione al server e inviare un messaggio JSON tramite il protocollo HTTP utilizzando la porta 80. All'inizio, il codice controlla la connessione attraverso il metodo connect() della classe client. Se la connessione ha successo, il client stabilisce una connessione e procede con la richiesta. Viene quindi inviata la richiesta POST con POST serverPath HTTP/1.1, dove serverPath è l'URL del server al quale si vuole inviare la richiesta. Successivamente, vengono inviati alcune intestazioni tra cui Host, User-Agent e Content-Type che specificano il tipo di contenuto. Viene poi inviato il body della richiesta sotto forma di messaggio JSON. Infine, viene chiusa la connessione con il server, tramite il metodo stop().

## Metodo setup()

Si inizia impostando la velocità della comunicazione seriale attraverso Serial.begin(115200). Viene poi dichiarato il pin 2 come input e il pin 4 come output attraverso pinMode(2, INPUT) e pinMode(4, OUTPUT). Viene poi impostata la modalità di connessione alla rete attraverso Fishino.setMode(STATION\_MODE). Dopodiché vengono richiamate le unzioni “printSDCardFiles()” e “parseFromSDCardFile()”. Vengono salvate le informazioni di strs all’interno di singole variabili globali, converto attraverso un for la stringa per il percorso del server da stringa ad array di caratteri, le stampo e infine richiamo il metodo per la connessione al Wi-Fi

## Metodo loop

Il codice inizia dichiarando una variabile "oldSensorVal" con valore iniziale di 1. Nella funzione "loop" viene poi letto il valore del sensore collegato al pin 2 e salvato nella variabile "currentSensorVal". Se il sensore passa da uno stato di alta a bassa attivazione (da HIGH a LOW), viene richiamata la funzione "dataSender". La variabile "oldSensorVal" viene poi aggiornata al nuovo stato del sensore (0). In alternativa, se il sensore passa da bassa a alta attivazione (da LOW a HIGH), l' "oldSensorVal" viene semplicemente aggiornato al nuovo stato (1).

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-012 | **Nome** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | Testo o immagine a conferma del test | Passato |
| TC-002 | Testo o immagine a conferma del test | Fallito |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo

# Conclusioni

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

**Alex Ierardi**

**Diego Muniz**

**Alessandro Perri**

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| ASCII | American Standard Code for Information Interchange, codice per la codifica di caratteri. |
|  |  |
|  |  |
|  |  |
|  |  |
| ZFS | Zettabyte File System, filesystem opernsource a 128 bit con funzionalità avanzate. |

# Indice delle figure

[Figura 1 Esempio di diagramma di Gantt 7](#_Toc124775984)

[Figura 2 Esempio di diagramma di Gantt 8](#_Toc124775985)

[Figura 3 Diagramma ER Chen 10](#_Toc124775986)

[Figura 4 Diagramma ER Barker 11](#_Toc124775987)

[Figura 5 Esempio di diagramma di Gantt consuntivo 16](#_Toc124775988)

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …