Progetto 2

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 6

1.7 Pianificazione 6

1.8 Analisi dei mezzi 6

1.8.1 Software 6

1.8.2 Hardware 6

2 Progettazione 7

2.1 Design dell’architettura del sistema 7

2.2 Design dei dati e database 7

2.3 Design delle interfacce 7

2.4 Design procedurale 7

3 Implementazione 8

4 Test 8

4.1 Protocollo di test 8

4.2 Risultati test 9

4.3 Mancanze/limitazioni conosciute 9

5 Consuntivo 9

6 Conclusioni 9

6.1 Sviluppi futuri 9

6.2 Considerazioni personali 9

7 Bibliografia 9

7.1 Bibliografia per articoli di riviste: 9

7.2 Bibliografia per libri 9

7.3 Sitografia 9

8 Allegati 10

# Introduzione

## Informazioni sul progetto

Autore: Matan Davidi e Filippo Finke

Scuola: Arti e Mestieri Trevano

Classe: I3AA

Anno scolastico: 2018/19

Sezione: Informatica

Materia: Modulo 306

Docenti responsabili: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Elisa Nannini

Data di inizio: 14.11.2018

Data di consegna: 30.01.2018

## Abstract

How many times has a programmer decided to learn a new language only to be discouraged for whatever reason at the beginning of the road? How many people may actually be interested in learning to program but are afraid because it looks too complicated?

This document contains the technical documentation of a user-friendly library for new programmers to help them get comfortable with programming in Arduino, a programming language based on C++, at their own pace.

Please note that this document contains the initial analysis, the design of the library and associated circuits, an explanation of the code contained in the library and of the realization of the associated electronic circuits.

## Scopo

Creare una libreria di codice utilizzabile tramite un modello base di [Digispark USB Development Board](http://digistump.com/products/1) per avvicinare nuovi utenti, per esempio studenti di terza media, al mondo dell’informatica e dell’elettronica.

## Analisi

## Analisi del dominio

Adesso come adesso, prima della realizzazione del nostro progetto, la programmazione in Arduino implica come prerequisiti delle conoscenze base di programmazione in linguaggi C-like e di montaggio di circuiti elettronici. Questo rischia di allontanare i nuovi utenti a questo mondo che combina programmazione con utilità pratica. La nostra libreria è progettata per aiutare le persone che si interfacciano agli Arduino oppure all’informatica per la prima volta senza dover scrivere troppo codice, in modo da potersi concentrare sulla comprensione di quello che si scrive.

Idealmente, questa libreria è stata pensata per essere utilizzata da studenti delle scuole medie, che non posseggono alcuna conoscenza né di programmazione né di elettronica, che vengono a fare una giornata informativa alla Scuola Arti e Mestieri di Trevano in modo che possano portare a casa un lavoro fatto completamente da loro.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| Nome | Libreria Arduino |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | Bisogna realizzare una libreria compatibile con il linguaggio Arduino |
| 002 | La libreria deve essere realizzata in linguaggio C++ |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| Nome | LEDLib |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | È necessario implementare una libreria per controllare lo stato di un LED |
| 002 | La libreria deve implementare un metodo che ottiene lo stato del LED associato |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| Nome | ButtonLib |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | È necessario implementare una libreria per controllare lo stato di un bottone |
| 002 | La libreria deve implementare un metodo che ottiene lo stato del bottone associato |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| Nome | PhotocellLib |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | È necessario implementare una libreria per controllare lo stato di una fotocellula |
| 002 | La libreria deve implementare un metodo che ottiene lo stato del bottone associato |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| Nome | LED – Bottone |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | È necessario realizzare tre circuiti di esempio che contengano una combinazione del LED e del bottone |
| 002 | Uno dei circuiti da realizzare deve accendere il LED quando viene premuto il bottone |
| 003 | Uno dei circuiti da realizzare deve invertire lo stato del LED quando viene premuto il bottone |
| 004 | Uno dei circuiti da realizzare deve invertire lo stato del LED velocemente da quando viene premuto il bottone fino a quando non viene rilasciato |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| Nome | LED – Fotocellula |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | È necessario realizzare tre circuiti di esempio che contengano una combinazione della fotocellula e del LED |
| 002 | Uno dei circuiti da realizzare deve accendere o spegnere il LED in base al valore rilevato dalla fotocellula: se il valore è al di sopra di una soglia il LED viene acceso; se il valore è al di sotto di una soglia il LED viene spento |
| 003 | Uno dei circuiti da realizzare deve regolare l’intensità del LED in base al valore rilevato dalla fotocellula: più è alta la luminosità rilevata, maggiore è l’intensità del LED. |
| 004 | Uno dei circuiti da realizzare deve regolare la frequenza di lampeggiamento del LED in modo inversamente proporzionale al valore rilevato dalla fotocellula: più è alta la luminosità rilevata, minore è la frequenza di lampeggiamento del LED. |

Il progettista, dopo aver ricevuto il mandato, in collaborazione con il committente redige una lista di requisiti. Durante questi incontri, tramite interviste (da inserire nei diari), il progettista deve cercare di rispondere alle seguenti domande:

* Quali sono i bisogni del committente?
* Quali funzioni deve svolgere il prodotto?
* Come devono essere implementate?
* L’utente, come vorrebbe/dovrebbe interagire con il prodotto?
* Come verrà utilizzato il prodotto?
* Che tipo di interfaccia si immagina?
* Che prestazioni minime deve fornire il prodotto?
* Che grado di sicurezza deve avere il prodotto?
* …

In base alla lista dei requisiti e all’analisi degli stessi, il progettista redige una *specifica dei requisiti* in cui elenca e descrive in modo dettagliato quali sono le funzionalità che il prodotto fornirà. La specifica dovrebbe essere abbastanza dettagliata da poter essere utilizzata come base per lo sviluppo, ma non troppo; ad esempio non dovrebbe contenere dettagli di implementazione, o definizioni dettagliate dell’interfaccia grafica a meno che questi non siano considerati cruciali. Non si deve scordare che i requisiti non rappresentano delle attività bensì delle caratteristiche che il prodotto dovrà possedere.

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Creazione interfaccia banca dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano i permessi di root / Dipende dal requisito REQ-001 (Creazione DB) |
| **Sotto requisiti** | |
| **001** | Si necessita una maschera di login |
| **002** | Si dovranno poter immettere nuovi allievi |
| **003** | Dovrà essere possibile la ricerca di allievi |

**Spiegazione elementi tabella dei requisiti:**

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

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

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

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Figura 1: Esempio di diagramma di Gantt. |

Se si usano altri metodi di pianificazione (es scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

Il progetto è stato sviluppato su un sistema operativo Windows 10 Home a 64 bit versione 10.0.17134 build 17134 e macOS Mojave 10.14.1 utilizzando il seguente software:

* Arduino 1.8.7
* Atom 1.32.2
* Fritzing 0.9.3
* GanttProject 2.8.9
* GitHub Desktop 1.5.0
* Google Chrome 70.0.3538.110
* Microsoft Visio 2010 14.0.4756.1000
* Microsoft Visual Studio Code 1.29.1
* Microsoft Word 16.0.10730.20102
* Mozilla Firefox 63.0.3
* SourceTree 3.0.8

Le librerie utilizzate comprendono:

* Arduino (Arduino.h) per il linguaggio C++, che permette di scrivere codice in C++ utilizzando i metodi e le funzioni di Arduino

### Hardware

* HP Pavilion - 15-au147nz:
  + Monitor da 15.6 pollici 1920x1080
  + Intel® Core™ i7-7500U 2.7 GHz Dual-core
  + 16 GB RAM
  + NVIDIA® GeForce® 940MX
  + Intel® HD Graphics 620
* MacBook Pro 2018 13”:
  + Monitor da 13 pollici 2560×1600
  + Intel® Core™ i5-8259U 2.3 Ghz Quad-core
  + 8 GB RAM 2133 MHz LPDDR3
  + Intel Iris Plus Graphics 655 1536 MB
* Digispark USB Development Board ([specifiche](http://digistump.com/products/1" \l "product_description))
  + Alimentazione tramite USB o fonte esterna - 5v o 7-35v (12v o meno consigliato, selezione automatica)
  + Regolatore da 500mA e 5V incorporato
  + USB incorporato
  + 6 Pin I/O (2 vengono utilizzati per USB solo se il programma comunica attivamente tramite USB, altrimenti è possibile utilizzare tutti e 6 anche se si sta programmando via USB)
  + Memoria Flash da 8k (circa 6k dopo il bootloader)
  + Pin di I2C e SPI
  + PWM su 3 pin (altri possibili con il software PWM)
  + ADC su 4 pin
  + LED di alimentazione e LED di stato/test

# 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

|  |  |
| --- | --- |
| Figura 2: Bottone - LED, schema di circuito | Figura 3: Fotocellula - LED, schema di circuito |
| Figura 4: Bottone - LED, schema elettrico | Figura 5: Fotocellula - LED, schema elettrico |

## Design procedurale

Classe Button (ButtonLib):

* button.cpp
* button.h
* keywords.txt

Metodi:

* + Button(int pin), che istanzia nuovi oggetti di tipo Button, accettando come parametro il numero del pin da cui leggere lo stato del bottone
  + bool getState(), che ottiene lo stato del bottone: 1 se premuto, 0 se non premuto.

Classe Led (LedLib):

* led.cpp
* led.h
* keywords.txt

Metodi:

* Led(int pin), che istanzia nuovi oggetti di tipo Led, accettando come parametro il numero del pin al quale è collegato il LED
* void on(), che imposta lo stato del LED a 1: acceso
* void off(), che imposta lo stato del LED a 0: spento
* void toggle(), che inverte lo stato del LED: da acceso a spento e da spento ad acceso
* void setState(bool state), che imposta lo stato del LED al valore passato come parametro, ‘true’ lo accende e ‘false’ lo spegne
* void setAnalogState(int value), che imposta lo stato del LED con un valore analogico, da 0 a 1023
* bool getState(), che restituisce lo stato del LED, ‘true’ è acceso e ‘false’ è spento

Classe Photocell (PhotocellLib):

* photocell.cpp
* photocell.h
* keywords.txt

Metodi:

* Photocell(int pin), che istanzia nuovi oggetti di tipo Photocell, accettando come parametro il numero del pin da cui leggere il valore restituito dalla fotocellula
* int getLux(), che restituisce il valore della luminosità rilevato dalla fotocellula, da 0 a 1023

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

Tutte le librerie descritte in seguito importano la libreria di Arduino, i cui metodi sono dichiarati nel file header ‘Arduino.h’.

## ButtonLib

Questa libreria viene usata per controllare lo stato di un bottone attraverso i metodi presenti nella classe Button, quindi ogni bottone presente all’interno di un circuito dovrà corrispondere ad un’istanza di questa classe. Questa classe è composta dal file ‘button.cpp’, ‘button.h’ e ‘keywords.txt’. La libreria è composta dai seguenti metodi:

### Costruttore

Button::Button(int pin)

Il metodo costruttore Button istanzia un nuovo oggetto di tipo Button, permettendo di specificare il pin al quale è attaccato il filo che permette di leggere lo stato del bottone.

### getState

bool Button::getState()

Il metodo getState permette di ottenere lo stato del bottone. Esso ritorna, infatti, un valore booleano che può assumere il valore true quando il bottone è premuto, oppure false quando il bottone non è premuto.

## LedLib

Questa libreria viene usata per controllare lo stato di un LED attraverso i metodi presenti nella classe Led, quindi ogni LED presente all’interno di un circuito dovrà corrispondere ad un’istanza di questa classe. Questa classe è composta dal file ‘led.cpp’, ‘led.h’ e ‘keywords.txt’. La libreria è composta dai seguenti metodi:

### Costruttore

Led::Led(int pin)

Il metodo costruttore Led istanzia un nuovo oggetto di tipo Led, permettendo di specificare il pin al quale è attaccato il filo che permette di leggere e/o scrivere lo stato del LED.

### on

void Led::on()

Il metodo on permette di impostare lo stato del LED a 1, che significa accendere il LED fino al momento in cui non viene spento.

### off

void Led::off()

Il metodo off permette di impostare lo stato del LED a 0, che significa spegnere il LED fino al momento in cui non viene acceso.

### toggle

void Led::toggle()

Il metodo toggle permette di impostare lo stato del LED al valore inverso rispetto a quello corrente, che significa accendere il LED se è correntemente spento, oppure spegnerlo se dovesse essere acceso. Questo stato viene mantenuto fino al prossimo cambiamento di stato provocato da una chiamata a uno dei metodi on, off o toggle.

### setState

void Led::setState(bool state)

Il metodo setState permette di impostare lo stato del LED al valore booleano passato come parametro state. I valori accettabili per il parametro state di questo metodo sono true per accendere il LED e false per spegnerlo.

### setAnalogState

void Led::setAnalogState(int value)

Il metodo setAnalogState permette di impostare lo stato del LED al valore del numero intero passato come parametro value. Questo permette di regolare l’intensità della luce emanata dal LED con un valore da 0 a 255, dove 0 significa completamente spento e 255 significa acceso alla massima luminosità.

### getState

bool Led::getState()

Il metodo getState permette di ottenere un valore che rappresenta lo stato del LED. Esso ritorna, infatti, un valore booleano che simboleggia lo stato del LED: se il valore ritornato è true il LED è acceso; se il valore ritornato è false il LED è spento.

### getAnalogState

int Led::getAnalogState()

Il metodo getAnalogState permette di ottenere un valore che rappresenta lo stato analogico del LED, che significa l’intensità della luce che emana. Esso ritorna, infatti, un valore compreso tra 0, che significa che il LED è completamente spento, e 255, che significa che il LED è acceso alla massima intensità.

## PhotocellLib

### Costruttore

Photocell::Photocell(int pin)

Il metodo costruttore Photocell istanzia un nuovo oggetto di tipo Photocell, permettendo di specificare il pin al quale è attaccato il filo che permette di leggere lo stato della fotoresistenza.

### getLux

int Photocell::getLux()

Il metodo getLux restituisce il valore misurato dalla fotoresistenza. Infatti esso ritorna un valore intero tra 0 e 1023, dove 0 significa che non è stata rilevata alcuna luce e 1023 significa che è stata rilevata una luminosità maggiore o uguale al valore massimo rilevabile dalla fotoresistenza.

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente/print screen di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# 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.

## 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 Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# 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

* <https://digistump.com/wiki/digispark/tutorials/connecting>  
  *digispark:tutorials:connecting [Digistump Wiki]*  
  05 Dicembre 2018
* <https://learn.adafruit.com/photocells/arduino-code>  
  Arduino Code | Photocells | Adafruit Learning System

05 Dicembre 2018

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.

# 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
* …