Progetto 2

1 Introduzione

1.1 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, Massimo Sartori
Data di inizio: 14.11.2018
Data di consegna: 08.02.2018

1.2 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 of the project and of the current situation, the hardware and software that were used to implement it, 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 and the test cases that were run to ensure the correct functioning of the library.

1.3 Scopo

Creare una libreria di codice utilizzabile tramite un modello base di Digispark Development Board (vedi sitografia) per avvicinare nuovi utenti, per esempio studenti delle scuole medie senza previe conoscenze di programmazione, al mondo dell'informatica e dell'elettronica.

1.4 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 alla programmazione 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 in cui sia il montaggio del circuito elettronico sia la programmazione della logica di funzionamento siano state fatte da loro.

1.5 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
003	La libreria deve implementare un metodo che ottiene lo stato analogico del LED associato
004	La libreria deve implementare un metodo che imposta lo stato del LED associato a un valore definito
005	La libreria deve implementare un metodo che imposta lo stato del LED associato a ALTO
006	La libreria deve implementare un metodo che imposta lo stato del LED associato a BASSO
007	La libreria deve implementare un metodo che inverte lo stato del LED associato
008	La libreria deve implementare un metodo che imposta lo stato del LED associato a un valore analogico definito

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 della fotocellula associata

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 continuamente 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 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 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 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.

ID: REQ-007	
Nome	LCD – Display a segmenti liquidi
Priorità	2
Versione	1.0
Note	
Sotto-requisiti	
001	È necessario realizzare tre circuiti di esempio che contengano il display LCD
002	Uno dei circuiti deve stampare sullo schermo LCD la scritta “Hello World”.
003	Uno dei circuiti deve stampare del testo all’interno del display LCD e mostrare il lampeggiamento del cursore, attivandolo e disattivandolo ad intervalli regolari.
004	Uno dei circuiti da realizzare deve stampare del testo sullo schermo LCD e dargli un effetto di scorrimento attraverso il display a segmenti liquidi.

1.6 Pianificazione

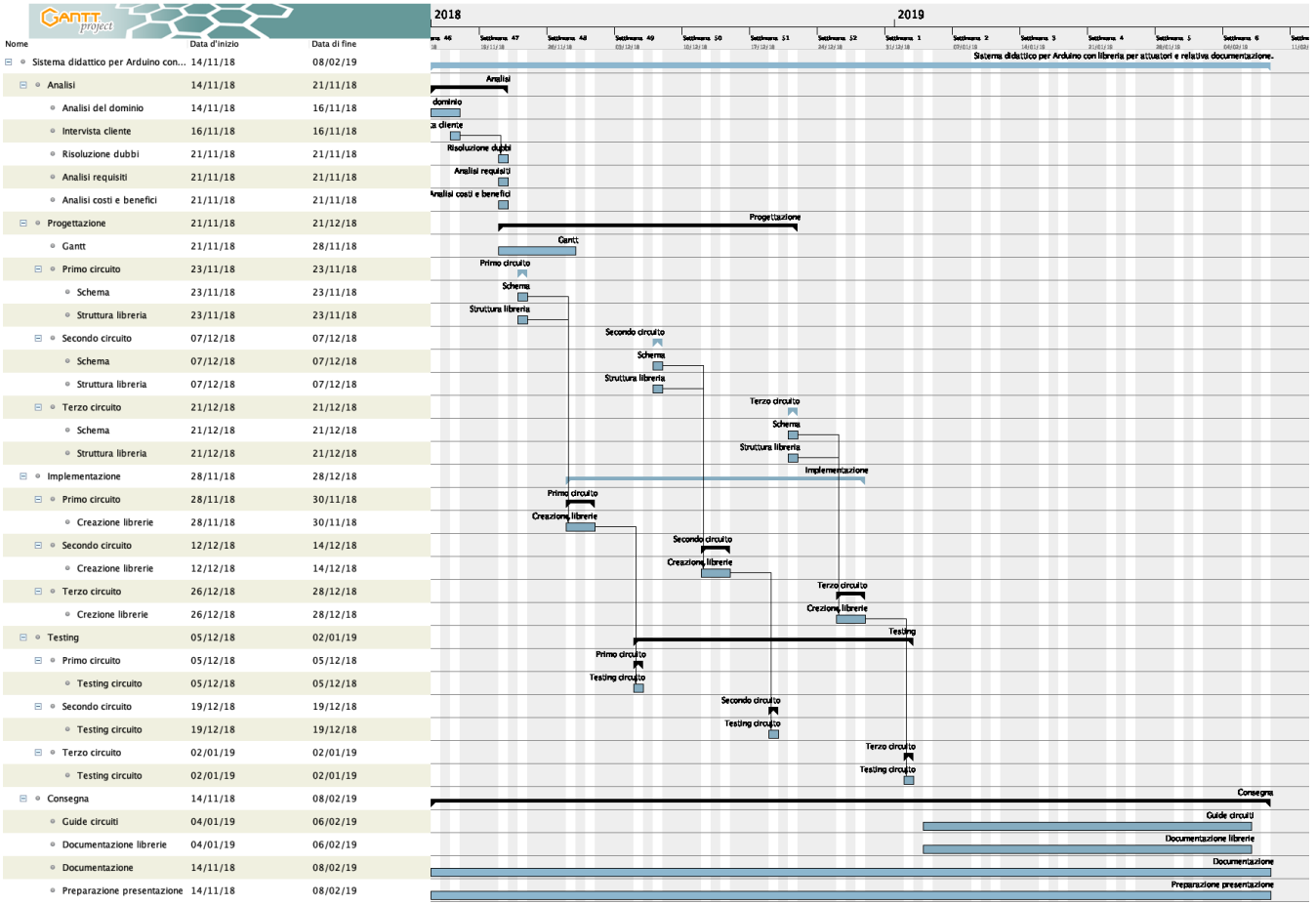


Figura 1: Diagramma di Gantt utilizzato per la pianificazione.

La pianificazione si divide in 5 fasi distinte: Analisi, Progettazione, Implementazione, Testing e Consegna; ognuna delle quali si suddivide nuovamente in attività.

1.6.1 Analisi

Analisi	14/11/18	21/11/18
Analisi del dominio	14/11/18	16/11/18
Intervista cliente	16/11/18	16/11/18
Risoluzione dubbi	21/11/18	21/11/18
Analisi requisiti	21/11/18	21/11/18
Analisi costi e benefici	21/11/18	21/11/18

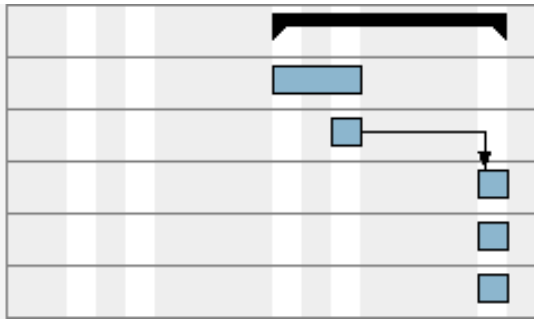


Figura 2: Ingrandimento della pianificazione della fase di analisi

In questa fase ricadono tutte le attività preliminari che servono per capire la situazione attuale e i requisiti del cliente sotto ogni aspetto. Dopodiché vengono stilati i requisiti e viene effettuata l'analisi di costi e benefici per definire se vale la pena lavorare al progetto. In questo progetto sono state svolte 5 attività:

- L'analisi del dominio, dove è stata analizzata la situazione corrente prima della realizzazione del progetto
- L'intervista con il cliente, grazie alla quale è stato possibile capire i requisiti da seguire meticolosamente durante la realizzazione del progetto
- La risoluzione dei dubbi, in cui abbiamo potuto smussare tutti gli angoli dei requisiti grazie alle domande poste al cliente in modo da implementare il progetto esattamente come vuole
- L'analisi dei requisiti, che porta a stilare i requisiti che è possibile vedere nel capitolo Analisi e specifica dei requisiti
- L'analisi dei costi e dei benefici, che permette di valutare qualora valga la pena o meno svolgere il progetto

1.6.2 Progettazione

Progettazione	21/11/18	21/12/18
Gantt	21/11/18	28/11/18
Primo circuito	23/11/18	23/11/18
Schema	23/11/18	23/11/18
Struttura libreria	23/11/18	23/11/18
Secondo circuito	07/12/18	07/12/18
Schema	07/12/18	07/12/18
Struttura libreria	07/12/18	07/12/18
Terzo circuito	21/12/18	21/12/18
Schema	21/12/18	21/12/18
Struttura libreria	21/12/18	21/12/18

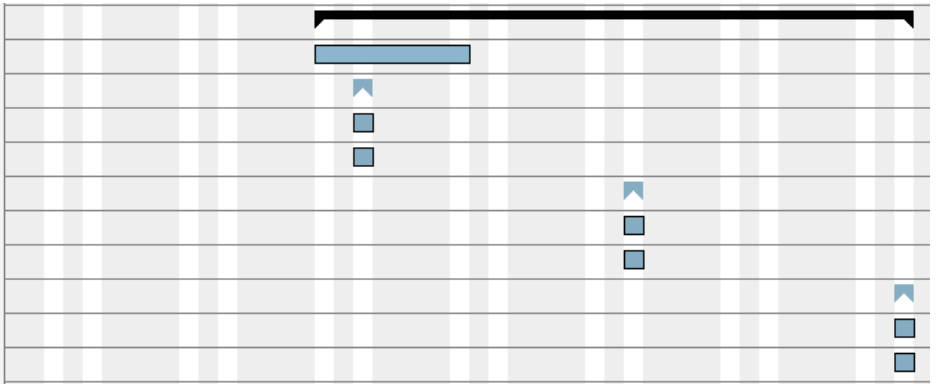


Figura 3: Ingrandimento della pianificazione della fase di progettazione

All'interno di questa fase si trovano le attività che definiscono ogni aspetto del progetto prima dell'implementazione:

- La realizzazione del diagramma di Gantt, che permette una suddivisione visiva chiara del tempo e dei costi per ogni aspetto del progetto

- La progettazione delle tre librerie per gli attuatori definiti nei requisiti: ButtonLib, LedLib e PhotocellLib, con funzionalità e metodi specifici, e dei tre circuiti di esempio per ognuna e realizzazione dei diagrammi UML per le classi da implementare. Non viene contata la libreria per il display LCD perché una libreria uguale esiste già.

1.6.3 Implementazione

Implementazione	28/11/18	28/12/18
Primo circuito	28/11/18	30/11/18
Creazione librerie	28/11/18	30/11/18
Secondo circuito	12/12/18	14/12/18
Creazione librerie	12/12/18	14/12/18
Terzo circuito	26/12/18	28/12/18
Crezione librerie	26/12/18	28/12/18

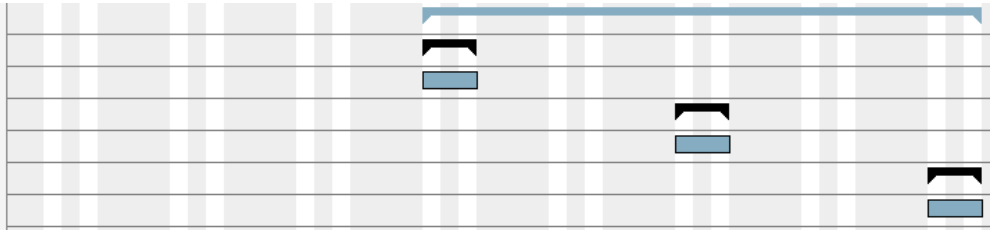


Figura 4: Ingrandimento della pianificazione della fase di implementazione

Dentro questa fase vi sono tutte le attività relative all'implementazione del progetto esattamente come descritto nella progettazione:

- La creazione delle tre librerie per gli attuatori: LED, Interruttore e Fotocellula. Non viene contata la libreria per il display LCD perché una libreria uguale esiste già.

1.6.4 Testing

Testing	05/12/18	02/01/19
Primo circuito	05/12/18	05/12/18
Testing circuito	05/12/18	05/12/18
Secondo circuito	19/12/18	19/12/18
Testing circuito	19/12/18	19/12/18
Terzo circuito	02/01/19	02/01/19
Testing circuito	02/01/19	02/01/19

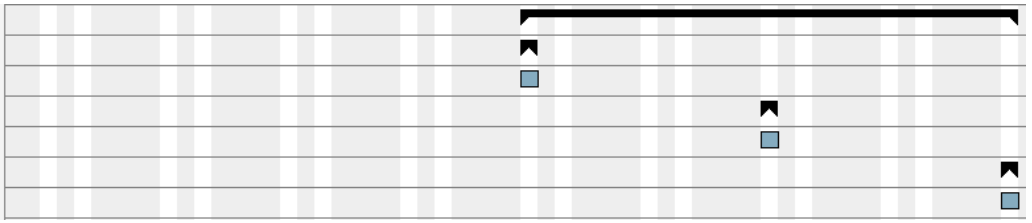


Figura 5: Ingrandimento della pianificazione della fase di testing

Nella fase di testing ricadono tutte le verifiche di funzionamento effettuate come descritto nel capitolo 4, Test:

- I test per ogni circuito realizzato: il primo per la coppia di attuatori LED – Bottone, il secondo per la coppia LED – Fotocellula e il terzo per il display a cristalli liquidi (LCD)

1.6.5 Consegna

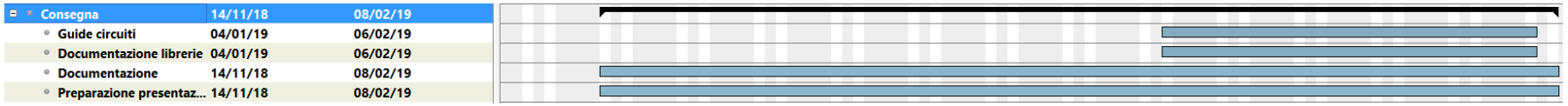


Figura 6: Ingrandimento della pianificazione della fase di consegna

All'interno dell'ultima fase, quella di consegna, si trovano le attività di preparazione per la consegna del progetto implementato al cliente:

- La realizzazione di una guida di utilizzo per ogni circuito realizzato in modo che ne sia spiegato il funzionamento
- La documentazione delle singole librerie per far capire cosa fa ogni membro di ogni classe
- La documentazione del progetto nel suo intero per sapere come è stato realizzato e cosa contiene
- La realizzazione della presentazione del progetto

1.7 Analisi dei mezzi

1.7.1 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
- draw.io

Le librerie utilizzate comprendono:

- Arduino (Arduino.h), versione incorporata nell'IDE Arduino 1.8.5, per il linguaggio C++, che permette di scrivere codice in C++ utilizzando i metodi e le funzioni di Arduino
- LiquidCrystal_I2C (LiquidCrystal_I2C.h) 1.5.8A utilizzata per la gestione del display LCD.
- TinyWireM (TinyWireM.h) 1.0.1 utilizzata per interfacciarsi con il bus I2C.

1.7.2 Hardware

- Digispark USB Development Board (specifiche)
  - 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

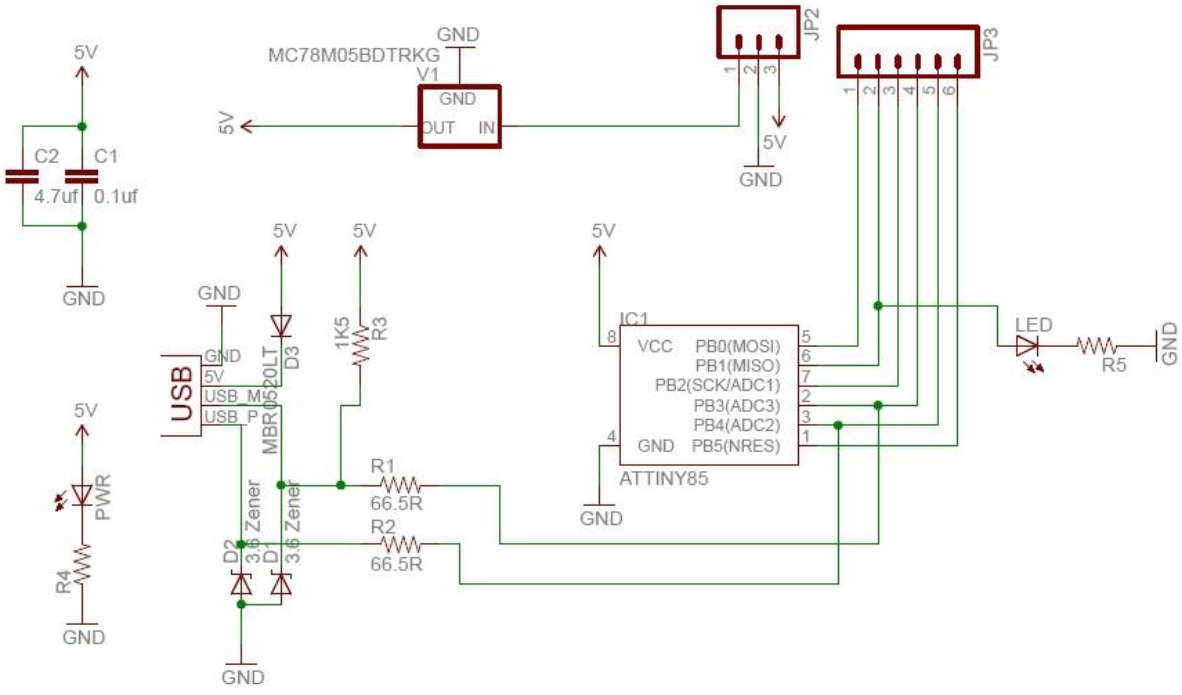


Figura 7: schema elettrico del Digispark USB Development Board

2 Progettazione

2.1 Design dell'architettura del sistema

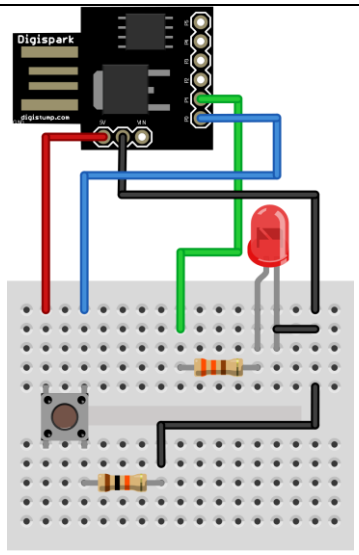


Figura 8: Bottone - LED, schema di circuito

All'interno del circuito sono presenti quattro componenti: 1 bottone, 1 LED e 2 resistenze. Il bottone è collegato in pull-down tramite una resistenza da 10kΩ, il suo stato viene letto attraverso il pin "P0" del micro controllore. Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω.

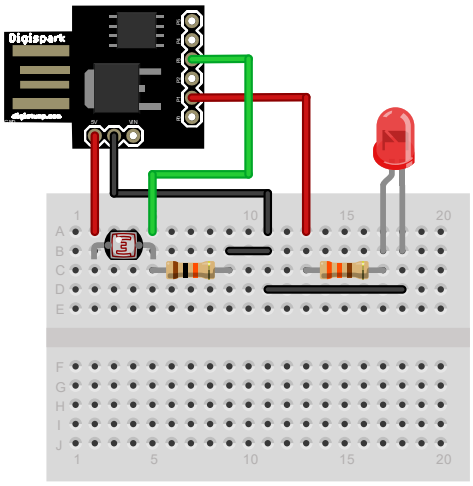


Figura 9: Fotocellula - LED, schema di circuito

All'interno del circuito sono presenti quattro componenti: 1 fotocellula, 1 LED e 2 resistenze. La fotocellula è collegata in pull-down attraverso una resistenza da 10kΩ al pin "P3". Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω.

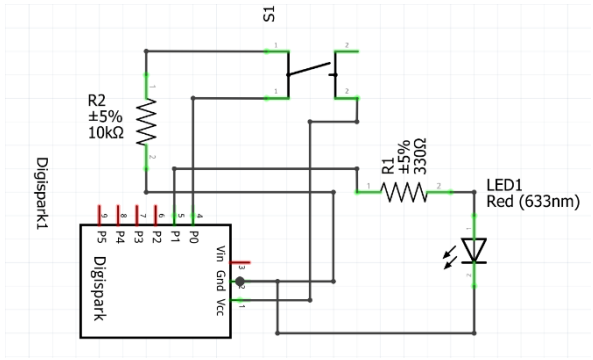


Figura 10: Bottone - LED, schema elettrico

Schema elettrico del circuito "Bottone – LED".

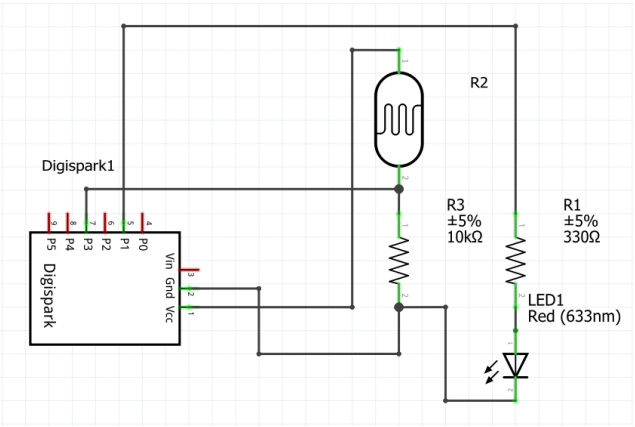


Figura 11: Fotocellula - LED, schema elettrico

Schema elettrico del circuito "Fotocellula – LED".

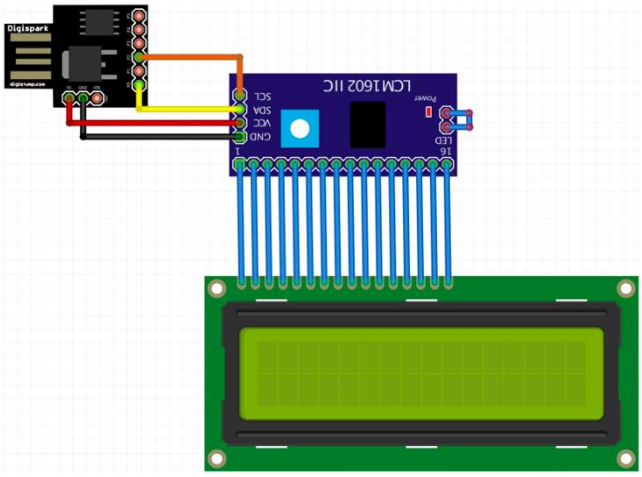


Figura 12: LCD, schema di circuito

All'interno del circuito sono presenti due componenti: 1 display a cristalli liquidi (LCD) e 1 shift register. Lo shift register viene alimentato dal digispark ed è collegato ai relativi pin SCL("P2") e SDA("P0"). Mentre il display LCD è collegato interamente allo shift register.

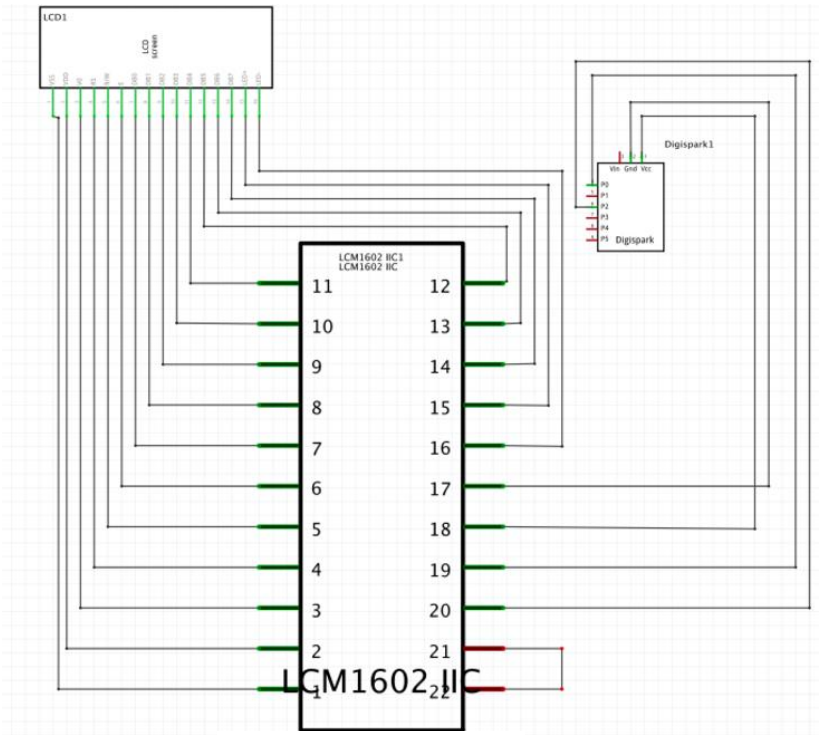


Figura 13: LCD, schema elettrico

Schema elettrico del circuito "LCD".

2.2 Design procedurale

Classe Button (ButtonLib):

- button.cpp
- button.h
- keywords.txt

Campi:

- int _pin, che contiene il numero del pin al quale è collegato il LED

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.

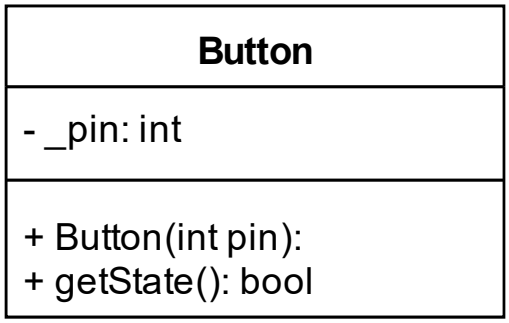


Figura 14: Schema UML della classe Button

Classe Led (LedLib):

- led.cpp
- led.h
- keywords.txt

Campi:

- int _pin, che contiene il numero del pin al quale è collegato il LED

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 255
- bool getState(), che restituisce lo stato del LED, 'true' è acceso e 'false' è spento

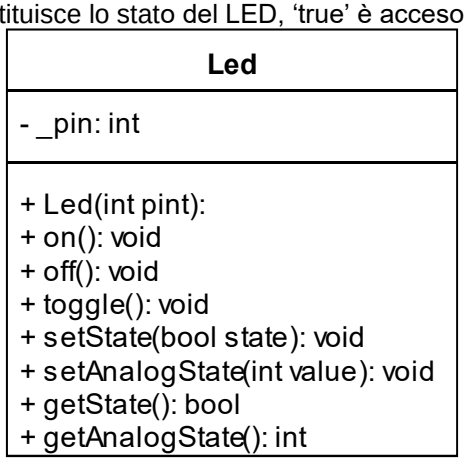


Figura 15: Schema UML della classe Led

Classe Photocell (PhotocellLib):

- photocell.cpp
- photocell.h
- keywords.txt

Campi:

- int _pin, che contiene il numero del pin al quale è collegato il LED

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 255

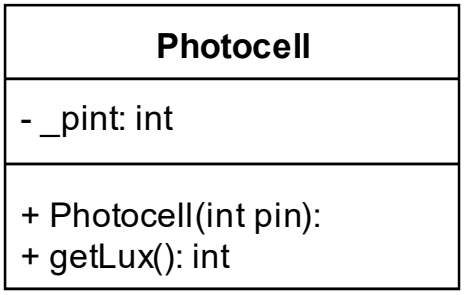


Figura 16: schema UML della classe Photocell

3 Implementazione

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

3.1 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:

3.1.1 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.

3.1.2 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.
Il valore viene ricavato attraverso un metodo della libreria Arduino utilizzando il seguente pezzo di codice:

```
return digitalRead(_pin);
```

3.2 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:

3.2.1 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.

3.2.2 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.
Questo metodo utilizza il seguente pezzo di codice per richiamare un altro metodo della classe stessa in modo da poter accendere il LED (vedi setState):

```
setState(HIGH);
```

3.2.3 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.
Questo metodo utilizza il seguente pezzo di codice per richiamare un altro metodo della classe stessa in modo da poter spegnere il LED (vedi setState):
```
setState(LOW);
```

3.2.4 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.
Questo metodo fa utilizzo di due metodi della classe stessa per ricavare il valore corrente, invertirlo ed infine impostarlo. Il tutto viene effettuato utilizzando il seguente blocco di codice (vedi setState e getState):
```
setState(!getState());
```

3.2.5 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.
Questo metodo utilizza un metodo predisposto dalla libreria di Arduino per poter impostare lo stato del LED:
```
digitalWrite(pin, state);
```

3.2.6 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.
Questo metodo utilizza un metodo predisposto dalla librearia di Arduino per poter impostare un valore analogico ad un determinato pin:
```
analogWrite(pin, value);
```

3.2.7 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.
Questo metodo utilizza un metodo predisposto dalla libreria di Arduino per poter ricavare lo stato di un determinato pin:
```
return digitalRead(pin);
```

3.2.8 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.
Questo metodo utilizza un metodo predisposto dalla libreria di Arduino per poter ricavare il valore analogico di un determinato pin:

```
return analogRead(pin);
```

3.3 PhotocellLib

Questa libreria viene usata per controllare lo stato di una resistenza fotovoltaica attraverso i metodi presenti nella classe `Photocell`, quindi ogni fotocellula presente all'interno di un circuito dovrà corrispondere ad un'istanza di questa classe. Questa classe è composta dal file 'photocell.cpp', 'photocell.h' e 'keywords.txt'. La libreria è composta dai seguenti metodi:

3.3.1 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.

3.3.2 getLux

`int Photocell::getLux()`
Il metodo `getLux` restituisce il valore misurato dalla fotoresistenza. Infatti esso ritorna un valore intero tra 0 e 255, dove 0 significa che non è stata rilevata alcuna luce e 255 significa che è stata rilevata una luminosità maggiore o uguale al valore massimo rilevabile dalla fotoresistenza.
Questo metodo utilizza un metodo predisposto dalla libreria di Arduino per poter ricavare il valore analogico di un determinato pin:

```
return analogRead(pin);
```

4 Test

4.1 Protocollo di test

Test Case: **Riferimento**:	TC-001	**Nome:**	Il Digispark funziona
Descrizione:	Il Digispark USB Development Board utilizzato deve funzionare.		
Prerequisiti:	- Un Digispark USB		
Procedura:	1. Aprire l'IDE di Arduino 2. Cliccare sull'etichetta "File" in alto a sinistra 3. Portare il mouse su "Examples" 4. Portare il puntatore su "01.Basics" 5. Selezionare la linguetta "Blink" 6. Aggiungere la seguente riga di codice al di fuori dei metodi loop e setup: `int LED_BUILTIN = 1;` 7. Avviare il programma		
Risultati attesi:	Il LED sul Digispark si accende e si spegne a intervalli regolari.		

Test Case: **Riferimento**:	TC-002	**Nome:**	Le librerie sono state importate correttamente
Descrizione:	Le librerie facenti parte di questo progetto, quindi ButtonLib, LcdLib, LedLib e PhotocellLib, sono state importate all'interno dell'ambiente di sviluppo integrato (IDE) Arduino correttamente.		
Prerequisiti:	- La libreria ButtonLib scaricata - La libreria LcdLib scaricata - La libreria LedLib scaricata - La libreria PhotocellLib scaricata		
Procedura:	1. Seguire la guida per l'utilizzo di ogni libreria contenuta all'interno della cartella della libreria, nel file README.pdf (o, eventualmente, README.md) 2. Aprire un circuito di esempio qualsiasi 3. Compilarlo		
Risultati attesi:	La compilazione avviene senza alcun errore o problema		

Test Case: TC-003 **Riferimento**: REQ-005		**Nome:**	Uno dei circuiti deve accendere il LED quando viene premuto il bottone
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Bottone deve permettere di premere un interruttore per accendere il LED, che si deve spegnere una volta rilasciato il pulsante.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad un interruttore e un LED (vedi Figura 8: Bottone - LED, schema di circuito) - Una versione della libreria “ButtonLib” implementata nel progetto - Una versione della libreria “LedLib” implementata nel progetto		
Procedura:	1. Avviare il codice di esempio “LedOnOff.ino” 2. Tenere premuto l’interruttore collegato al Digispark 3. Rilasciare l’interruttore collegato al Digispark		
Risultati attesi:	Quando l’interruttore viene premuto, il LED si accende fino al momento in cui il pulsante non viene rilasciato, dopodiché si spegne.		

Test Case: TC-004 **Riferimento**: REQ-005		**Nome:**	Uno dei circuiti deve invertire lo stato del LED quando viene premuto il bottone
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Bottone deve permettere di premere un interruttore per invertire lo stato LED, che si deve accendere se è spento e spegnere se è acceso.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad un interruttore e un LED (vedi Figura 8: Bottone - LED, schema di circuito) - Una versione della libreria “ButtonLib” implementata nel progetto - Una versione della libreria “LedLib” implementata nel progetto		
Procedura:	1. Avviare il codice di esempio “LedToggle.ino” 2. Premere l’interruttore collegato al Digispark 3. Premere nuovamente l’interruttore collegato al Digispark		
Risultati attesi:	Quando l’interruttore viene premuto per la prima volta, il LED si accende fino al momento in cui il pulsante non viene premuto la seconda volta, dopodiché si spegne.		

Test Case: **Riferimento**:	TC-005 REQ-005	**Nome:**	Uno dei circuiti deve invertire lo stato del LED da quando viene premuto il bottone fino a quando non viene rilasciato
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Bottone deve permettere di tenere premuto un interruttore per cominciare a invertire lo stato LED velocemente fino a quando il pulsante non viene rilasciato, dopodiché il LED assumerà l'ultimo stato in cui si trova.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad un interruttore e un LED (vedi Figura 8: Bottone - LED, schema di circuito) - Una versione della libreria "ButtonLib" implementata nel progetto - Una versione della libreria "LedLib" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "LedFlickering.ino" 2. Premere l'interruttore collegato al Digispark 3. Rilasciare l'interruttore collegato al Digispark		
Risultati attesi:	Quando l'interruttore viene premuto il LED comincia a cambiare rapidamente il proprio stato, alternando tra acceso e spento fino al momento in cui il pulsante non viene rilasciato, dopodiché rimane sull'ultimo valore che ha assunto prima che fosse rilasciato il pulsante.		

Test Case: **Riferimento**:	TC-006 REQ-006	**Nome:**	Uno dei circuiti deve accendere o spegnere il LED in base al valore rilevato dalla fotocellula
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Fotocellula deve impostare lo stato di un LED in base al valore rilevato da una fotocellula: quando il valore misurato scende sotto una certa soglia il LED viene spento; quando il valore misurato sale al di sopra di una certa soglia il LED viene acceso.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad una fotocellula e un LED (vedi Figura 9: Fotocellula - LED, schema di circuito) - Una versione della libreria "PhotocellLib" implementata nel progetto - Una versione della libreria "LedLib" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "PhotocellThreshold.ino" 2. Oscurare la fotocellula posizionandovi sopra un materiale opaco 3. Rimuovere il materiale opaco dalla fotocellula		
Risultati attesi:	Quando la fotocellula viene oscurata il LED si spegne; quando viene rimosso il materiale opaco il LED si accende nuovamente.		

Test Case:	TC-007	**Nome:**	Uno dei circuiti deve regolare l'intensità del LED in base al valore rilevato dalla fotocellula.
Riferimento:	REQ-006		
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Fotocellula deve impostare la luminosità di un LED in base al valore rilevato da una fotocellula: maggiore è la luminosità rilevata, maggiore è la luminosità del LED.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad una fotocellula e un LED (vedi Figura 9: Fotocellula - LED, schema di circuito) - Una versione della libreria "PhotocellLib" implementata nel progetto - Una versione della libreria "LedLib" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "PhotocellToLEDIntensity.ino" 2. Oscurare la fotocellula posizionandovi sopra gradualmente un materiale opaco 3. Rimuovere gradualmente il materiale opaco dalla fotocellula		
Risultati attesi:	Quando il materiale opaco oscura completamente il sensore il LED è spento, man mano che il materiale viene rimosso il LED si accende con un'intensità sempre maggiore fino a quando il corpo opaco non oscura più la fotocellula e il LED brilla alla luminosità massima.		

Test Case:	TC-008	**Nome:**	Uno dei circuiti deve regolare la frequenza di lampeggiamento del LED in modo inversamente proporzionale al valore rilevato dalla fotocellula
Riferimento:	REQ-006		
Descrizione:	Uno dei circuiti di esempio per la coppia di attuatori LED – Fotocellula deve impostare la frequenza di lampeggiamento di un LED in modo inversamente proporzionale al valore rilevato dalla fotocellula: maggiore è la luminosità rilevata, minore è la velocità di sfarfallio del LED.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad una fotocellula e un LED (vedi Figura 9: Fotocellula - LED, schema di circuito) - Una versione della libreria "PhotocellLib" implementata nel progetto - Una versione della libreria "LedLib" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "PhotocellDelay.ino" 2. Oscurare la fotocellula posizionandovi sopra gradualmente un materiale opaco 3. Rimuovere gradualmente il materiale opaco dalla fotocellula		
Risultati attesi:	Quando il materiale opaco oscura completamente il sensore il LED rimane sull'ultimo stato che ha assunto, man mano che il materiale viene rimosso il LED si comincia a lampeggiare con una frequenza sempre maggiore fino a quando il corpo opaco non oscura più la fotocellula e il LED lampeggia alla velocità massima.		

Test Case:	TC-009	**Nome:**	Uno dei circuiti deve stampare sullo schermo LCD la scritta "Hello World".
Riferimento:	REQ-007		
Descrizione:	Uno dei circuiti di esempio per il display a cristalli liquidi (LCD) deve stampare la scritta "Hello, World" sul display LCD.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad uno Shift register e un display a cristalli liquidi (vedi Figura 12: LCD, schema di circuito) - Una versione della libreria "LcdLib" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "LCDHelloWorld.ino"		
Risultati attesi:	Il display a cristalli liquidi (LCD) mostra la scritta "Hello World".		

Test Case: **Riferimento**:	TC-010 REQ-007	**Nome:**	Uno dei circuiti deve stampare del testo all'interno del display LCD e mostrare il lampeggiamento del cursore
Descrizione:	Uno dei circuiti di esempio per il display a cristalli liquidi (LCD) deve mostrare una stringa di testo all'interno del display LCD ed evidenziare il lampeggiamento del cursore		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad uno Shift register e un display a cristalli liquidi (vedi Figura 12: LCD, schema di circuito) - Una versione della libreria "LiquidCrystal_I2C" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "LCDCursorBlink.ino"		
Risultati attesi:	Il display a cristalli liquidi (LCD) mostrera la scritta "Non blinka!" con un cursore che rimane acceso per 5 secondi, dopodiché mostra il testo "Blinka!" e il cursore passa da acceso a spento a intervalli regolari. Questi due stati del display si alternano a intervalli di 5 secondi.		

Test Case: **Riferimento**:	TC-011 REQ-007	**Nome:**	Uno dei circuiti da realizzare deve stampare del testo sullo schermo LCD e dargli un effetto di scorrimento attraverso il display.
Descrizione:	Uno dei circuiti di esempio per il display a cristalli liquidi (LCD) deve stampare del testo sul display dandogli un effetto di scorrimento lungo la sua superficie da un lato verso l'altro.		
Prerequisiti:	- Un montaggio elettrico contenente un Digispark USB collegato ad uno Shift register e un display a cristalli liquidi (vedi Figura 12: LCD, schema di circuito) - Una versione della libreria "LiquidCrystal_I2C" implementata nel progetto		
Procedura:	1. Avviare il codice di esempio "LCDScroll.ino"		
Risultati attesi:	Il display a cristalli liquidi (LCD) mostra la scritta "Hello, World" che scorre da un lato del display verso l'altro tornando al lato inziale quando esce dal LCD.		

4.2 Risultati test

Test case	Risultato
TC-001	Il LED sul Digispark si accende e si spegne a intervalli regolari.
TC-002	La compilazione avviene senza alcun errore o problema
TC-003	Quando l'interruttore viene premuto, il LED si accende fino al momento in cui il pulsante non viene rilasciato, dopodiché si spegne.
TC-004	Quando l'interruttore viene premuto per la prima volta, il LED si accende fino al momento in cui il pulsante non viene premuto la seconda volta, dopodiché si spegne.
TC-005	Quando l'interruttore viene premuto il LED comincia a cambiare rapidamente il proprio stato, alternando tra acceso e spento fino al momento in cui il pulsante non viene rilasciato, dopodiché rimane sull'ultimo valore che ha assunto prima che fosse rilasciato il pulsante.
TC-006	Quando la fotocellula viene oscurata il LED si spegne; quando viene rimosso il materiale opaco il LED si accende nuovamente.
TC-007	Quando il materiale opaco oscura completamente il sensore il LED è spento, man mano che il materiale viene rimosso il LED si accende con un'intensità sempre maggiore fino a quando il corpo opaco non oscurerà più la fotocellula e il LED brilla alla luminosità massima.
TC-008	Quando il materiale opaco oscura completamente il sensore il LED rimane sull'ultimo stato che ha assunto, man mano che il materiale viene rimosso il LED si comincia a lampeggiare con una frequenza sempre maggiore fino a quando il corpo opaco non oscurerà più la fotocellula e il LED lampeggerà alla velocità massima.
TC-009	Il display a cristalli liquidi (LCD) mostra la scritta "Hello World".
TC-010	Il display a cristalli liquidi (LCD) mostrerà la scritta "Non blinka!" con un cursore che rimane acceso per 5 secondi, dopodiché mostrerà il testo "Blinka!" e il cursore passera da acceso a spento a intervalli regolari. Questi due stati del display si alternano a intervalli di 5 secondi.
TC-011	Il display a cristalli liquidi (LCD) mostrerà la scritta "Hello, World" che scorre da un lato del display verso l'altro tornando al lato inziale quando esce dal LCD.

5 Consuntivo

Non abbiamo riscontrato particolari problemi nella gestione del tempo, infatti durante tutto il progetto siamo stati in orario con la pianificazione e siamo riusciti a finire in tempo per la consegna senza difficoltà. Infatti il Gantt di pianificazione e quello consuntivo corrispondono.

6 Conclusioni

Essendo pensata per i programmatori alla prime armi o, addirittura, che non si sono mai affacciati al mondo della programmazione, questo progetto avrà un impatto piuttosto grande in chiunque si trovi in questa posizione e potrebbe portare loro a cominciare a programmare per passione e interessarsene sempre di più. Arduino potrebbe essere un primo passo verso linguaggi più usati, come Java o C++, da progetti semplici come Hello World a applicazioni web con integrazione a un database. Oppure l'esatto contrario: grazie a queste librerie qualcuno potrebbe capire che la programmazione non gli piace e cambiare strada. I risultati sono generali e facilmente implementabili in qualsiasi tipo di progetto Arduino che faccia utilizzo di uno o più degli attuatori descritti in queste librerie.

6.1 Sviluppi futuri

Eventuali sviluppi futuri includono l'aggiunta di nuovi circuiti di esempio per gli attuatori già presenti, come un circuito che unisca un bottone con un display a cristalli liquidi; di nuovi attuatori con conseguenti librerie e, di conseguenza, circuiti di esempio, magari il potenziometro o il sensore di ultrasuoni; e di ulteriori funzionalità per le librerie implementate, per esempio si potrebbe aggiungere un metodo alla libreria del display LCD che mostri un'animazione di scrittura "lettera per lettera" di un testo passato come parametro.

6.2 Considerazioni personali

In questo progetto abbiamo imparato a realizzare una libreria per Arduino e le basi della programmazione a oggetti in C++, che si è rivelata molto più simile a Java e C# di quanto pensassimo.

7 Bibliografia

7.1 Sitografia

- http://digistump.com/products/1
  Digispark USB Development Board – Digistump
  14 novembre 2018
- https://digistump.com/wiki/digispark/tutorials/basics
  digispark:tutorials:basics [Digistump Wiki]
  14 novembre 2018
- https://www.adrirobot.it/arduino/digispark/digispark.htm
  Scheda Digispark
  16 novembre 2018
- 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
- https://digistump.com/wiki/digispark/tutorials/lcd
  digispark:tutorials:lcd [Digistump Wiki]
  14 dicembre 2018
- https://www.draw.io/
  Draw.io
  1 febbraio 2019

8 Allegati

- Diari di lavoro
- Guide di utilizzo dei circuiti di esempio

SECONDO PROGETTO | Diario di lavoro - 14.11.2018

Matan Davidi, Filippo Finke

Trevano, 14.11.2018

Lavori svolti

Oggi ci è stato consegnato il quaderno dei compiti del nuovo progetto, e ci è stata data la scelta tra la realizzazione di due progetti:

- Sistema didattico per Arduino con libreria per attuatori e relativa documentazione
- Sistema didattico per LEGO Mindstorm EV3 / NTX con libreria e relativa documentazione

Noi abbiamo scelto il primo perché ci troviamo entrambi più comodi a programmare per Arduino piuttosto che in Java per Mindstorm, inoltre non disponiamo di un kit LEGO Mindstorm per poter, eventualmente, lavorare a casa.

Mentre uno lavorava alla creazione del repository, della struttura delle cartelle, del file README e delle domande per la prossima lezione da consegnare al cliente riguardo il quaderno dei compiti, l'altro assisteva alla presentazione di alcuni compagni sul progetto appena concluso.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Consegnare la lista di domande realizzata in classe al cliente (docente).

SECONDO PROGETTO | Diario di lavoro - 16.11.2018

Matan Davidi, Filippo Finke

Trevano, 16 Novembre 2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Verifica teorica
15:00 - 16:30	Risposte alle domande

Durante la lezione abbiamo svolto la verifica e abbiamo ricevuto le risposte alle domande riguardanti il QDC.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Consegnare la lista di domande realizzata in classe al cliente (docente).

SECONDO PROGETTO | Diario di lavoro - 21.11.2018

Matan Davidi, Filippo Finke

Trevano, 21 Novembre 2018

Lavori svolti

Oggi Matan ha cominciato a documentare il proprio progetto, svolgendo i capitoli fino al punto 1.2, "Abstract", mentre Filippo realizzava la progettazione tramite diagramma di Gantt.

Problemi riscontrati e soluzioni adottate

Non sapendo quante librerie e circuiti di esempio realizzare, si è presentato il problema di pianificare il progetto. Alla fine abbiamo optato per una pianificazione parziale che modificheremo andando avanti con il progetto

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Finire il punto 1.2 della documentazione (Abstract).

SECONDO PROGETTO | Diario di lavoro - 23.11.2018

Matan Davidi, Filippo Finke

Trevano, 23 Novembre 2018

Lavori svolti

Oggi Matan ha continuato la documentazione, finendo il capitolo 1.2, "Abstract", e 1.3, "Scopo", mentre Filippo ha cominciato ad implementare la libreria iniziando da quella del bottone, che ha chiamato ButtonLib:
la libreria in questo momento contiene il codice, spiegato in seguito, un esempio del codice di un montaggio che utilizza il bottone ed una bozza di documentazione sotto forma di file README.
Questa libreria contiene i metodi:

```
bool getState()
```

che restituisce lo stato del bottone sotto forma di valore booleano (vero --> premuto, falso --> rilasciato)

Inoltre Filippo ha proseguito l'implementazione realizzando una libreria per i LED, chiamata LedLib:
anche questa libreria
questa libreria contiene i metodi:

```
void on()
```

che accende il LED

```
void off()
```

che spegne il LED

```
void toggle()
```

che inverte lo stato del LED (spento --> acceso, acceso --> spento)

```
bool getState()
```

che restituisce lo stato del LED sotto forma di valore booleano (vero --> acceso, falso --> spento)

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Pensare e documentare i tre o più esempi per le librerie ButtonLib e LedLib.

SECONDO PROGETTO | Diario di lavoro - 28.11.2018

Matan Davidi, Filippo Finke

Trevano, 28 Novembre 2018

Lavori svolti

Oggi Matan ha proseguito con la documentazione, completando il capitolo 1.4. - "Analisi del dominio" e cominciando il capitolo 1.5 - "Analisi e specifica dei requisiti". Durante questo tempo Filippo ha presentato il progetto scorso ai docenti e non ha quindi potuto lavorare.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Pensare e documentare i tre o più esempi per le librerie ButtonLib e LedLib.

SECONDO PROGETTO | Diario di lavoro - 30.11.2018

Matan Davidi, Filippo Finke

Trevano, 30 Novembre 2018

Lavori svolti

Matan ha continuato con la documentazione e ha cominciato e finito il capitolo 1.7 - Analisi dei mezzi
Filippo ha cominciato l'implementazione dei suoi esempi per la libreria del LED:

- LedOnOff: Legge lo stato del bottone e accende il LED quando il bottone è premuto
- LedToggle: Legge lo stato del bottone e inverte lo stato del LED quando il bottone viene premuto
- LedFlickering: Legge lo stato del bottone e inverte ripetutamente lo stato del LED, in modo che esso passi velocemente tra acceso e spentos

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 05.11.2018

Matan Davidi, Filippo Finke

Trevano, 05 Novembre 2018

Lavori svolti

Oggi abbiamo chiesto ai docenti le prossime coppie di componenti da implementare:

- Una cellula fotovoltaica e un LED
- Un sensore infrarosso e un LED

Ci è stato imposto di fare in modo che quando la luce è leggermente sopra o leggermente sotto la soglia tra 0 e 1, quando normalmente uscirebbe un output che varia velocemente tra 0 e 1, produca un output fisso a 0 oppure 1.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in anticipo di un giorno rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 07.11.2018

Matan Davidi, Filippo Finke

Trevano, 07 Novembre 2018

Lavori svolti

Filippo ha proseguito l'implementazione della libreria per la fotocellula iniziata la settimana scorsa, PhotocellLib.
Questa libreria implementa il seguente metodo:

```
int Photocell::getLux()
```

che legge lo stato della fotocellula e ritorna un valore tra 0 e 1023 corrispondente al valore rilevato dalla fotocellula.

Inoltre Filippo ha realizzato due esempi di circuiti che fanno uso di PhotocellLib:

PhotocellThreshold: un LED viene acceso o spento in base al valore rilevato dalla fotocellula.

PhotocellToLEDIntensity: un LED RGB ha un'intensità pari al valore rilevato dalla fotocellula.

Matan nel frattempo ha continuato a scrivere la documentazione, aggiungendo cinque requisiti al capitolo 1.5 - Analisi e specifica dei requisiti, aggiornando l'immagine del diagramma di Gantt del capitolo 1.6 - Pianificazione e cominciando il capitolo 2.4 - Design procedurale, che contiene la descrizione dei file contenuti in tutte le librerie (.cpp, .h e keyw ords) e una descrizione di tutti i metodi contenuti in esse.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Pensare e realizzare il terzo e ultimo esempio di circuito che utilizza la combinazione Fotocellula-LED.

SECONDO PROGETTO | Diario di lavoro - 12.12.2018

Matan Davidi, Filippo Finke

Trevano, 12 Dicembre 2018

Lavori svolti

Oggi ci è stato imposto il lavoro di far funzionare i primi programmi di esempio presenti all'interno dell'IDE di Arduino sul nostro Digispark.

In seguito Filippo ha pensato e implementato l'esempio rimanente per la libreria della coppia Fotocellula-LED: un esempio che 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.

Matan ha finito il capitolo 2.1 - "Design dell'architettura del sistema" della documentazione.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Richiedere la prossima coppia di componenti di cui scrivere la libreria

SECONDO PROGETTO | Diario di lavoro - 14.12.2018

Matan Davidi, Filippo Finke

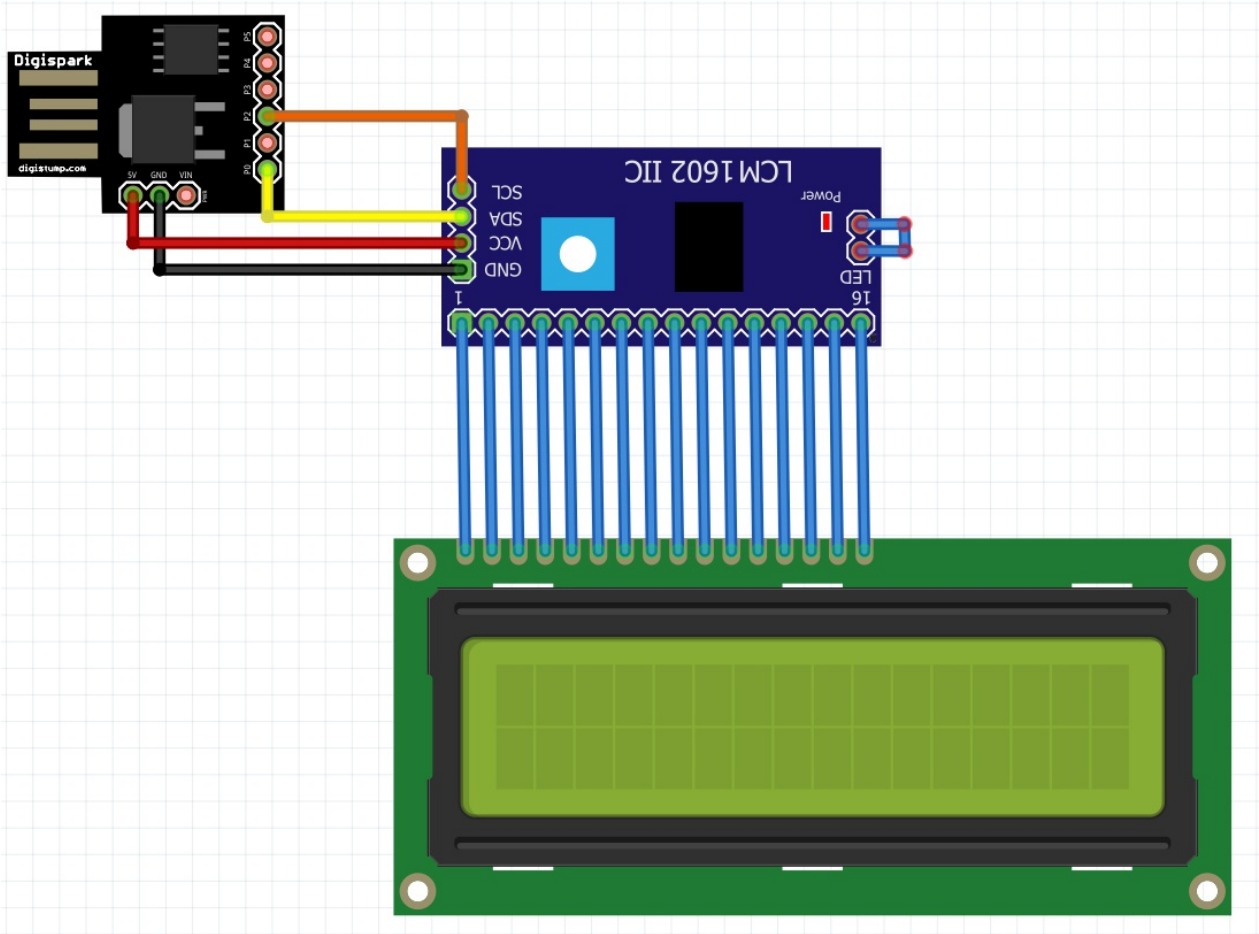
Trevano, 14 Dicembre 2018

Lavori svolti

Oggi i clienti (docenti) ci hanno commissionato una nuova libreria e relativi circuiti: il display a cristalli liquidi (LCD).
Filippo ha cominciato a cercare di implementare un circuito di prova con il display LCD. In seguito ha pensato, progettato e implementato i tre circuiti di esempio per il componente:

- LCDCursorBlink
- LCDHelloWorld
- LCDScroll

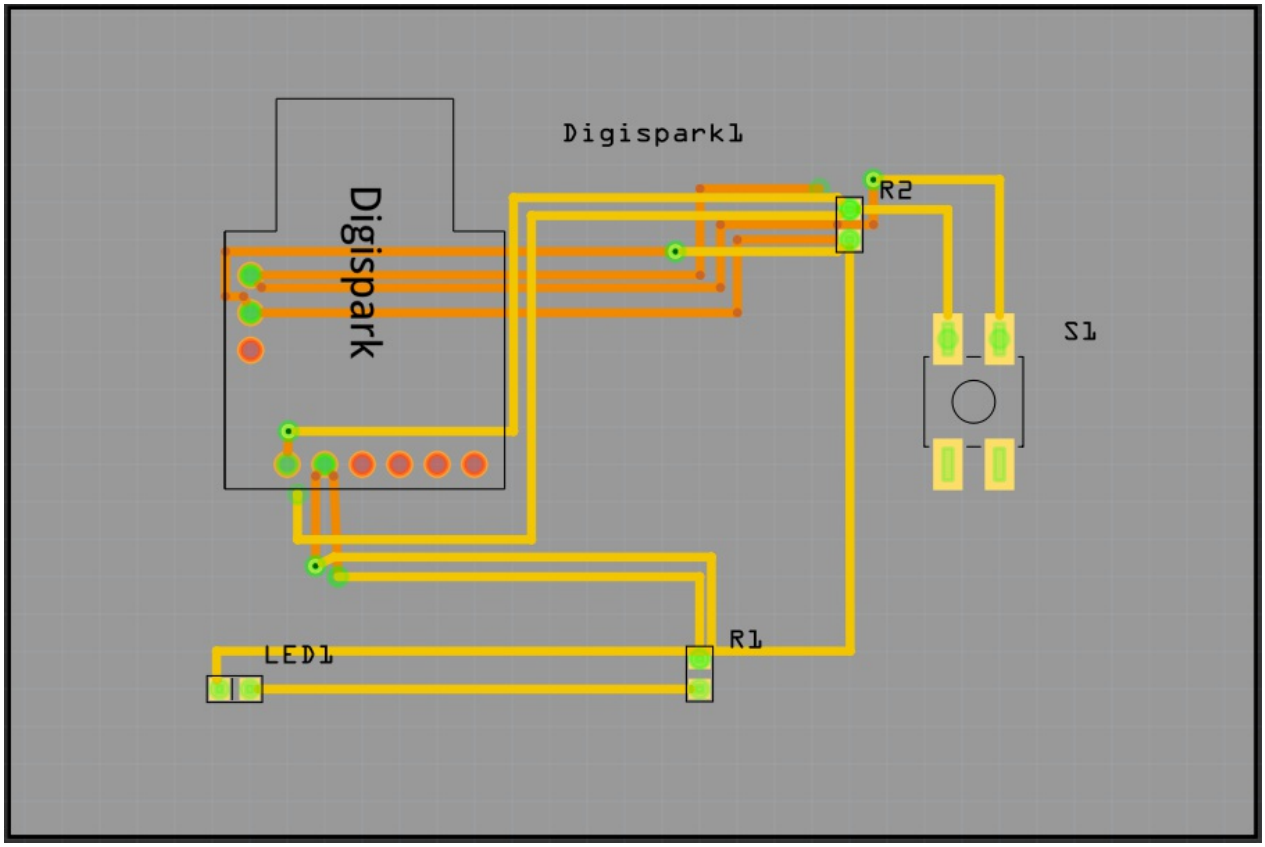
Tutti e tre gli esempi utilizzano lo stesso circuito, ossia il seguente:



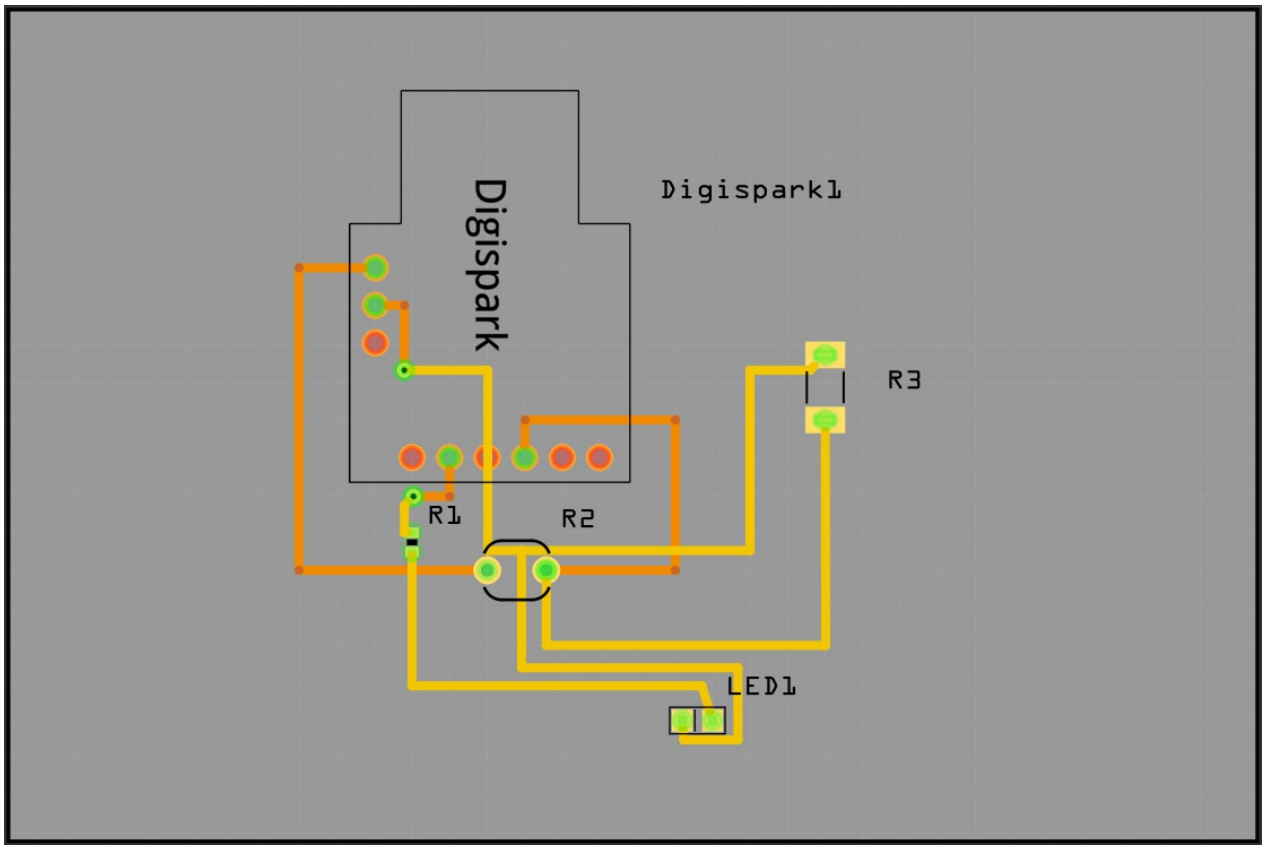
Come è possibile vedere, il circuito consiste in un Digispark USB collegato ad uno shift register, a sua volta attaccato ad un display a cristalli liquidi. È importante notare come l'alimentazione dello shift register è collegata a sé stessa.

Matan intanto ha generato gli schemi PCB per i tre circuiti esistenti, ButtonLed, PhotocellLed e LCD.

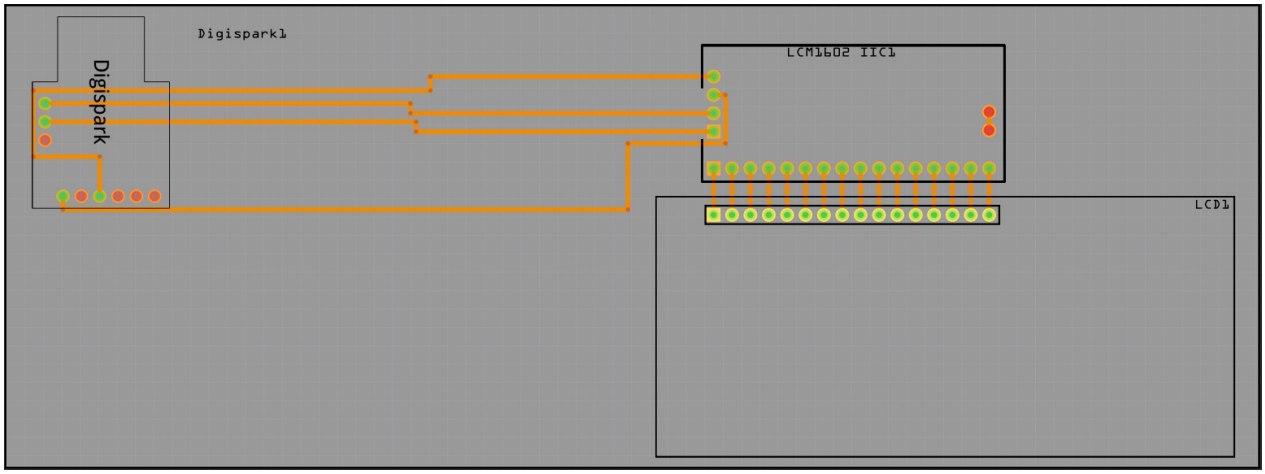
ButtonLed:



PhotocellLed:



LCD:



Inoltre, ha completato il capitolo 2 della documentazione, rimuovendo i sottocapitoli 2.2, "Design dei dati e database" e 2.3, "Design delle interfacce", iniziando il capitolo 3, quindi la documentazione dell'implementazione, documentando la prima classe, "3.1 ButtonLib".

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 19.12.2018

Matan Davidi, Filippo Finke

Trevano, 19 Dicembre 2018

Lavori svolti

Oggi Matan è andato avanti con la documentazione del capitolo 3, aggiungendo le revisioni al file della documentazione in modo da permettere la modifica da parte di entrambi i membri del gruppo.

Filippo ha aggiunto l'esempio rimanente alla libreria del display a cristalli liquidi: LCDSCroll. Questo esempio scrive una stringa di testo sul display a cristalli liquidi che viene traslata in orizzontale lungo la superficie del display.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 21.12.2018

Matan Davidi, Filippo Finke

Trevano, 21 Dicembre 2018

Lavori svolti

Oggi abbiamo cominciato e terminato la teoria sulle presentazioni.
Per iniziare il docente ci ha mostrato una presentazione su PowerPoint contenente gli tutti gli errori da non commettere quando si realizza una presentazione del proprio prodotto, come non inserire troppi dati nella stessa diapositiva e non usare troppe Word Art. In seguito abbiamo assistito alle presentazioni del primo progetto prima di Carlo Pezzotti, poi di Mattia Lazzaroni.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

In teoria abbiamo perso una lezione di lavoro, quindi siamo indietro di una lezione?

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 09.01.2019

Matan Davidi, Filippo Finke

Trevano, 9 Gennaio 2019

Lavori svolti

Durante la lezione di oggi, Matan ha proseguito con la documentazione di progetto, completando il capitolo 3: Implementazione.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in ***perfetto*** orario con la pianificazione

Programma di massima per la prossima giornata di lavoro

Richiedere il prossimo componente di cui realizzare la libreria e implementarla, oppure andare avanti con la documentazione.

SECONDO PROGETTO | Diario di lavoro - 11.01.2019

Matan Davidi, Filippo Finke

Trevano, 11 Gennaio 2019

Lavori svolti

Oggi abbiamo messo a punto le guide di utilizzo delle librerie sviluppate.
Sono state modificati i file README.md delle librerie:

- ButtonLib
- LedLib
- PhotocellLib

Le modifiche includono una modifica della descrizione per renderla più esaustiva, un'aggiunta di una piccola procedura da seguire per inserire le librerie all'interno del proprio progetto, una descrizione più informativa dei parametri usati per istanziare un oggetto e, infine, è stato sistemato un errore di formattazione del testo nei file che rendeva i prototipi dei metodi scritti con uno stile normale, mentre la loro descrizione scritta con uno stile da codice.

È inoltre stato creato un file README.md contenente la guida di utilizzo per la libreria LcdLib, attualmente non ancora aggiunta al progetto su GitHub.

Inoltre Matan ha aggiornato due diari di lavoro tenuti in precedenza, ossia quello della giornata di lavoro del 21 novembre e del 14 dicembre 2018, perché ritenuti troppo scarni e vuoti.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Finire di aggiornare eventuali diari di lavoro fatti male in passato

SECONDO PROGETTO | Diario di lavoro - 16.01.2019

Matan Davidi, Filippo Finke

Trevano, 16 Gennaio 2019

Lavori svolti

Durante la giornata di oggi, Matan ha finito di aggiornare tutti i diari di lavoro tenuti in precedenza, perché ritenuti troppo scarni e vuoti. Dopo qualche settimana di lavoro, il 14 novembre 2018, ho incominciato ad adottare Atom come editore di testo, rimpiazzando Visual Studio Code che avevo usato fino a quel punto. L'estensione usata per esportare i file markdown, che uso per redigere i diari di lavoro come quello che state leggendo adesso, utilizza un formato diverso tra Atom e Visual Studio Code, quindi i diari di lavoro fino al 14 novembre utilizzano un'impaginazione diversa rispetto a quelli successivi, quindi ho avuto un motivo in più per esportarli nuovamente. Infine, il diario di lavoro della giornata del 16 novembre 2018 era stato fatto da Filippo, quindi utilizzava una formattazione ancora diversa, quindi sono stato obbligato ad aggiornarlo.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

SECONDO PROGETTO | Diario di lavoro - 18.01.2019

Matan Davidi, Filippo Finke

Trevano, 18 Gennaio 2019

Lavori svolti

Oggi Filippo ha dato un'occhiata alla documentazione redatta fino a questo punto, correggendo eventuali errori e aggiungendo un requisito relativo alla libreria sul display a segmenti liquidi, con i successivi esempi di implementazione, esattamente come le librerie documentate in precedenza. Inoltre ha descritto brevemente i circuiti già presenti nel capitolo 2, Progettazione, aggiungendo anche le immagini dello schema di circuito ed elettrico del circuito di esempio per la libreria del display a segmenti liquidi. In seguito ha proseguito con la documentazione del capitolo 3, Implementazione, inserendo la realizzazione concreta, a livello di codice, con esempi di codice.
Solo in seguito (a causa del problema menzionato nella sezione "Problemi riscontrati e soluzioni adottate") Matan ha revisionato i cambiamenti apportati, eventualmente mettendo o togliendo del testo ove gli pareva necessario. Per esempio, ha modificato la descrizione di due dei sottorequisiti del requisito con ID "REQ-007", cambiando il suo titolo in modo che fosse più esplicativo.

Problemi riscontrati e soluzioni adottate

Come al solito, c'è il problema che solo una persona del gruppo può lavorare alla documentazione perché Microsoft Word salva i file in formato binario, non di testo, quindi Git non riesce a vedere le differenze e rimpiazza il file vecchio con quello appena caricato sul repository di GitHub. Se i due file contengono il lavoro di due persone diverse, è come se quello che ha caricato per primo non avesse modificato niente.

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare con la documentazione del progetto, cominciando il capitolo 4, Test, in cui documentare la verifica del corretto funzionamento di tutti i circuiti realizzati.

SECONDO PROGETTO | Diario di lavoro - 23.01.2019

Matan Davidi, Filippo Finke

Trevano, 23 Gennaio 2019

Lavori svolti

Durante la lezione di oggi, Filippo ha aggiornato il diagramma di Gantt utilizzato per la pianificazione del progetto, aggiungendo le fasi e le attività relative all'ultima libreria e all'ultimo circuito di esempio, ossia quello del display a segmenti liquidi (LCD).

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Adesso siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare con la documentazione del progetto, cominciando il capitolo 4, Test, in cui documentare la verifica del corretto funzionamento di tutti i circuiti realizzati.

SECONDO PROGETTO | Diario di lavoro - 25.01.2019

Matan Davidi, Filippo Finke

Trevano, 25 Gennaio 2019

Lavori svolti

Oggi Filippo ha aggiunto una guida di utilizzo per i circuiti di esempio contenuti all'interno della cartella "Examples", quindi:

- LedFlickering
- LedOnOff
- LedToggle
- LCDCursorBlink
- LCDHelloWorld
- LCDScroll
- PhotocellDelay
- PhotocellThreshold
- PhotocellToLedIntensity

Queste guide di utilizzo contengono una breve descrizione del circuito, il materiale che richiede la sua realizzazione, una foto dello schema di circuito con una descrizione e il codice che richiede il suo utilizzo.

Una guida di utilizzo ha questo aspetto (la guida presa in esempio è quella per il circuito "LedFlickering"):

Esempio LedFlickering

Descrizione

Circuito nel quale viene mostrato l'utilizzo di un bottone per fare lampeggiare un led.

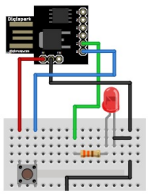
Materiale

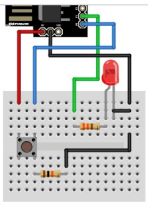
- Un Digispark
- Un led
- Un bottone
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All'interno del circuito sono presenti quattro componenti: 1 bottone, 1 LED e 2 resistenze. Il bottone è in pull-down tramite una resistenza da 10kΩ, il suo stato viene letto attraverso il pin "P0" del micro controllore. Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω. Quando il bottone viene premuto il LED al pin "P1" inizia ad accendersi e spegnersi ad un intervallo di 100 millisecondi. Appena il bottone viene rilasciato il LED viene spento.





Codice

```
/* Includo la libreria button */
#include <button.h>
/* Includo la led button */
#include <led.h>

/* Istanzio un oggetto di tipo led */
Led led(1);
/* Istanzio un oggetto di tipo button */
Button b(0);

/* Metodo che viene chiamato solo all'avvio */
void setup() {
}

/* Metodo che viene chiamato all'infinito */
void loop() {
  /* Prendo lo stato del bottone */
  bool state = b.getState();
  /* Se il bottone è premuto lo faccio lampeggiare, sennò lo spengo. */
  if(state) {
    led.toggle();
    /* Aspetto 100 millisecondi */
    delay(100);
  } else {
    led.off();
  }
}
```

Nel frattempo Matan ha continuato a documentare il progetto, iniziando il capitolo 4, Test, aggiungendo i primi protocolli di test relativi al requisito REQ-005, contenenti la verifica del corretto funzionamento dei circuiti di esempio per la coppia di attuatori LED - Bottone. Inoltre ha sistemato dei piccoli errori di impaginazione in tutto il documento.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Proseguire con la documentazione del capitolo 4, Test, in cui documentare la verifica del corretto funzionamento di tutti i circuiti realizzati.

SECONDO PROGETTO | Diario di lavoro - 30.01.2019

Matan Davidi, Filippo Finke

Trevano, 30 Gennaio 2019

Lavori svolti

Oggi Filippo ha rinnovato il diagramma di Gantt utilizzato per la pianificazione del progetto, aggiungendo una fase per la realizzazione delle guide di utilizzo create durante l'ultima lezione. Inoltre ha riguardato queste guide sistemando lo stile e l'impaginazione. In seguito ha documentato i file header delle librerie implementate, ossia:

- button.h
- led.h
- photocell.h

Per il resto della lezione ha lavorato alla presentazione del progetto.

Nel frattempo Matan ha continuato a documentare il progetto, proseguendo con il capitolo 4, Test, aggiungendo i protocolli di test relativi al requisito REQ-006, contenenti la verifica del corretto funzionamento dei circuiti di esempio per la coppia di attuatori LED - Fotocellula, e quelli relativi al REQ-007, contenenti la verifica del corretto funzionamento dei circuiti di esempio per il display a cristalli liquidi (LCD). Poi ha proseguito cominciando e finendo il capitolo 5, "Consuntivo", e iniziando e terminando la documentazione del capitolo 6, "Conclusioni".

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Terminare la documentazione e la presentazione del progetto.

SECONDO PROGETTO | Diario di lavoro - 01.02.2019

Matan Davidi, Filippo Finke

Trevano, 1 Febbraio 2019

Lavori svolti

Il lavoro di oggi consisteva nel proseguire con la realizzazione della presentazione e nel concludere la documentazione di progetto, lavori svolti rispettivamente da Filippo e Matan.

In seguito Filippo ha documentato i metodi della libreria del display LCD, finora priva di documentazione perché la libreria utilizzata per interfacciarsi al LCD, "LiquidCrystal_I2C", rende inutile la creazione di una libreria nostra che richiama i suoi metodi. Quindi, Filippo ha cercato la documentazione di LiquidCrystal_I2C traducendola e aggiustandola ove opportuno.

Matan invece ha dato un'ultima passata alla documentazione completandola e sistemandola dove necessario. Per esempio, ha finito gli ultimi capitoli iniziati l'ultima lezione e ha aggiunto una descrizione al capitolo 1.6, "Pianificazione", descrivendo ogni fase e ogni attività del diagramma di Gantt.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Terminare la presentazione del progetto.

SECONDO PROGETTO | Diario di lavoro - 06.02.2019

Matan Davidi, Filippo Finke

Trevano, 6 Febbraio 2019

Lavori svolti

Durante la lezione di oggi Matan e Filippo hanno dato l'ultima occhiata alla documentazione in modo che essa sia pronta per la consegna venerdì. Matan ha infatti aggiunto due protocolli di test prima di quelli già presenti: TC-001 che controlla se il Digispark USB funziona correttamente e TC-002 che verifica che le librerie siano state importate in modo giusto. Inoltre ha svolto alcuni leggeri cambiamenti:

- Rimosso la tabella dal capitolo 1.6, pianificazione
- Aggiunto draw.io, usato per creare i diagrammi UML delle classi, alla lista di software e alla sitografia
- Sistemata la didascalia sotto la figura 12, ossia lo schema di circuito del LCD, in modo che non si sovrapponesse al testo descrittivo del circuito
- Cambiato i valori di ritorno del metodo getLux da 0-1023 in 0-255
- Aggiunto le pagine del Digispark alla sitografia
- Piccole modifiche grammaticali

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione

Programma di massima per la prossima giornata di lavoro

Chiedere ai docenti cosa mettere sotto il capitolo 1.7.2, Hardware.

SECONDO PROGETTO | Diario di lavoro - 08.02.2019

Matan Davidi, Filippo Finke

Trevano, 8 Febbraio 2019

Lavori svolti

Oggi Matan e Filippo si sono assicurati che tutto fosse perfetto per la consegna sistemando la documentazione e la presentazione del progetto dove fosse necessario, creando il file PDF della documentazione e delle guide di utilizzo e unire insieme la documentazione, i diari di lavoro e le guide di utilizzo in un unico file PDF da consegnare.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Siamo in orario con la consegna del progetto

Programma di massima per la prossima giornata di lavoro

Esempio LedFlickering

Descrizione

Circuito nel quale viene mostrato l'utilizzo di un bottone per fare lampeggiare un led.

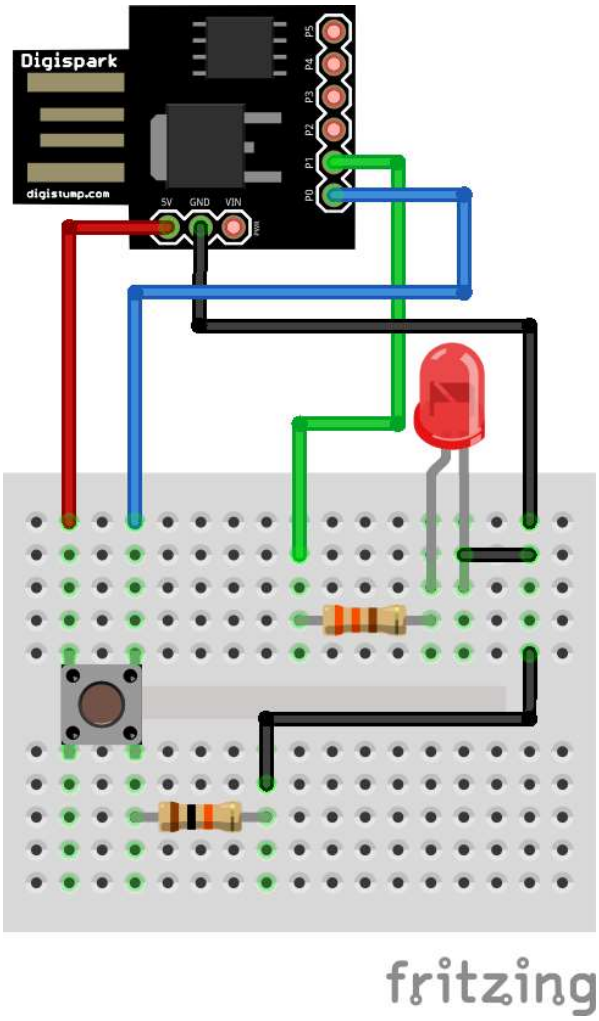
Materiale

- Un Digispark
- Un led
- Un bottone
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti quattro componenti: 1 bottone, 1 LED e 2 resistenze. Il bottone è in pull-down tramite una resistenza da 10kΩ, il suo stato viene letto attraverso il pin “P0” del micro controllore. Il LED è attaccato al pin “P1” del Digispark attraverso una resistenza da 330Ω. Quando il bottone viene premuto il LED al pin "P1" inizia ad accendersi e spegnersi ad un intervallo di 100 millisecondi. Appena il bottone viene rilasciato il LED viene spento.



Codice

```
/* Includo la libreria button */
#include <button.h>
/* Includo la led button */
#include <led.h>

/* Istanzio un oggetto di tipo led */
Led led(1);
/* Istanzio un oggetto di tipo button */
Button b(0);

/* Metodo che viene chiamato solo all'avvio */
void setup() {
}

/* Metodo che viene chiamato all'infinito */
void loop() {
  /* Prendo lo stato del bottone */
  bool state = b.getState();
  /* Se il bottone è premuto lo faccio lampeggiare, sennò lo spengo. */
  if(state) {
    led.toggle();
    /* Aspetto 100 millisecondi */
    delay(100);
  } else {
    led.off();
  }
}
```

Esempio LedOnOff

Descrizione

Circuito nel quale viene mostrato l'utilizzo di un bottone per accendere e spegnere un led.

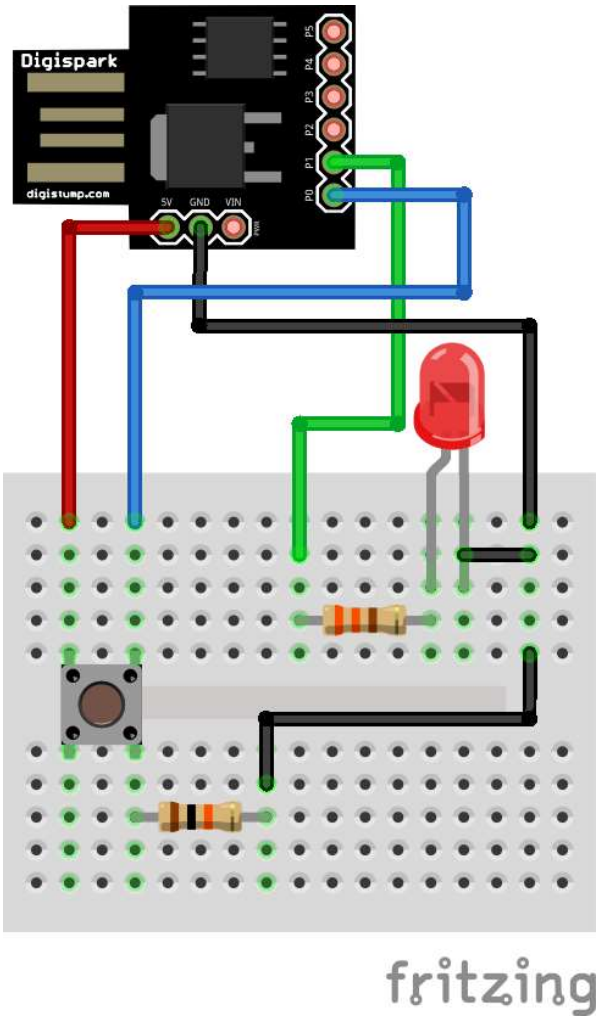
Materiale

- Un Digispark
- Un led
- Un bottone
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti quattro componenti: 1 bottone, 1 LED e 2 resistenze. Il bottone è in pull-down tramite una resistenza da 10kΩ, il suo stato viene letto attraverso il pin “P0” del micro controllore. Il LED è attaccato al pin “P1” del Digispark attraverso una resistenza da 330Ω. Quando il bottone viene premuto il LED al pin "P1" si accende. Appena il bottone viene rilasciato il LED viene spento.



Codice

```
/* Includo la libreria button */
#include <button.h>
/* Includo la led button */
#include <led.h>

/* Istanzio un oggetto di tipo led */
Led led(1);
/* Istanzio un oggetto di tipo button */
Button b(0);

/* Metodo che viene chiamato solo all'avvio */
void setup() {
}

/* Metodo che viene chiamato all'infinito */
void loop() {
  /* Prendo lo stato del bottone */
  bool state = b.getState();
  /* Se il bottone è premuto lo accendo, sennò lo spengo. */
  if(state) {
    led.on();
  } else {
    led.off();
  }
}
```

Esempio LedToggle

Descrizione

Circuito nel quale viene mostrato l'utilizzo di un bottone per accendere e spegnere un led con mantenimento.

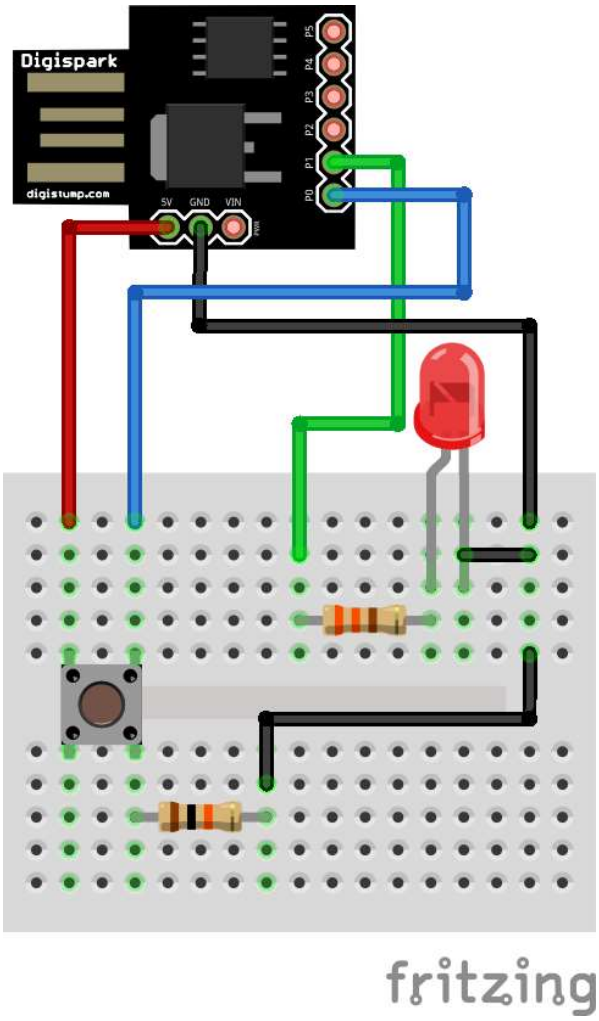
Materiale

- Un Digispark
- Un led
- Un bottone
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti quattro componenti: 1 bottone, 1 LED e 2 resistenze. Il bottone è in pull-down tramite una resistenza da 10kΩ, il suo stato viene letto attraverso il pin “P0” del micro controllore. Il LED è attaccato al pin “P1” del Digispark attraverso una resistenza da 330Ω. Quando il bottone viene premuto il LED al pin "P1" si accende e se si rilascia il bottone il LED rimane acceso. Se si ripreme sul bottone il LED si spegne.



Codice

```
/* Includo la libreria button */
#include <button.h>
/* Includo la led button */
#include <led.h>

/* Istanzio un oggetto di tipo led */
Led led(1);
/* Istanzio un oggetto di tipo button */
Button b(0);

/* Variabile nel quale viene salvato l'ultimo stato del bottone. */
bool lastButtonState = LOW;

/* Variabile nel quale viene salvato il tempo del click sul bottone. */
int time = millis();

/* Metodo che viene chiamato solo all'avvio */
void setup() {
}

/* Metodo che viene chiamato all'infinito */
void loop() {
  /* Prendo lo stato del bottone */
  bool state = b.getState();

  /* Se il bottone è premuto accendo il led e al prossimo cambiamento spengo il led. */
  if(state && !lastButtonState && millis() - time > 200) {
    lastButtonState = HIGH;
    led.toggle();
    time = millis();
  } else if(!state && lastButtonState) {
    lastButtonState = LOW;
  }
}
```

Esempio LCDCursorBlink

Descrizione

Circuito che permette di mostrare la posizione del cursore all'interno del display LCD.

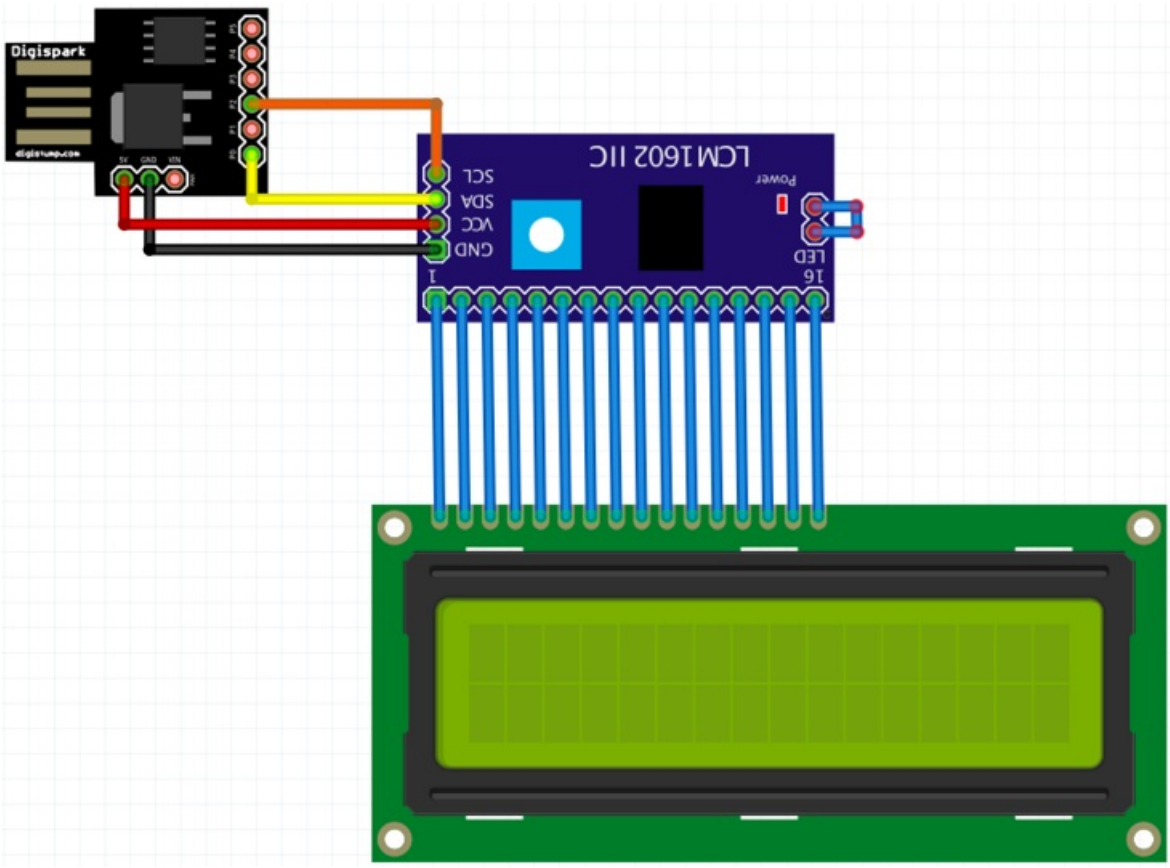
Materiale

- Un Digispark
- Un LCD
- Uno shift register

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti due componenti: 1 display a cristalli liquidi (LCD) e 1 shift register. Lo shift register viene alimentato dal digispark ed è collegato ai relativi pin SCL(“P2”) e SDA(“P0”). Mentre il display LCD è collegato interamente allo shift register. Il display stampa del testo ad un intervallo specifico mostrando la posizione del cursore.



Codice

```
/**
 * Includo le librerie.
 */
#include <TinyWireM.h>
#include <LiquidCrystal_I2C.h>

/**
 * Istanzio un oggetto di tipo LCD all'indirizzo 0x27, con 16 caratteri e due righe.
 */
LiquidCrystal_I2C lcd(0x27,16,2);

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
```

```
void setup() {
  //Avvio la comunicazione I2C.
  TinyWireM.begin();
  //Preparo il display LCD.
  lcd.init();
  //Accendo la retroilluminazione.
  lcd.backlight();
}

/**
 * Metodo che viene eseguito all'infinito.
 */
 void loop() {
   //Disabilito lo sfarfallio del cursore.
   lcd.noBlink();
   //Stampo all'interno del display la scritto "Non blinka!" e aspetto 5 secondi.
   printMessage("Non blinka!", 5);
   //Abilito lo sfarfallio del cursore.
   lcd.blink();
   //Stampo all'interno del display la scritto "Blinka!" e aspetto 5 secondi.
   printMessage("Blinka!", 5);
 }
 /**
  * Metodo utilizzato per stampare all'interno del display con un timer.
  *
  * @param message Il messaggio da stampare.
  * @param seconds Quanti secondi aspettare.
  */
 void printMessage(String message, int seconds) {
  //Per ogni secondo eseguo:
  for(int x = seconds; x > 0; x--) {
      //Cancello le scritte all'interno del display.
      lcd.clear();
      //Imposto il cursore in alto a sinistra.
      lcd.setCursor(0, 0);
      //Scrivo nel display il messaggio.
      lcd.print(message);
      //Mi sposto in basso a sinistra.
      lcd.setCursor(0, 1);
      //Stampo un messaggio di attesa.
      lcd.print("Aspetto " + String(x) + " sec!");
      //Sposto il cursore nella posizione 10 della prima riga.
      lcd.setCursor(10,0);
      //Aspetto 1 secondo.
      delay(1000);
  }
}
```

Esempio LCDHelloWorld

Descrizione

Circuito che permette di mostrare del testo all'interno del display LCD.

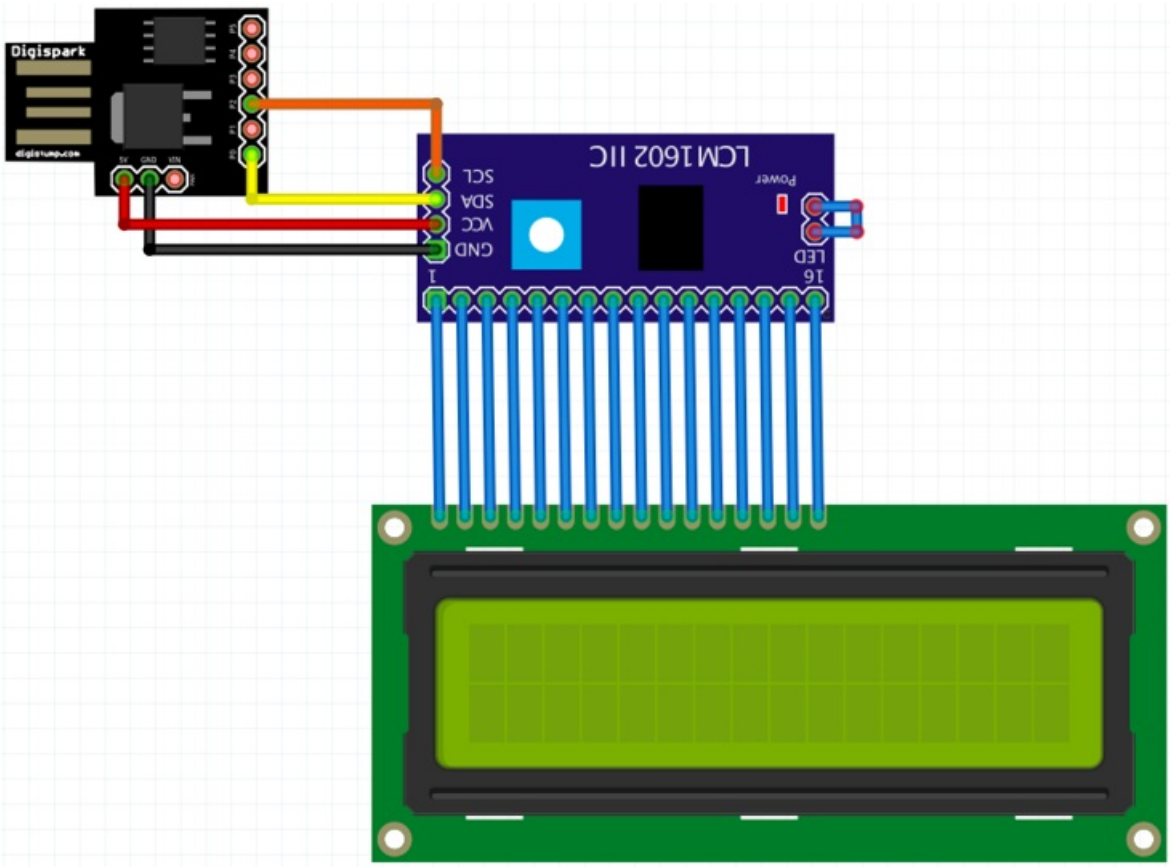
Materiale

- Un Digispark
- Un LCD
- Uno shift register

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti due componenti: 1 display a cristalli liquidi (LCD) e 1 shift register. Lo shift register viene alimentato dal digispark ed è collegato ai relativi pin SCL(“P2”) e SDA(“P0”). Mentre il display LCD è collegato interamente allo shift register. Il display LCD stampa la scritta "Hello World".



Codice

```
/**
 * Includo le librerie.
 */
#include <TinyWireM.h>
#include <LiquidCrystal_I2C.h>

/**
 * Istanzio un oggetto di tipo LCD all'indirizzo 0x27, con 16 caratteri e due righe.
 */
LiquidCrystal_I2C lcd(0x27,16,2);

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
```

```
void setup() {
  //Avvio la comunicazione I2C.
  TinyWireM.begin();
  //Preparo il display LCD.
  lcd.init();
  //Accendo la retroilluminazione.
  lcd.backlight();
  //Stampo all'interno del display LCD la scritta "Hello World!"
  lcd.print("Hello world!");
}

/**
 * Metodo che viene eseguito all'infinito.
 */
void loop() {

}
```

Esempio LCDScroll

Descrizione

Circuito che permette di mostrare come spostarre del testo all'interno del display LCD.

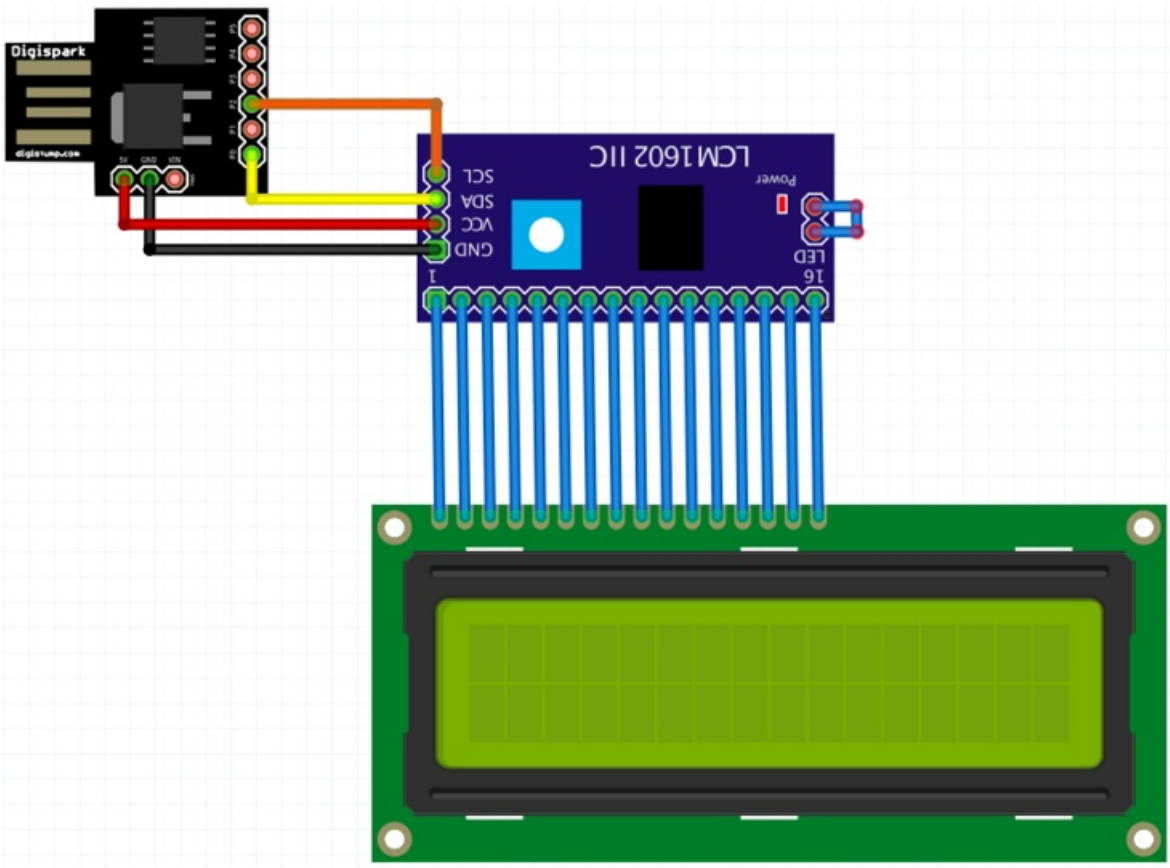
Materiale

- Un Digispark
- Un LCD
- Uno shift register

Schema

Descrizione del funzionamento

All’interno del circuito sono presenti due componenti: 1 display a cristalli liquidi (LCD) e 1 shift register. Lo shift register viene alimentato dal digispark ed è collegato ai relativi pin SCL(“P2”) e SDA(“P0”). Mentre il display LCD è collegato interamente allo shift register. Il display LCD stampa la scritta "Hello World" e la fa scorrere a destra e sinistra.



Codice

```
/**
 * Includo le librerie.
 */
#include <TinyWireM.h>
#include <LiquidCrystal_I2C.h>

/**
 * Istanzio un oggetto di tipo LCD all'indirizzo 0x27, con 16 caratteri e due righe.
 */
LiquidCrystal_I2C lcd(0x27,16,2);

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
```

```
void setup() {
  //Avvio la comunicazione I2C.
  TinyWireM.begin();
  //Preparo il display LCD.
  lcd.init();
  //Accendo la retroilluminazione.
  lcd.backlight();
  //Stampo all'interno del display LCD la scritta "Hello World!"
  lcd.print("Hello world!");
}

/**
 * Metodo che viene eseguito all'infinito.
 */
 void loop() {
   //Sposto il testo a sinistra di un carattere alla volta.
   for (int positionCounter = 0; positionCounter < 12; positionCounter++) {
     //Sposto a sinistra di un carattere.
     lcd.scrollDisplayLeft();
     //Aspetto 150ms.
     delay(150);
   }

   //Sposto il testo a destra di 30 posizioni (lunghezza stringa + lunghezza display).
   for (int positionCounter = 0; positionCounter < 30; positionCounter++) {
     //Sposto a destra di un carattere.
     lcd.scrollDisplayRight();
     //Aspetto 150ms.
     delay(150);
   }

   //Aspetto 1 secondo.
   delay(1000);

 }
```

Esempio PhotocellDelay

Descrizione

Circuito nel quale viene mostrato il valore rilevato da una fotocellula utilizzato come frequenza per il lampeggiamento del LED.

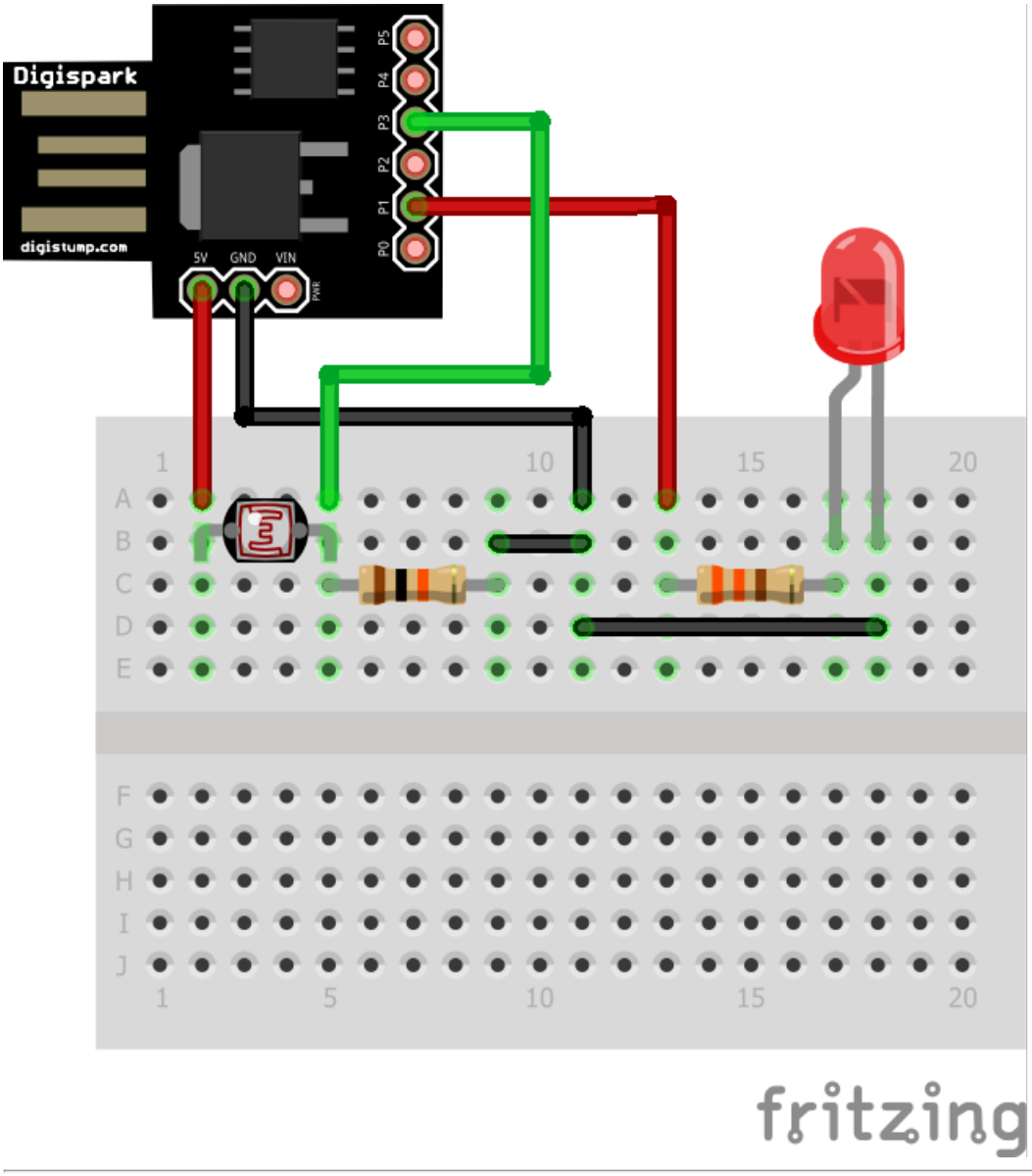
Materiale

- Un Digispark
- Un led
- Una fotocellula
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All'interno del circuito sono presenti quattro componenti: 1 fotocellula, 1 LED e 2 resistenze. La fotocellula è collegata in pull-down attraverso una resistenza da 10kΩ al pin "P3". Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω. Il valore letto dalla fotocellula viene assegnato ad un delay che gestisce la frequeza del lampeggiamento del LED.



Codice

```
/**
 * Includo le librerie.
 */
#include <photocell.h>
#include <led.h>

/**
 * Istanzio un oggetto di tipo Photocell.
 */
Photocell photocell(4);

/**
 * Istanzio un oggetto di tipo Led.
 */
Led led(1);

/**
 * Imposto un valore minimo di luminosità della resistenza.
 */
int minLux = 0;

/**
 * Imposto un valore massimo di luminosità della resistenza.
 */
int maxLux = 400;

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
void setup() {

}

/**
 * Metodo che viene eseguito all'infinito.
 */
void loop() {
  //Ricavo la luminosità.
  int lux = photocell.getLux();
  //Converto il valore di luminosità in un altra scala.
  int delayValue = map(lux, minLux, maxLux, 0, 1023);
  //Attendo il valore mappato.
  delay(delayValue);
  //Imposto la luminosità del led.
  led.toggle();
}
```

Esempio PhotocellThreshold

Descrizione

Circuito che in base al livello di luminosità della fotocellula accende il LED.

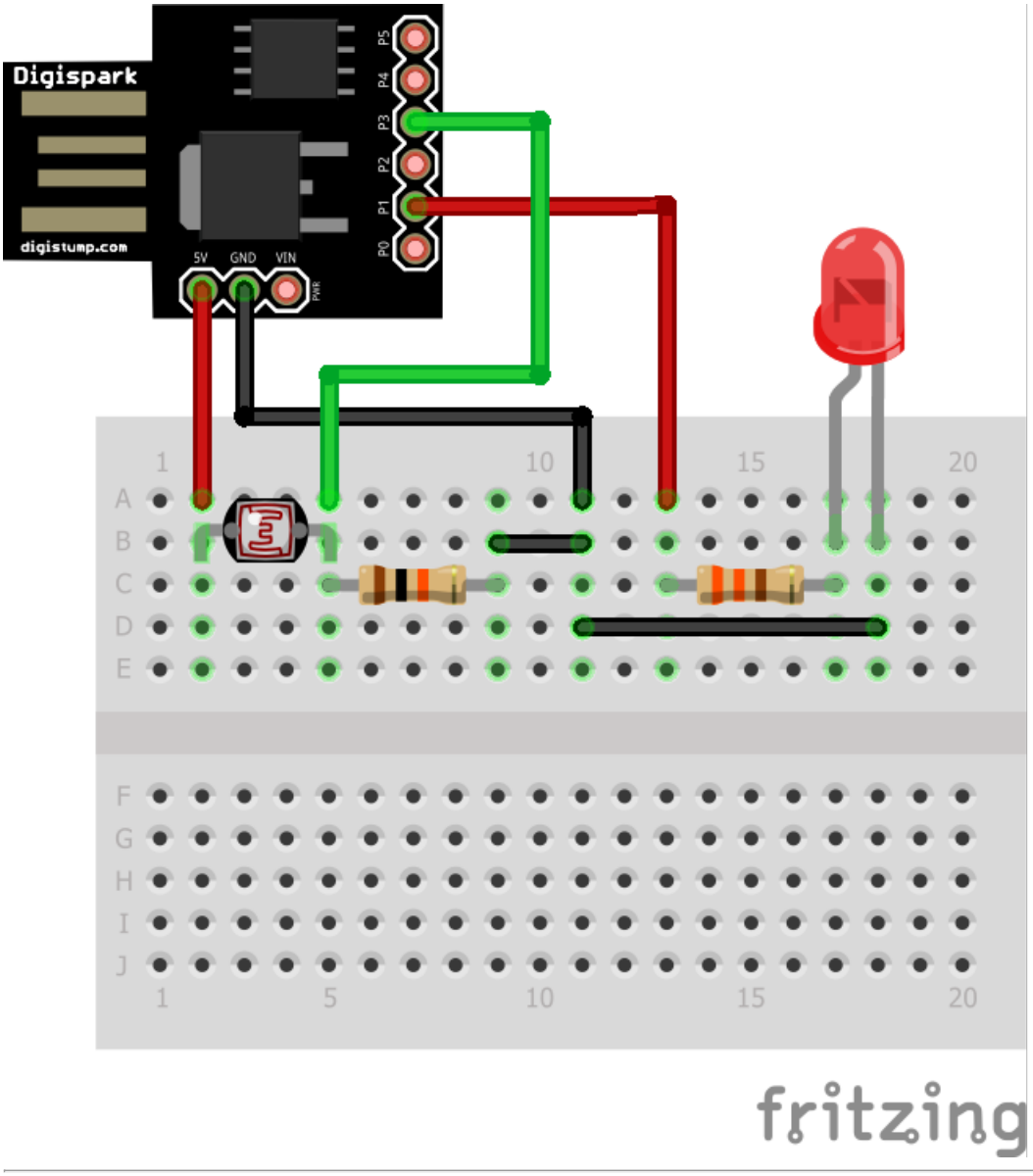
Materiale

- Un Digispark
- Un led
- Una fotocellula
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All'interno del circuito sono presenti quattro componenti: 1 fotocellula, 1 LED e 2 resistenze. La fotocellula è collegata in pull-down attraverso una resistenza da 10kΩ al pin "P3". Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω. Quando il valore letto dalla fotocellula supera i 100 lux il LED viene acceso, quando scende al di sotto della soglia il LED viene spento.



Codice

```
/**
 * Includo le librerie.
 */
#include <photocell.h>
#include <led.h>

/**
 * Istanzio un oggetto di tipo Photocell.
 */
Photocell photocell(4);

/**
 * Istanzio un oggetto di tipo Led.
 */
Led led(1);

/**
 * Imposto una soglia dopo il quale accendere il led.
 */
int soglia = 100;

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
void setup() {

}

/**
 * Metodo che viene eseguito all'infinito.
 */
void loop() {
  //Ricavo la luminosità.
  int lux = photocell.getLux();
  //Controllo che la luminosità sia maggiore o uguale alla soglia stabilita.
  if(lux >= soglia) {
    //Accendo il led.
    led.on();
  } else {
    //Spengo il led.
    led.off();
  }
}
```

Esempio PhotocellToLEDIntensity

Descrizione

Circuito che mostra come leggere il valore della luminosità ed assegnarlo al LED.

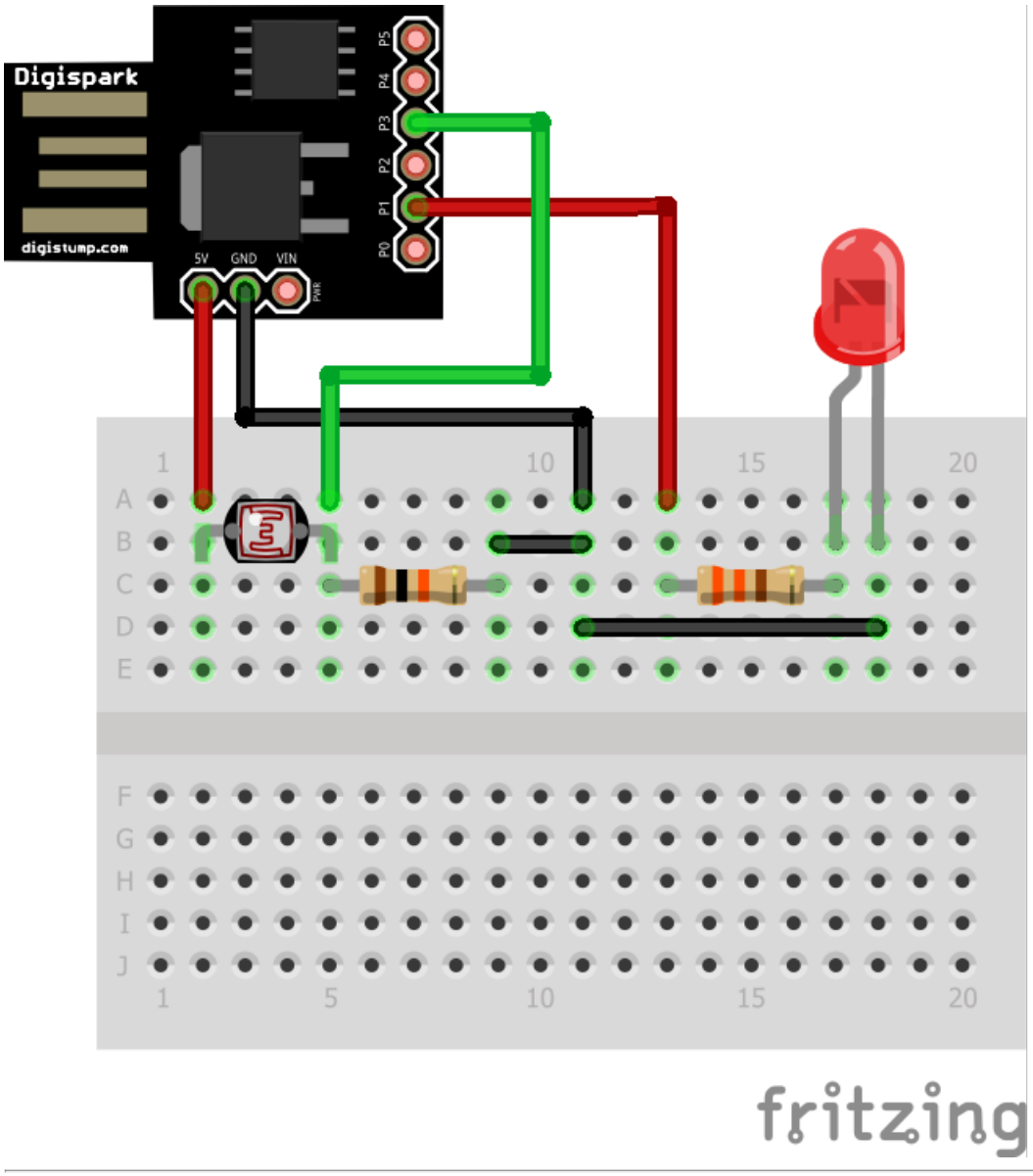
Materiale

- Un Digispark
- Un led
- Una fotocellula
- Una resistenza da 330Ω
- Una resistenza da 10KΩ

Schema

Descrizione del funzionamento

All'interno del circuito sono presenti quattro componenti: 1 fotocellula, 1 LED e 2 resistenze. La fotocellula è collegata in pull-down attraverso una resistenza da 10kΩ al pin "P3". Il LED è attaccato al pin "P1" del Digispark attraverso una resistenza da 330Ω. Il valore letto dalla fotocellula viene assegnato al LED.



Codice

```
/**
 * Includo le librerie.
 */
#include <photocell.h>
#include <led.h>

/**
 * Istanzio un oggetto di tipo Photocell.
 */
Photocell photocell(3);
/**
 * Istanzio un oggetto di tipo Led.
 */
Led led(4);
/**
 * Imposto un valore minimo di luminosità della resistenza.
 */
int minLux = 0;
/**
 * Imposto un valore massimo di luminosità della resistenza.
 */
int maxLux = 1023;

/**
 * Metodo di setup, viene eseguito una sola volta.
 */
void setup() {
}

/**
 * Metodo che viene eseguito all'infinito.
 */
void loop() {
  //Ricavo la luminosità.
  int lux = photocell.getLux();
  //Converto il valore di luminosità in un altra scala.
  int analogValue = map(lux, minLux, maxLux, 0, 255);
  //Imposto la luminosità del led.
  led.setAnalogState(analogValue);
}
```