



Università degli Studi di Milano Bicocca

**Scuola di Scienze**

**Dipartimento di Informatica, Sistemistica e Comunicazione**

**Corso di laurea in Informatica**

# **Progettazione di un sistema di acquisizione di immagini in ambiente controllato**

**Relatore:** *Gianluigi Ciocca*

**Co-relatore:** *Paolo Napoletano*

**Relazione della prova finale di:**

*Mirko Rima*

*Matricola 793435*

**Anno Accademico 2016-2017**

A mia madre,  
mio punto di riferimento  
e fonte d'ispirazione.

# Ringraziamenti

Al termine di questo mio percorso universitario ci terrei a fare alcuni ringraziamenti.

In primis volevo ringraziare, dal punto di vista didattico, il professor Bianco, il professor Napoletano e in particolare il professor Ciocca per avermi seguito nel mio percorso di tirocinio fino alla stesura della tesi, dandomi consigli e chiarimenti che mi hanno portato al raggiungimento dell'obiettivo finale.

Un ringraziamento speciale va alla mia fidanzata Antonella, che mi ha sopportato e supportato in tutti questi anni, con l'augurio che continui a farlo per molto altro tempo.

Un sentito grazie va ai miei amici, compagni di risate, avventura ma anche di importanti consigli.

Vorrei ringraziarvi uno per uno ma siete tanti e ognuno di voi ha e ha avuto una grande importanza per me.

Questo percorso universitario mi ha fatto conoscere splendide persone con cui ho stretto un legame di amicizia e con cui ho reso più bella questa esperienza sia nello studio sia nel divertimento, quindi grazie anche a voi.

Alla fine l'ultimo ringraziamento, ma per me il più importante.

Un grazie alla mia famiglia, mia sorella e in particolare mia madre, che in tutti questi anni ha creduto in me, non facendomi mai mancare nulla, sostenendo le mie passioni e i miei sogni, dandomi la forza di continuare sempre verso i miei obiettivi.

Quindi grazie a tutti, chi più chi meno ha contribuito a questo mio risultato e alla fine di un percorso.

# Sommario

Questa tesi si pone l'obiettivo di sviluppare un dispositivo in grado di acquisire delle immagini in un ambiente controllato.

Con acquisizione di immagini si intende riuscire ad ottenere delle immagini tramite un sistema di scatto automatico o manuale.

Con ambiente controllato si intende riuscire ad ottenere queste immagini nelle condizioni create e scelte.

I problemi che saranno trattati si basano sulle tempistiche, sui materiali e i metodi per raggiungere i risultati preposti.

Nella tesi verranno elencati gli obiettivi primari, i requisiti e i vincoli da rispettare. Verrà inoltre illustrato l'hardware e il software risultato necessario per la realizzazione del congegno.

Sarà effettuata un'attenta analisi sui passaggi seguiti nella costruzione, una dettagliata spiegazione sul funzionamento del dispositivo e alcune informazioni sugli aggiornamenti e miglioramenti disponibili su di esso.

Un importante obiettivo della tesi, oltre a spiegare il progetto realizzato, è permettere a chiunque legga questo documento di ricostruire e utilizzare questo prototipo.

# Indice

<b>Ringraziamenti</b>	II
<b>Sommario</b>	III
<b>Elenco delle figure</b>	VI
<b>Introduzione</b>	1
<b>1 Fase iniziale</b>	3
1.1 Divisione dei compiti . . . . .	3
1.1.1 Acquisizione immagini e gestione luci . . . . .	3
1.1.2 Preprocessing . . . . .	4
1.1.3 Ricostruzione di un modello 3D . . . . .	6
1.2 Fase di consultazione del team . . . . .	9
1.3 Struttura dispositivo . . . . .	10
1.4 Caso di studi . . . . .	11
<b>2 Hardware e specifiche utilizzate</b>	12
2.1 Jetson TK1 . . . . .	12
2.1.1 Caratteristiche . . . . .	12
2.1.2 Utilizzi . . . . .	14
2.1.3 Scelta . . . . .	15
2.1.4 Software e librerie istallate . . . . .	16
2.2 Arduino . . . . .	20
2.2.1 Caratteristiche . . . . .	20
2.2.2 Utilizzi . . . . .	23
2.2.3 Scelta . . . . .	23
2.2.4 Software e librerie istallate . . . . .	24
2.3 Pulsanti . . . . .	25
2.4 Sensore di prossimità . . . . .	26
2.5 Alimentatore . . . . .	31

2.6	Striscia led . . . . .	32
2.6.1	Scelta . . . . .	35
2.6.2	Caratteristiche led scelti . . . . .	37
2.7	Fotoresistore . . . . .	39
2.8	Spettrofotometro . . . . .	40
2.8.1	Caratteristiche . . . . .	40
2.8.2	Scelta . . . . .	40
2.9	Cablaggio, resistenze, breadboard e singolo led . . . . .	43
2.9.1	Cablaggio . . . . .	43
2.9.2	Resistenze . . . . .	44
2.9.3	Breadboard . . . . .	45
2.9.4	Singolo led . . . . .	46
2.10	Fotocamere . . . . .	47
<b>3</b>	<b>Funzionamento</b>	<b>48</b>
3.1	Diagramma di flusso . . . . .	49
3.2	Sezione A . . . . .	51
3.2.1	Costruzione e collegamenti . . . . .	52
3.2.2	Avvio e modalità . . . . .	53
3.2.3	Modalità automatica . . . . .	54
3.2.4	Modalità manuale . . . . .	62
3.2.5	Schema elettrico . . . . .	64
3.3	Sezione B . . . . .	66
3.3.1	Costruzione e collegamenti . . . . .	67
3.3.2	Rilevazione Luce . . . . .	70
3.3.3	Avvio e modalità . . . . .	75
3.3.4	Schema elettrico . . . . .	77
3.4	Malfunzionamenti collegamenti Arduino e Sincronizzazione . . . . .	79
<b>4</b>	<b>Fase finale e aggiornamenti</b>	<b>80</b>
4.1	Jetson TX1 . . . . .	80
4.2	Miglioramenti fotocamere . . . . .	83
4.3	Diagramma di flusso aggiornamenti . . . . .	86
	<b>Conclusioni</b>	<b>88</b>
	<b>Codice</b>	<b>89</b>
	<b>Riferimenti bibliografici</b>	<b>90</b>

# Elenco delle figure

1.1	Applicazione algoritmo demosaicing. . . . .	4
1.2	Applicazione white balancing. . . . .	5
1.3	Immagini non rettificate. . . . .	6
1.4	Immagini rettificate. . . . .	6
1.5	Depth map dell'immagine. . . . .	7
1.6	Point cloud di una scatola di medicinali. . . . .	8
1.7	Point cloud di tutta l'immagine. . . . .	8
1.8	Struttura IVAR utilizzata per il dispositivo. . . . .	10
1.9	Esempio di farmaci utilizzati. . . . .	11
2.1	Struttura Jetson TK1. . . . .	12
2.2	Vista della Jetson TK1. . . . .	14
2.3	Logo Nvidia CUDA. . . . .	15
2.4	Logo Arduino. . . . .	20
2.5	Esempio IDE Arduino. . . . .	21
2.6	Struttura Arduino Uno. . . . .	22
2.7	Logo Adafruit. . . . .	24
2.8	Visualizzazione dei pulsanti A e B sulla breadboard. . . . .	25
2.9	Pulsante "A". . . . .	25
2.10	Pulsante "B". . . . .	25
2.11	Simulazione del sensore di prossimità. . . . .	26
2.12	Vista anteriore del sensore SRF-05. . . . .	27
2.13	Vista posteriore del sensore SRF-05. . . . .	28
2.14	Rilevazione del sensore. . . . .	30
2.15	Alimentatore PU100-05. . . . .	31
2.16	Striscia led. . . . .	32
2.17	Striscia led RGB. . . . .	35
2.18	Singolo led RGB. . . . .	36
2.19	Schema singolo led RGB. . . . .	36
2.20	Adesivo striscia led. . . . .	36
2.21	Test accensione dei led montati sul dispositivo, senza collegare entrambe le estremità all'alimentatore. . . . .	37

2.22	Test accensione dei led alla massima potenza, senza collegare entrambe le estremità all'alimentatore. . . . .	37
2.23	Schema collegamenti striscia led RGB. . . . .	38
2.24	Fotoresistore utilizzato. . . . .	39
2.25	Fotoresistore. . . . .	39
2.26	Schema del fotoresistore. . . . .	39
2.27	Kit di calibrazione i1Basic Pro 2. . . . .	40
2.28	Spettrofotometro i1Basic Pro 2. . . . .	41
2.29	Schermata i1Profiler v 1.7. . . . .	42
2.30	Cavi Jumper. . . . .	43
2.31	Resistenza 10k $\Omega$ . . . . .	44
2.32	Breadboard. . . . .	45
2.33	Singolo led. . . . .	46
2.34	Rappresentazione singolo led. . . . .	46
2.35	Fotocamere c525 posizionate sul dispositivo. . . . .	47
2.36	Vista fotocamere c525. . . . .	47
3.1	Flow chart per fotocamere USB. . . . .	50
3.2	Flow chart sezione A. . . . .	51
3.3	Vista del dispositivo completo. . . . .	52
3.4	Pulsante A sulla breadboard. . . . .	54
3.5	Oggetto posizionato nel macchinario. . . . .	54
3.6	Esempio raggio di rilevazione dispositivo. . . . .	55
3.7	Rilevato dal sensore di prossimità. . . . .	55
3.8	Non rilevato dal sensore di prossimità. . . . .	55
3.9	Accensione del led per l'avvenuta rilevazione dell'oggetto. . . . .	56
3.10	Change detection, trasformazione in scala di grigi. . . . .	57
3.11	Errore scatto, mano all'interno dell'immagine. . . . .	57
3.12	Applicazione white balancing scatto SX. . . . .	58
3.13	Applicazione white balancing scatto DX. . . . .	58
3.14	Console durante la modalità automatica. . . . .	60
3.15	Console al momento dello stop. . . . .	61
3.16	Pulsante B sulla breadboard. . . . .	62
3.17	Console durante la modalità manuale. . . . .	63
3.18	Rappresentazione del circuito della sezione A. . . . .	64
3.19	Schema elettrico del circuito della sezione A. . . . .	65
3.20	Circuito stampato per la sezione A. . . . .	65
3.21	Flow chart sezione B. . . . .	66
3.22	Disposizione dei led sull'IVAR. . . . .	67
3.23	Schema collegamento striscia led - Arduino. . . . .	68
3.24	Collegamento sensore luminosità - Arduino - striscia led. . . . .	69



3.25	Schermata iProfiler durante rilevazione luce ambiente. . . . .	71
3.26	Dispositivo con luce naturale sufficiente. . . . .	76
3.27	Dispositivo con i led attivi. . . . .	76
3.28	Rappresentazione del circuito della sezione B. . . . .	77
3.29	Schema elettrico del circuito della sezione B. . . . .	78
3.30	Circuito stampato per la sezione B. . . . .	78
4.1	Jetson TX1. . . . .	80
4.2	Jetson TX1 - Jetson TK1. . . . .	82
4.3	Modulo camera Jetson TX1. . . . .	83
4.4	Camera OV5647 con lente intercambiabile . . . . .	84
4.5	Camera OV5647 con bus CSI. . . . .	84
4.6	Parte anteriore dell'Auvideo J20. . . . .	84
4.7	Parte posteriore dell'Auvideo J20. . . . .	84
4.8	Prototipo Jetson TX1 con 6 camere. . . . .	85
4.9	Logo Auvideo. . . . .	86
4.10	Flow chart completo prototipo finale. . . . .	87

# Introduzione

Questo progetto di tesi nasce dall'esperienza di tirocinio maturata durante il terzo anno accademico presso il laboratorio IoT dell'università degli Studi di Milano-Bicocca.

All'inizio del percorso, insieme a due colleghi, in accordo con i nostri Tutor è stata presa la decisione di realizzare un dispositivo, partendo da un'idea e arrivando alla sua realizzazione materiale.

L'idea principale del progetto era creare un sistema di acquisizione di immagini in un ambiente controllato, che poi le elaborasse tramite preprocessing [1] e infine ne creasse il modello 3D [2].

Prima di iniziare questo percorso sono stati stabiliti alcuni elementi importanti per poter procedere, che saranno elencati qui in seguito.

## Obiettivi

Sono stati delineati tre obiettivi principali:

1. Creare un dispositivo che gestisca l'illuminazione al suo interno e ottenere da esso delle immagini in modo automatico e manuale;
2. Svolgere un preprocessing delle immagini acquisite in modo da poter modificare e ottenere immagini ad hoc;
3. Ricostruire un modello 3D dalle immagini ottenute.

Personalmente mi sono occupato del primo obiettivo, mentre i miei due colleghi dei restanti punti.

---

## Requisiti

Per riuscire a soddisfare gli obiettivi preposti è stato necessario pensare ad alcune soluzioni.

Ci è stata lasciata ampia libertà di scelta per portare a termine il progetto nel miglior modo possibile.

Si è quindi pensato che gli elementi principali che avrebbero potuto realizzare la parte di acquisizione delle immagini fossero:

- una struttura che contenesse il tutto;
- un microcontrollore [3];
- un sensore di prossimità [4];
- almeno due fotocamere USB.

Mentre gli elementi principali per la parte di gestione delle luci fossero:

- una striscia led [5];
- un microcontrollore;
- un sensore di luminosità [6];
- un alimentatore [7].

## Vincoli

Per portare al termine questi obiettivi sono stati fissati alcuni vincoli principali:

- avere più camere per poter avere una ricostruzione 3D attendibile;
- creare un dispositivo potenzialmente estendibile con altre camere;
- rendere il processo di scatto, elaborazione e ricostruzione 3D relativamente veloce (impiegare al massimo qualche minuto);
- rendere aggiornabile il dispositivo, rendendolo compatibile con la Jetson TX1 [8];
- mantenere una luminosità nel dispositivo intorno ai 400 - 500 lx [9].

# Capitolo 1

## Fase iniziale

### 1.1 Divisione dei compiti

#### 1.1.1 Acquisizione immagini e gestione luci

In questa prima fase ho creato un dispositivo che avrebbe rilevato alcuni oggetti inseriti al suo interno, ottenuto le immagini di questi ultimi e permesso ai miei due colleghi di ricevere il materiale, nelle condizioni migliori, per poter effettuare le fasi successive al progetto.

Per avere condizioni ideali è risultato necessario controllare la luminosità all'interno del congegno e perciò ho gestito la luce erogata tramite strisce led RGB programmabili [10].

Questi punti sono spiegati esaurientemente nei capitoli successivi.

Le prossime due fasi (Preprocessing e Ricostruzione modello 3D) sono state trattate dai miei due colleghi, perciò mi limiterò ad introdurle dandone una breve spiegazione.

### 1.1.2 Preprocessing

Il preprocessing è la fase di elaborazione dell'immagine o del segnale con la finalità di migliorare il futuro lavoro di estrazione delle caratteristiche.

Sono state trattate queste tre fasi principali:

1. **Demosaicing** [11]: Un algoritmo di demosaicing permette di ricostruire la rappresentazione a colori di un'immagine partendo dai dati grezzi ottenuti dal sensore di una fotocamera digitale che utilizza un color filter array (CFA).

I CFA sono utilizzati dalla maggior parte delle moderne fotocamere digitali, pertanto gli algoritmi di demosaicing sono parte fondamentale del processo di elaborazione dell'immagine effettuato da una fotocamera al fine di permettere la visualizzazione dell'immagine.

Inoltre molte fotocamere permettono anche di salvare le immagini in formato RAW [12] con la possibilità di scegliere la tecnica di demosaicing che si vuole utilizzare al posto dell'algoritmo proprietario della fotocamera.

L'obiettivo di un algoritmo di demosaicizzazione è la ricostruzione di un'immagine a colori (per esempio nello spazio RGB, composto da rosso, verde e blu) a partire dai dati acquisiti da un sensore con CFA.

In particolare, un buon algoritmo deve garantire:

- fedeltà all'immagine originale;
- bassa complessità computazionale, in modo da permetterne l'esecuzione in tempi ragionevoli.



Figura 1.1: Applicazione algoritmo demosaicing.

Nella figura 1.1 si può notare nella parte sinistra l'immagine CFA, mentre a destra l'immagine dopo che il demosaicing è stato applicato.

2. **Color constancy** [13]: Si bilancia il colore per portarlo alla condizione naturale, dove il bianco è il bianco e il nero è il nero.

Si cerca di far percepire l'oggetto allo stesso modo sotto ogni tipo di luce.

Della color constancy si è trattata la parte di White Balancing (bilanciamento del bianco) [14], regolando l'intensità dei vari colori (solitamente rosso, verde e blu).

L'obiettivo principale di questa regolazione è ottenere la rappresentazione corretta di un colore, solitamente uno dei colori neutri.

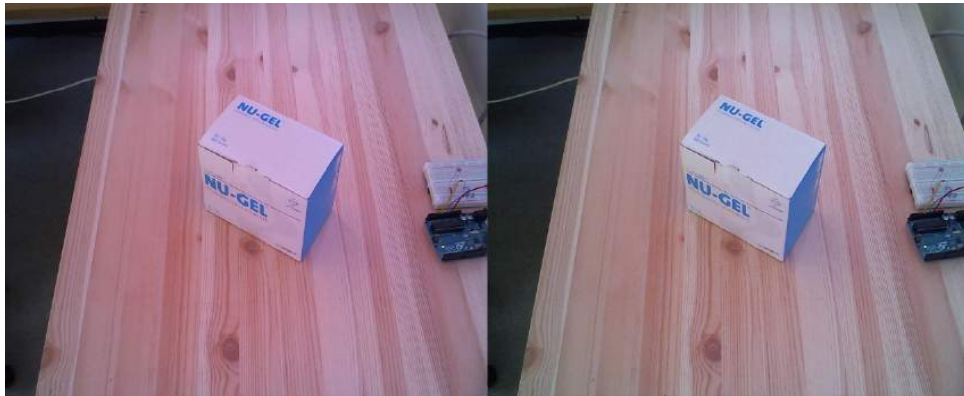


Figura 1.2: Applicazione white balancing.

Nella figura 1.2 si può notare a sinistra l'immagine originale, mentre a destra la stessa dopo il white balancing.

3. **Color mapping** [15]: La mappatura dei colori è una funzione che trasforma i colori di un'immagine (sorgente) nei colori di un'altra immagine (bersaglio). Bisogna ricordare che l'obiettivo della camera non è perfetto, perciò utilizzando una color check, si sapranno i valori reali.

Per il raggiungimento di un buon risultato finale si deve trovare una matrice da moltiplicare all'immagine sorgente per far sì che i colori siano il più simile possibili a quelli reali.

Poi si guarda quale tra le matrici minimizza l'errore e si prende la migliore.

### 1.1.3 Ricostruzione di un modello 3D

Una ricostruzione di un modello 3D è composta da cinque fasi principali:

1. **Calibrazione camere** [16]: In questa fase si crea un dataset di immagini, contenenti un pattern noto (nel nostro caso una scacchiera), nelle quali sono individuati i corner.  
Ciò viene fatto sia per l'immagine destra sia per quella sinistra.  
Da questi punti si calcolano delle matrici di rotazione/traslazione grazie alle quali è possibile ricavare la posizione dei punti in una camera rispetto all'altra.
2. **Rettifica immagini** [16]: Sulle immagini in input (figura 1.3) vengono applicate delle trasformazioni, sfruttando le matrici precedentemente calcolate, così che le immagini rettificate (figura 1.4) abbiano una particolare caratteristica: ogni punto del mondo reale è sulla stessa riga, stessa altezza, su entrambe le immagini.

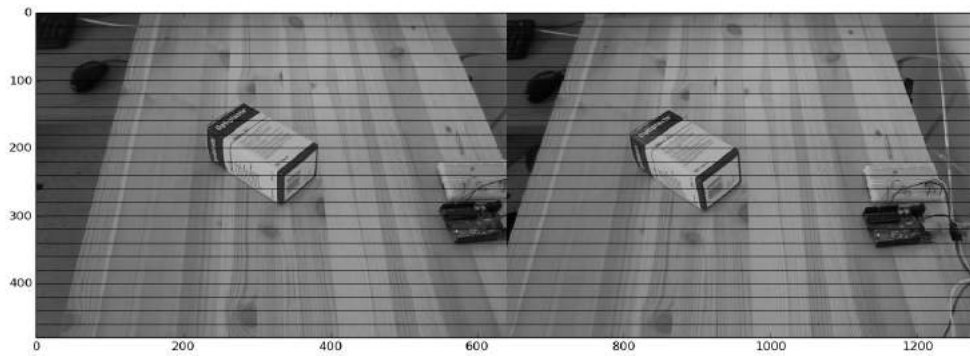


Figura 1.3: Immagini non rettificate.

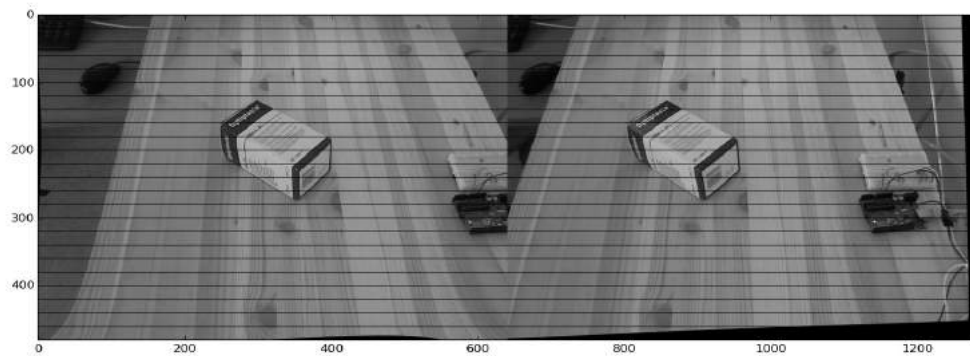


Figura 1.4: Immagini rettificate.

3. **Stereo matching** [17]: Con le immagini rettificate, come in figura 1.4, è possibile per ogni riga cercare un punto, dell'immagine di riferimento, nell'altra immagine.

La distanza, chiamata disparità, indica quanto è profondo quel punto rispetto le camere, grazie all'informazione di quanto esso si è spostato tra le due prospettive delle camere.

Più si è spostato più è vicino.

L'output è la depth map, mappa di profondità, disparità come si può vedere nella figura 1.5.

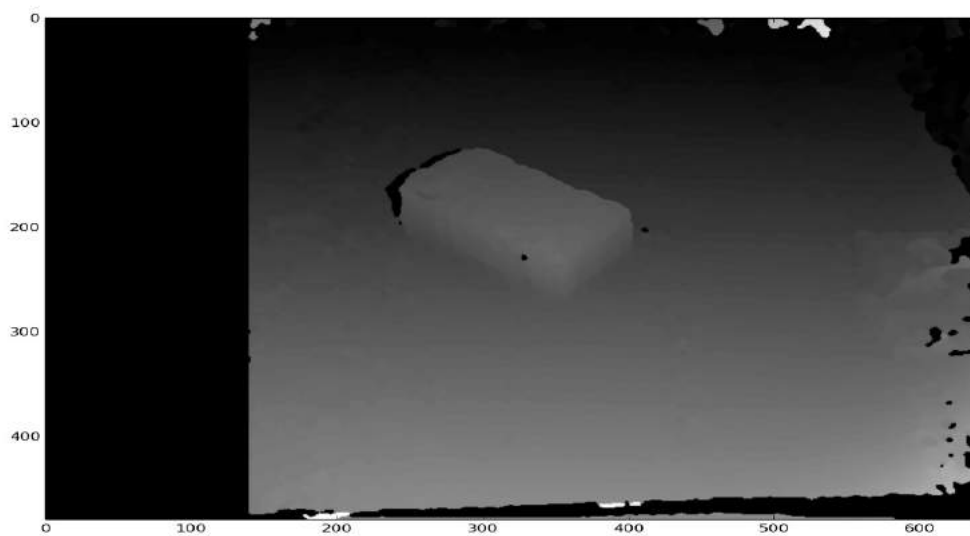


Figura 1.5: Depth map dell'immagine.



4. **Point cloud** [18]: Dalla mappa di profondità è possibile ricavare la tripla di coordinate per ogni punto dell'immagine, e quindi proiettarla nello spazio, creando appunto una nuvola di punti.

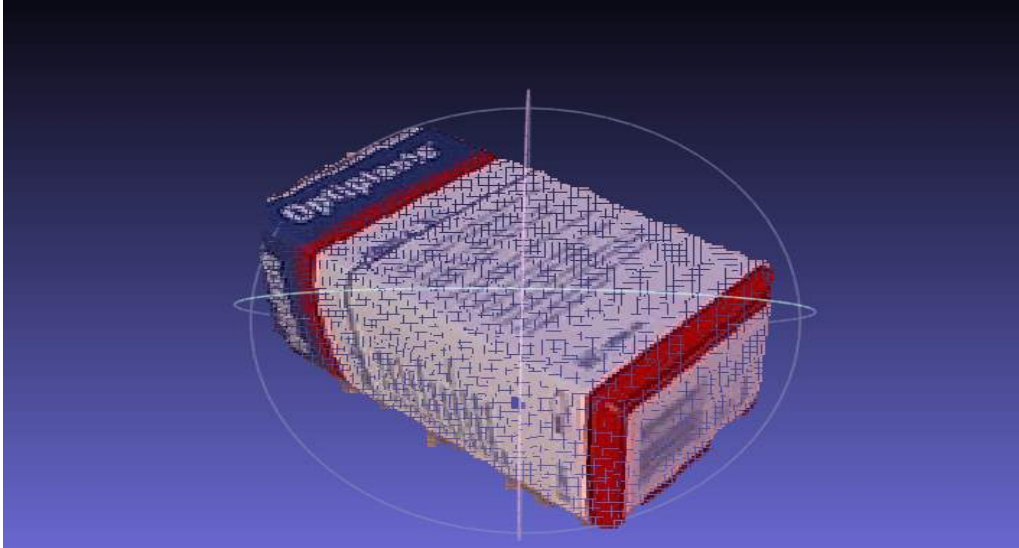


Figura 1.6: Point cloud di una scatola di medicinali.

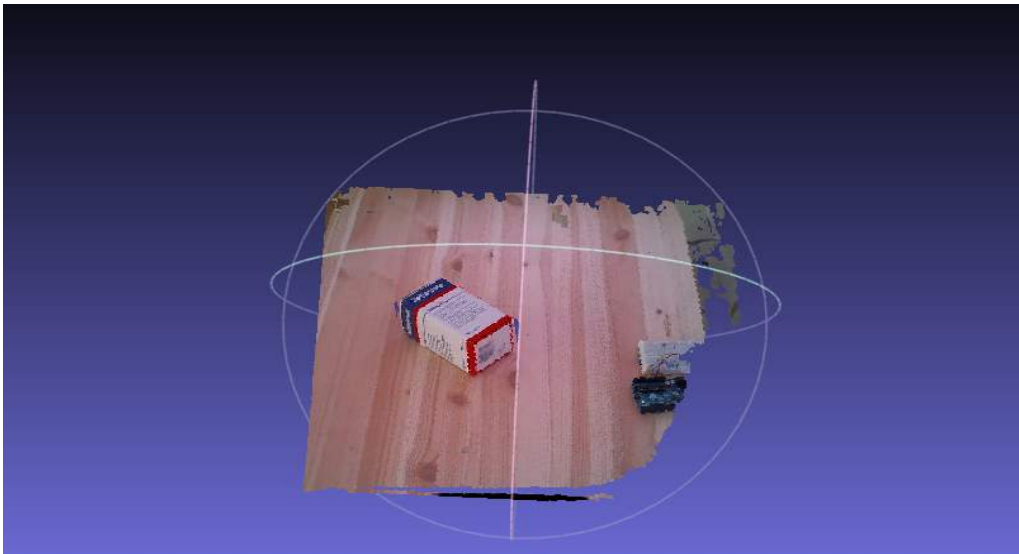


Figura 1.7: Point cloud di tutta l'immagine.

5. **Fit plane/Mesh point cloud e Stima volume scatola** [19]: Serve per chiudere quei buchi nella nuvola di punti, causati da regioni uniformi nell'immagine.

## 1.2 Fase di consultazione del team

Dopo avere ricevuto le specifiche dai nostri Tutor, è iniziata una fase di consultazione.

Ognuno di noi aveva delle mansioni da svolgere nel progetto, ma come prima parte si è rivelato necessario accordarci sugli elementi indispensabili per poter realizzare il tutto.

È stato fondamentale collaborare per creare il dispositivo nel modo migliore e senza sprecare risorse.

La comunicazione con varie società e le informazioni dettagliate ottenute, sono state un elemento rilevante per avere delle giuste alternative per poter scegliere consapevolmente.

Dopo un periodo di ricerche e consultazioni, il team è giunto a queste conclusioni su alcune scelte dei requisiti:

- come microcontrollori sarebbero stati utilizzati due Arduino Uno [20];
- come “computer” per elaborare il tutto la Jetson TK1 [21], poiché come sarà spiegato nel capitolo 2.1, questa scheda è una delle risorse più utilizzate da chi si occupa di Image e Vision Computing;
- per la parte dell’illuminazione avrei dovuto usare delle strisce led RGB programmabili, compatibili con Arduino;
- avrei utilizzato due fotocamere USB, lasciando però la possibilità in futuro di poter aggiungere un maggiore numero di camere, anche di diverso tipo come quelle con interfaccia a bus CSI (Camera Serial Interface) [22].

In seguito con lo svolgimento del progetto ho iniziato a pensare a ogni singolo componente, quali resistenze [23], sensori, ecc. che saranno trattati esaurientemente nel capitolo 2.

### 1.3 Struttura dispositivo

Prima di ogni cosa è stato necessario pensare allo “scheletro” del dispositivo, cioè al materiale e come sarebbe stato costruito.

Il consiglio datoci dai nostri Tutor era quello di comprare una struttura che potesse essere malleabile per le nostre esigenze.

Abbiamo quindi optato per l'IVAR dell'Ikea, mostrata nella figura 1.8, una struttura di legno adattabile a seconda delle necessità e sulla quale avremmo potuto iniziare a montare le parti del dispositivo.

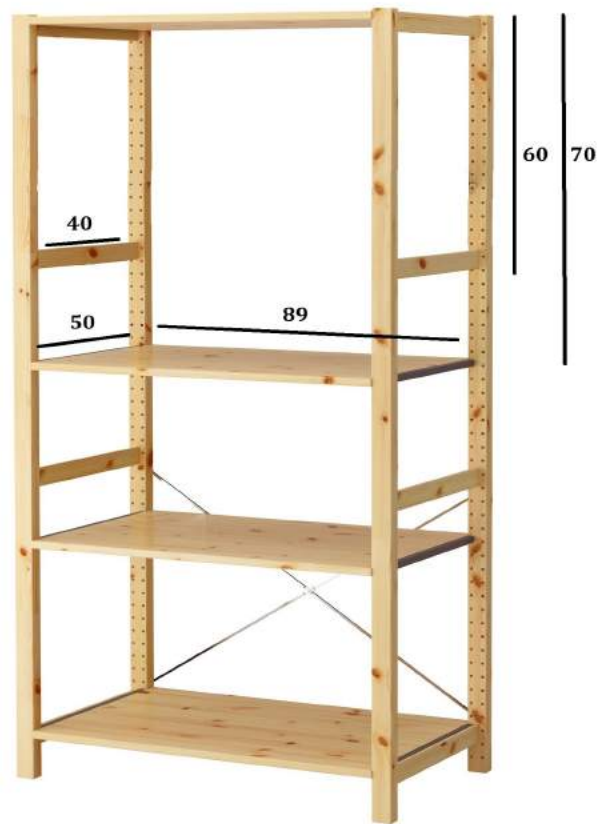


Figura 1.8: Struttura IVAR utilizzata per il dispositivo.

Abbiamo scelto questo genere di struttura aperta, perché ci permetteva una ampia libertà di movimento e inoltre esisteva la possibilità sia di ricevere luce naturale dall'ambiente circostante, sia di chiudere il dispositivo con dei teli per lavorare con la sola luce artificiale.

Queste circostanze permettevano di studiare tutte le varie possibilità, in modo tale da ricreare le condizioni di luce da noi concordate.

## 1.4 Caso di studi

In seguito abbiamo pensato a delle soluzioni sulle quali concentrarci per la ricerca di questo prototipo.

Concordando con i Tutor abbiamo deciso di utilizzare come caso di studi alcune scatole di medicinali, di ogni forma e colore.

Questo ha delineato la mia linea di pensiero e anche se il dispositivo risulta funzionante con moltissimi altri oggetti, ho pensato a soluzioni ideali per il nostro caso.



Figura 1.9: Esempio di farmaci utilizzati.

# Capitolo 2

## Hardware e specifiche utilizzate

### 2.1 Jetson TK1

#### 2.1.1 Caratteristiche

La Jetson TK1 [21], sviluppata dall'NVIDIA, è sostanzialmente un piccolo computer-on-a-board, una piattaforma di sviluppo basata sul primo supercomputer mobile al mondo per sistemi embedded.

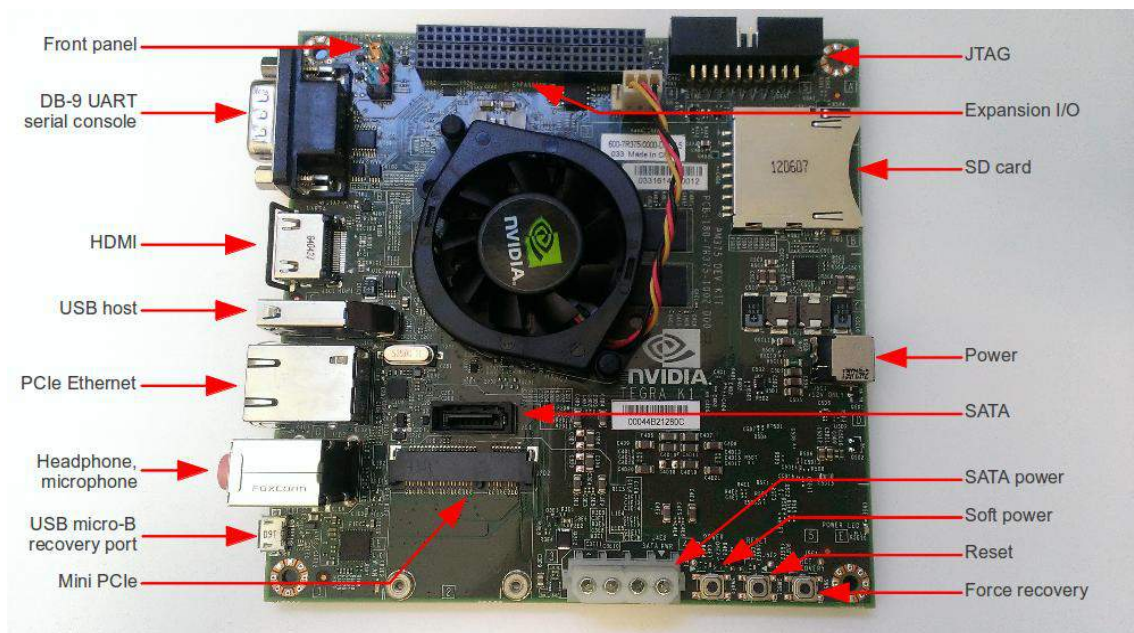


Figura 2.1: Struttura Jetson TK1.

Al centro del Jetson TK1 Developer Kit [24], in figura 2.1, vi è il processore mobile Tegra K1 [25], dotato di grafica ad alta efficienza energetica, anche grazie ai 192 core completamente programmabili che assicurano le migliori prestazioni di calcolo.

La scheda integra:

- 2GB di memoria RAM;
- 16GB di memoria interna eMMC;
- connettori di input/output;
- connettori USB 3.0;
- HDMI 1.4;
- Gigabit Ethernet;
- audio;
- SATA e uno slot per schede SD;
- una porta di espansione per UART.

Viene fornita completa di un pacchetto di supporto e alcuni software, tra cui OpenGL 4.4 [26], CUDA [27] e il toolkit VisionWorks [28].

Sono incluse anche una completa suite di strumenti di sviluppo e profili per le fotocamere e altre periferiche.

La Jetson TK1 è in grado di offrire prestazioni senza eguali pari a 326 gigaflop, quasi il triplo rispetto a qualsiasi altra piattaforma embedded simile.

### 2.1.2 Utilizzi

La piattaforma NVIDIA Jetson TK1 fornisce agli sviluppatori gli strumenti per creare sistemi e applicazioni che possono consentire ai robot di spostarsi autonomamente, ai medici di eseguire ecografie in mobilità, ai droni di evitare oggetti in movimento e alle auto di rilevare i pedoni.

La piattaforma Jetson TK1 supporta il toolkit NVIDIA VisionWorks, che offre una completa serie di algoritmi per la computer vision [29] e l'elaborazione delle immagini per creare applicazioni in modo rapido per settori come la robotica, la realtà aumentata, la fotografia computazionale, le interfacce uomo-computer e i sistemi avanzati di assistenza alla guida.



Figura 2.2: Vista della Jetson TK1.

### 2.1.3 Scelta

Ho scelto di utilizzare il Developer Kit Jetson TK1 (figura 2.1) per la veloce elaborazione di calcolo di cui esso dispone poiché il tempo per le immagini che dovevamo analizzare doveva essere il minore possibile per permettere al dispositivo di rendere al massimo delle prestazioni.

Infatti il modulo mette a disposizione un completo toolkit C/C++ basato su architettura NVIDIA CUDA, la più diffusa piattaforma di calcolo parallelo e modello di programmazione, che lo rende molto più semplice da programmare.

Ciò fa sì che il Jetson TK1 Developer Kit offra pieno supporto tramite CUDA 8 e porti alla piattaforma ARM [30] le librerie accelerate di NVIDIA per l'algebra lineare, le matrici sparse, nonché per l'immagine e il video processing [31].

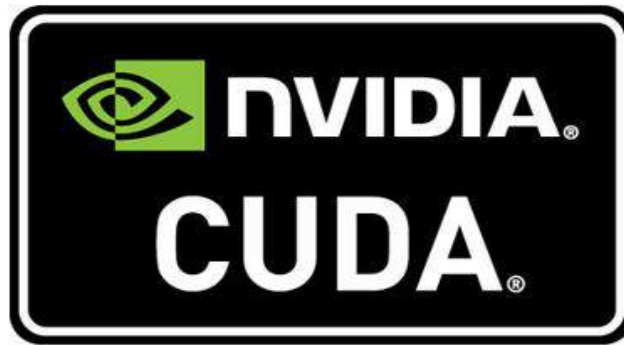


Figura 2.3: Logo Nvidia CUDA.



### 2.1.4 Software e librerie istallate

All'inizio ho istallato il pacchetto JetPack 3.1 [32] dell'NVIDIA scaricabile presso il sito ufficiale:

<https://developer.nvidia.com/embedded/jetpack>

NVIDIA JetPack è la soluzione più completa per la creazione di applicazioni AI (Artificial Intelligence) [33].

Ho scelto di istallare JetPack per esplorare il Jetson Developer Kit con estrema velocità e installare strumenti di sviluppo, librerie, API e la documentazione inerente in modo semplice ed efficace, in modo che l'ambiente di sviluppo fosse pronto il più rapidamente possibile.

Il pacchetto JetPack 3.1 comprende:

TensorRT 2.1	TensorRT è un runtime di inferenza di deep learning ad alte prestazioni per la classificazione delle immagini, la segmentazione e le reti neurali di rilevamento degli oggetti.
cuDNN 6.0	La libreria CUDA Deep Neural Network fornisce alte prestazioni per tutti i framework di deep learning. Include il supporto per convoluzioni.
VisionWorks 1.6	VisionWorks è un pacchetto di sviluppo software per Computer Vision (CV) e l'elaborazione delle immagini.
CUDA 8.0	CUDA Toolkit fornisce un ambiente di sviluppo completo per gli sviluppatori C e C++ sviluppando applicazioni GPU accelerate e fornisce gli strumenti per il debug e l'ottimizzazione delle prestazioni delle applicazioni.
Multimedia API	Il pacchetto Jetson Multimedia API fornisce API a basso livello per lo sviluppo di applicazioni flessibili.
L4T	L4T (o Linux For Tegra) è un pacchetto di driver NVIDIA che supporta lo sviluppo delle piattaforme Embedded Jetson. Per la Jetson TK1 si è arrivati alla versione 21.5 e si utilizza un sistema operativo a 32-bit Ubuntu 14.04.

Development Tools	<p>Tegra System Profiler 3.8 è un profiler di campionamento PC di CPU che fornisce una visione interattiva dei dati di profilazione acquisiti, contribuendo a migliorare le prestazioni dell'applicazione complessiva.</p> <p>Tegra Graphics Debugger 2.4 è uno strumento di console che consente agli sviluppatori di eseguire il debug consentendo di sfruttare al meglio la piattaforma Jetson Embedded.</p>
-------------------	---

Dato che la Jetson TK1, come espresso precedentemente, funziona con un sistema operativo Linux ARM (Linux For Tegra) tutti i software e le librerie istallate sono di conseguenza la versione per Ubuntu 14.04.

In questo progetto ho lavorato con Python.

Questo linguaggio dovrebbe essere incluso nell'installazione di Ubuntu 14.04, se così non fosse digitare da terminale:

```
sudo apt-get install python
```

e aspettare l'installazione.

Sono state istallate le seguenti librerie per Python:

### Elenco Librerie

- **OpenCV** [34]: OpenCV (Open Source Computer Vision Library) è una libreria software multiplatforma nell'ambito della visione artificiale in tempo reale.  
È un software libero, progettato per l'efficienza computazionale e con un forte focus sulle applicazioni in tempo reale.  
La libreria può trarre vantaggio dall'elaborazione multi-core.  
Il linguaggio di programmazione principalmente utilizzato per sviluppare con questa libreria è il C++, ma è possibile interfacciarsi anche attraverso il C, Python e Java.  
Noi per comodità abbiamo deciso di utilizzare OpenCV 3.0, utilizzando la libreria tramite comandi Python.

La libreria è scaricabile dal sito ufficiale:

<http://opencv.org/releases.html>

- **NumPy** [35]: NumPy è un'estensione open source del linguaggio di programmazione Python, che aggiunge supporto per vettori e matrici multidimensionali e di grandi dimensioni con funzioni matematiche di alto livello con cui operare.

La libreria è scaricabile dal sito ufficiale:

<https://www.scipy.org/install.html>

- **SciPy** [36]: libreria open source di algoritmi e strumenti matematici per il linguaggio di programmazione Python.  
Contiene moduli per l'ottimizzazione, per l'algebra lineare, l'integrazione, funzioni speciali, elaborazione di segnali ed immagini, altri strumenti comuni nelle scienze e nell'ingegneria.  
Trova utilizzo in quei programmatori che usano anche MATLAB.

La libreria è scaricabile dal sito ufficiale:

<https://www.scipy.org/install.html>

- **Matplotlib** [37]: libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy.  
Fornisce API orientate agli oggetti che permettono di inserire grafici all'interno di applicativi usando toolkit GUI generici.  
C'è anche una interfaccia "pylab" procedurale basata su una macchina degli stati progettata per assomigliare a quella di MATLAB.

La libreria è scaricabile dal sito ufficiale:

<https://matplotlib.org/users/installing.html>

- **CSV** [38]: se si desidera importare o esportare fogli di calcolo e database per l'utilizzo nell'interprete Python, è necessario contare sul modulo CSV o sul formato dei valori separati da virgole.

La libreria è scaricabile dal sito:

[www.object-craft.com.au/projects/csv/install.html](http://www.object-craft.com.au/projects/csv/install.html)

- **time** [39]: questo modulo fornisce varie funzioni correlate al tempo. Per le funzionalità correlate, vedere anche i moduli di data e di calendario. La libreria è già integrata nell'installazione di Python.
- **pySerial** [40]: questo modulo incapsula l'accesso per la porta seriale. Fornisce backend per Python in esecuzione su Windows, OSX, Linux. Il modulo denominato "serial" seleziona automaticamente il backend appropriato. Abbiamo utilizzato questa libreria poiché dovevamo comunicare in via seriale tra la Jetson e Arduino, perciò c'era la necessità di scambiare dati.

La libreria è scaricabile dal sito ufficiale:

<http://pyserial.readthedocs.io/en/latest/pyserial.html>

## 2.2 Arduino

### 2.2.1 Caratteristiche

Arduino [41] è una piattaforma hardware composta da una serie di schede elettroniche dotate di un microcontrollore.

Questo componente è stato sviluppato da alcuni membri dell'Interaction Design Institute di Ivrea come strumento per la prototipazione rapida e per scopi didattici e professionali.



Figura 2.4: Logo Arduino.

L'ambiente di sviluppo integrato (IDE) [42] di Arduino, rappresentanto con un esempio in figura 2.5, è un'applicazione multiplatforma scritta in Java.

I programmi sono scritti in un linguaggio di programmazione semplice e intuitivo, chiamato Wiring, derivato dal C e dal C++, liberamente scaricabile e modificabile.

I programmi in Arduino vengono chiamati sketch.

Per permettere la stesura del codice sorgente, l'IDE include un editore di testo dotato di alcune particolarità, come il syntax highlighting, il controllo delle parentesi e l'indentazione automatica.

L'editor è inoltre in grado di compilare e caricare il programma funzionante ed eseguibile sulla scheda con un solo click.

In genere non vi è bisogno di creare dei Makefile o far girare programmi dalla riga di comando.

Per comodità e per vedere i risultati di uno sketch è attivabile dall'IDE una finestra seriale di monitoring, sulla quale far comparire l'output di istruzioni `Serial.print(parametro)` incorporate nello sketch stesso.

La piattaforma fisica si basa su un circuito stampato che integra un microcontrollore con dei pin connessi alle porte I/O, un regolatore di tensione e, quando necessario, un'interfaccia USB che permette la comunicazione con il computer utilizzato per programmare.

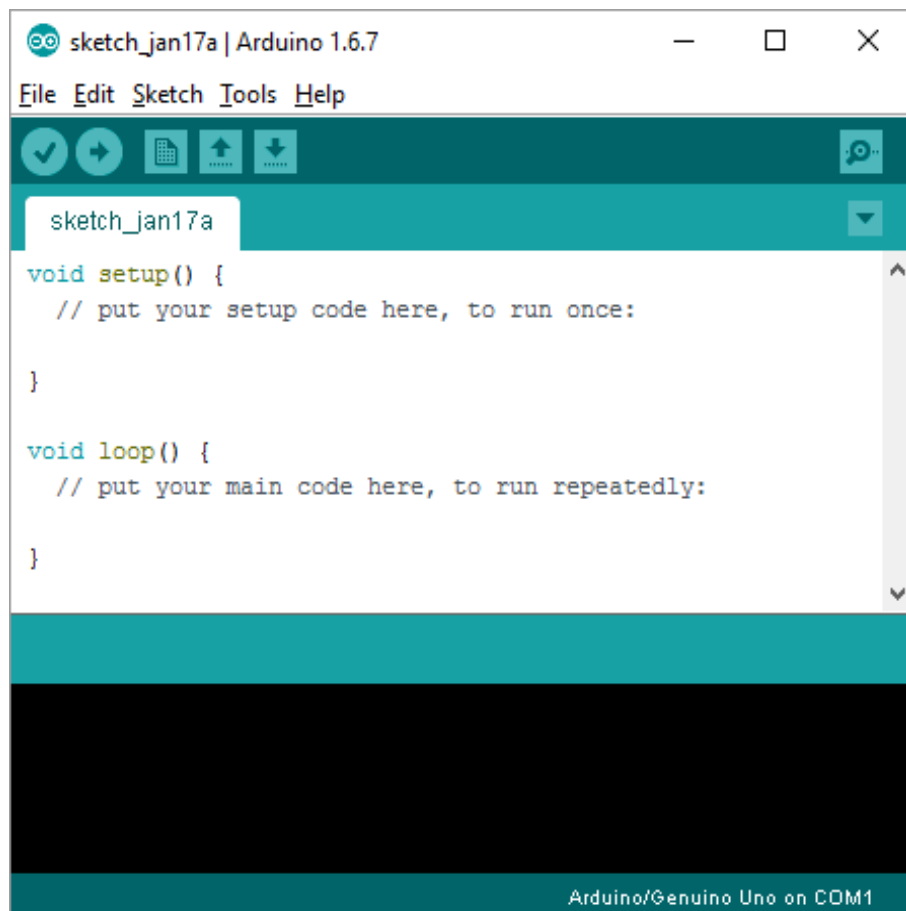


Figura 2.5: Esempio IDE Arduino.

Esistono vari modelli di Arduino, tuttavia ho optato per Arduino Uno [20] per le motivazioni che sono espresse in seguito nel capitolo 2.2.3.

Caratteristiche di Arduino Uno:

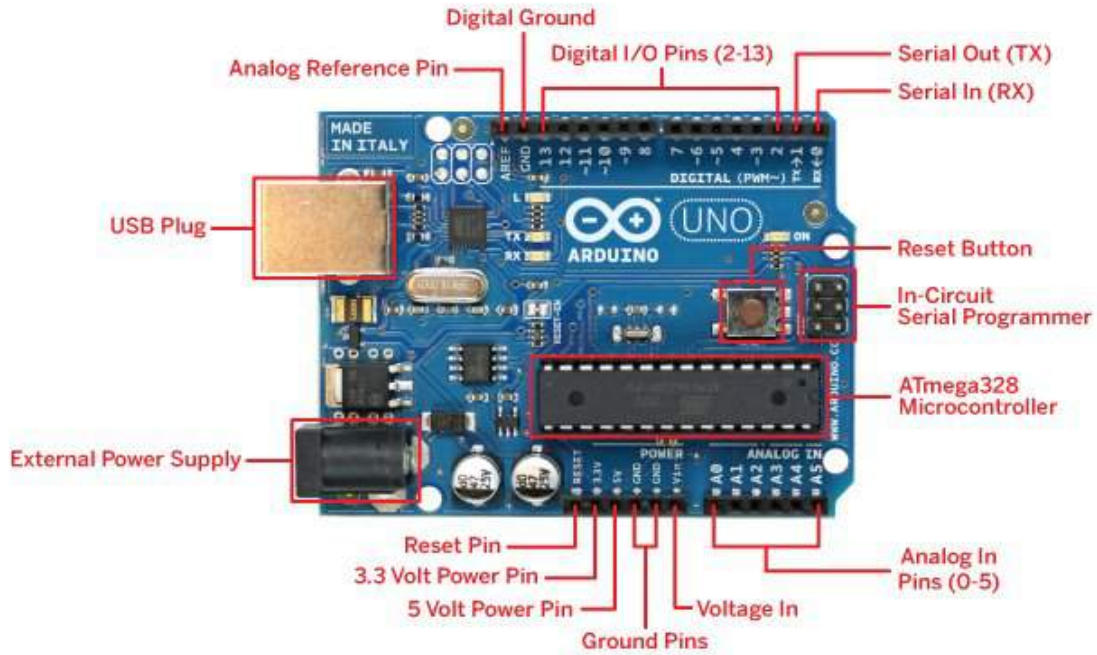


Figura 2.6: Struttura Arduino Uno.

Arduino Uno, in figura 2.6, è una scheda basata sul microcontrollore ATmega328. Ha 14 pins digitali programmabili in ingresso e in uscita; 6 di questi pins possono essere usati come uscita PWM [43].

Ha 6 ingressi analogici, un oscillatore a 16 Mhz, una connessione USB, un connettore di potenza, un ICSP e un pulsante di reset.

Arduino contiene tutto ciò che serve di supporto al microcontrollore.

Per alimentarlo basta connetterlo al computer tramite un cavo USB o tramite una batteria o un trasformatore AC-DC.

ATmega ha 32KB di memoria flash, 2KB di memoria SRAM, 1KB di EEPROM, 32 registri di lavoro, una interfaccia orientata al byte seriale, un AD a 10 bit (memoria che può essere utilizzata per scrivere programmi: 32256 byte, pari a 63 Kbyte).

Le caratteristiche si possono riassumere con la seguente tabella:

Micro controllore	ATmega328
Tensione di lavoro	5V
Tensione d'ingresso raccomandata	7:12V
Pin I/O digitali	14
Ingressi analogici	6
Memoria flash	32 Kbyte
SRAM	2 Kbyte
EEPROM	1 Kbyte
Clock Speed	16 MHz

### 2.2.2 Utilizzi

Con l'utilizzo di Arduino si possono realizzare in maniera rapida e semplice piccoli dispositivi come controllori di luci, di velocità per motori, sensori di luce e molti altri progetti che utilizzano sensori.

Un altro utilizzo di questa piattaforma è la comunicazione con altri dispositivi.

### 2.2.3 Scelta

Per il nostro progetto si è deciso di utilizzare Arduino Uno per la sua duttilità e semplicità di utilizzo.

Ideato particolarmente per gestire sensori, ho ritenuto che sarebbe stato ideale per la gestione delle luci e del sensore di prossimità.

Un'altra ragione del suo utilizzo è il suo alto range di input voltage, poiché nonostante l'intervallo raccomandato sia 7-12V in caso di "necessità" si può arrivare fino a un minimo di 6V e salire fino al limite di 20V.

Tutto ciò lo rendeva compatibile con entrambi i voltaggi delle schede Jetson TK1 e TX1.



### 2.2.4 Software e librerie istallate

Per lavorare con Arduino è stato necessario istallare sulla Jetson TK1 l'ambiente di sviluppo Arduino IDE per ubuntu 14.04 (scaricando la versione Linux ARM) dal sito ufficiale:

<https://www.arduino.cc/en/Main/Software>

O per comodità se disponibile è possibile scaricare la versione di Arduino direttamente dall'Ubuntu Software Center.

Dopodiché è necessario avere istallato la libreria pySerial, per comunicare in via seriale con la Jetson.

Le informazioni riguardo la libreria e le modalità di installazione sono presenti nel capitolo 2.1.4.

Inoltre per lavorare con le strip led RGB è stato necessario ricorrere ad una libreria che semplificasse la programmazione delle striscie led mediante Arduino.

La scelta è ricaduta sulla libreria NeoPixel [44] di “Adafruit”.

- **NeoPixel:** Questa libreria permette la programmazione di strip led RGB, dando la possibilità di cambiare colore, intensità sia al singolo led sia all'intera striscia.

La libreria è scaricabile dal sito ufficiale:

<https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-installation>



Figura 2.7: Logo Adafruit.

## 2.3 Pulsanti

Il pulsante [45] è un componente che appena premuto collega due punti in un circuito.

Quando il pulsante è aperto (non premuto) non esiste alcuna connessione tra le due gambe del pulsante, quindi il perno è collegato a 5 volt e contrassegnato come HIGH. Quando il pulsante è chiuso (premuto), crea una connessione tra le sue due gambe, collegando il perno a terra, in modo da leggere LOW.

Ho utilizzato due pulsanti (push button) con una funzione di switch (ON/OFF).

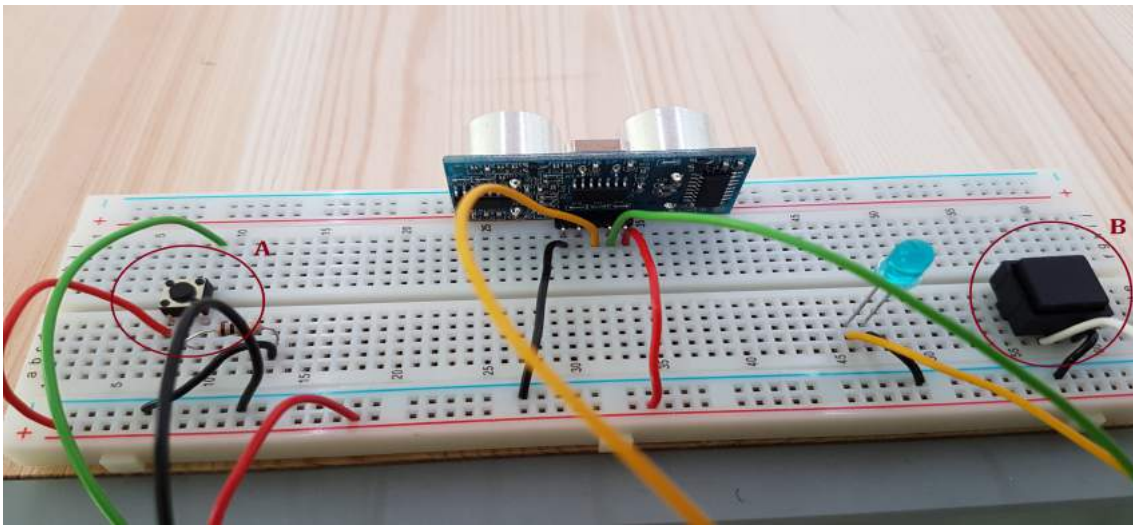


Figura 2.8: Visualizzazione dei pulsanti A e B sulla breadboard.



Figura 2.9: Pulsante “A”.



Figura 2.10: Pulsante “B”.

Il pulsante “A” nella figura 2.9 ha 4 pin ed è stato utilizzato come interruttore di ON/OFF per accendere o spegnere il sensore di prossimità.

Il pulsante “B” nella figura 2.10 ha 2 pin ed è stato utilizzato per eseguire la funzione “scatta” e scattare manualmente la foto senza passare dal sensore di prossimità.

## 2.4 Sensore di prossimità

Spiegherò in seguito il funzionamento del sensore di prossimità (o trasduttore di prossimità ad ultrasuoni) [4] che abbiamo utilizzato:

La potenza emessa da questi trasduttori è molto bassa ed è lo stesso oggetto da rilevare che riflette il raggio verso il ricevitore.

La distanza d'intervento, per questi tipi di sensori, è naturalmente strettamente legata alle caratteristiche superficiali dell'oggetto da rivelare.

Pertanto i costruttori normalmente equipaggiano tali sensori a diffusione con un regolatore di sensibilità per permettere agli utilizzatori di poterli adattare allo specifico impiego.

I sensori di prossimità ad ultrasuoni sfruttano l'emissione di impulsi sonori a frequenza elevata (40 -200 KHz) per rilevare la presenza di oggetti posti nelle loro vicinanze grazie all'eco dovuta alla riflessione degli ultrasuoni da parte degli oggetti stessi.

Il segnale emesso è costituito in genere da un treno di impulsi viaggianti alla velocità del suono nell'aria (340 m/s circa).

L'emissione avviene tramite un apposito trasduttore elettroacustico.

L'onda riflessa è rilevata da un analogo trasduttore la cui funzione è quella di riconvertire i segnali acustici in segnali elettrici.

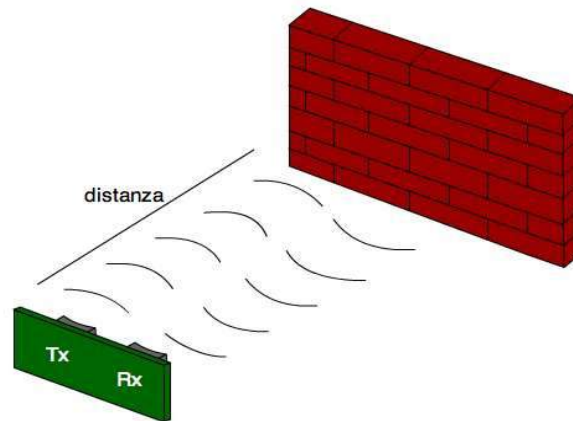


Figura 2.11: Simulazione del sensore di prossimità.

Nelle applicazioni dei sensori di prossimità interessa rilevare la presenza di un oggetto (azionatore) in un certo campo di rilevamento prefissato ed eventualmente programmabile in ampiezza dall'utilizzatore.

Pertanto il tempo di ritardo, cioè il tempo che intercorre fra l'istante di emissione di un impulso e l'istante di ricezione dello stesso, essendo proporzionale alla distanza azionatore/sensore, permette al sensore di stabilire se l'azionatore è o meno in campo.

Un esempio del funzionamento del sensore è visualizzato in figura 2.11.

Il sensore ad ultrasuoni rappresenta dunque una vantaggiosa alternativa nei confronti dei sensori ottici, comunemente utilizzati per rilevare oggetti distanti; infatti le caratteristiche di rilevamento di un sensore a ultrasuoni non dipendono dalle caratteristiche cromatiche superficiali dell'oggetto da rilevare e possono essere sentite anche superfici trasparenti.

Per questo progetto abbiamo optato per il sensore di prossimità SRF-05 (figura 2.12), che si basa sui principi espressi in precedenza.

Questo sensore è compatibile con Arduino, viene utilizzato con una tensione di 5V e riesce ad individuare ostacoli ed oggetti, anche di piccole dimensioni, da 2 centimetri fino a circa 4 metri.

Come tutti i sensori di questo tipo, è insensibile alla luce ambientale, quindi è ottimo per essere usato all'esterno o in condizioni di luce controllata come nel nostro caso.

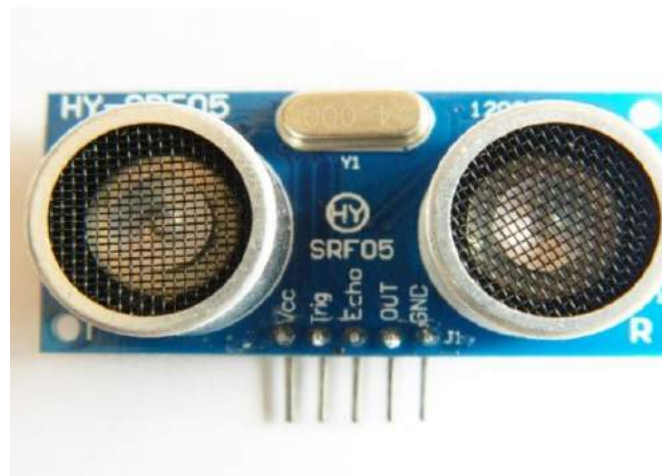


Figura 2.12: Vista anteriore del sensore SRF-05.

In seguito vengono visualizzati i PIN del sensore di prossimità SRF-05, mostrati dettagliatamente in figura 2.13.

**PIN:**

- VCC;
- Trig(T);
- Echo(R);
- OUT;
- GND.

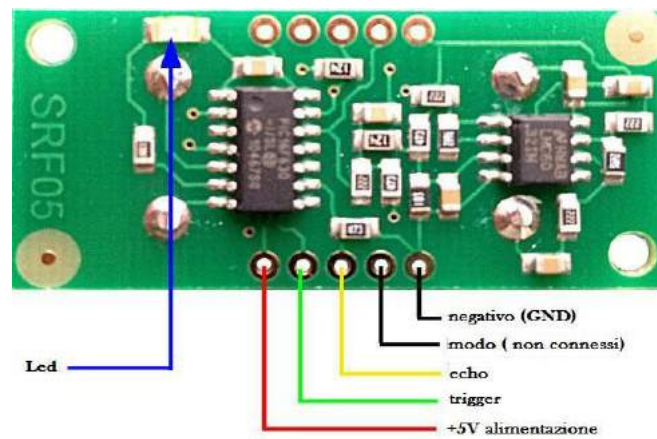


Figura 2.13: Vista posteriore del sensore SRF-05.

Specifiche sensore SRF-05 [46]:

Tensione di alimentazione	5V(DC)
Angolo del sensore	Massimo 15°
Precisione	circa 0.3 cm
Distanza di avvistamento	da 2 cm a 450 cm

Il calcolo per rilevare la distanza da questo sensore è semplice:

$$\text{Spazio} = \text{Velocità} * \text{Tempo}$$

La Velocità è un valore noto in quanto trattandosi di suono la sua velocità è 331,5 m/s (metri al secondo) ad una temperatura di 0° e si può calcolarla alla temperatura ambiente come:

$$331,5 \text{ m/s} + 0,62 * T$$

in cui T è la temperatura misurata in °C per cui a 20°C la velocità del suono è:

$$331,5 + 0,62 * 20 = 331,5 + 12,4 = 343,9 \text{ m/s}$$

che si può arrotondare a 343 m/s, approssimata a 0,0343 cm/microsecondi.

Mentre per la misura della distanza che puoi eseguire con il sensore ad ultrasuoni SRF-05 si può sostituire il valore del tempo appena trovato alla formula della velocità:

$$\text{Spazio} = (0,0343 * \text{Tempo})$$

Il suono viaggia sia in direzione dell'ostacolo sia come eco nel ritorno per cui si deve dimezzare il valore calcolato con la formula precedente:

$$\text{Spazio} = (0,0343 * \text{Tempo}) / 2$$

E da questa formula si ricava la distanza in cm a cui si trova l'oggetto rilevato con cui poi fare le opportune considerazioni.

Un esempio pratico di un oggetto posizionato nel dispositivo è rappresentato in figura 2.14.

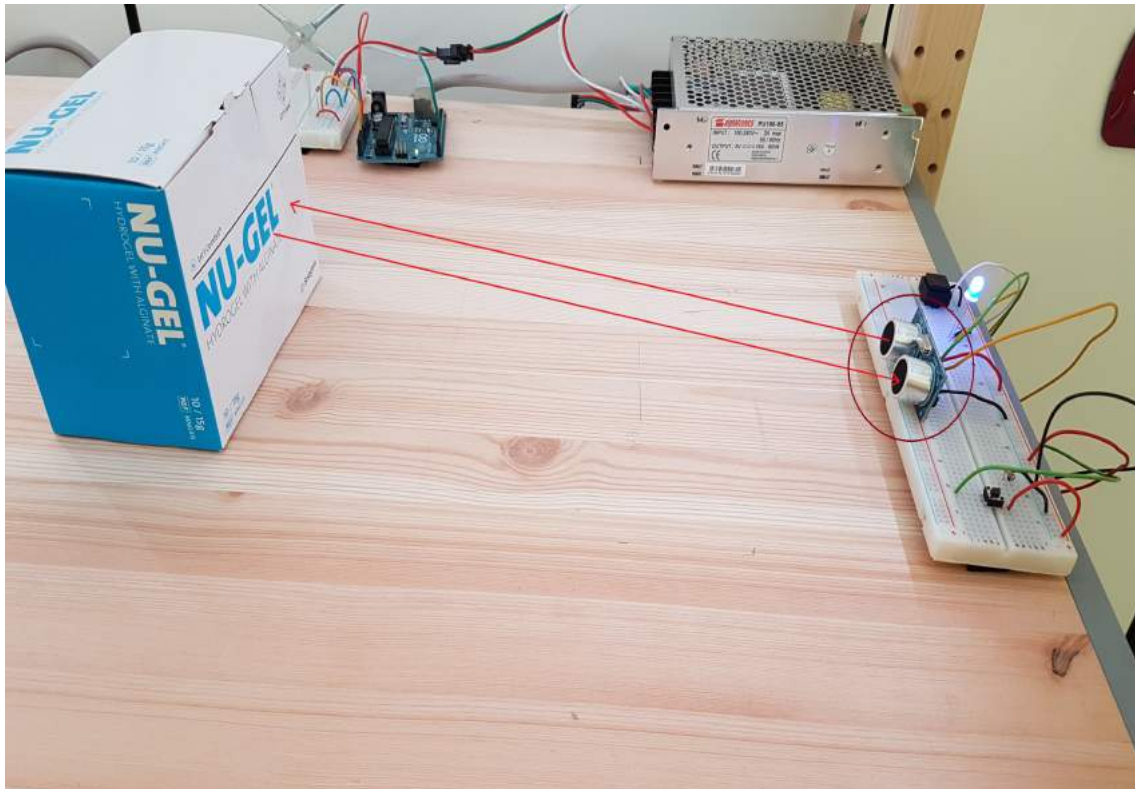


Figura 2.14: Rilevazione del sensore.



## 2.5 Alimentatore

Un alimentatore elettrico [7] è un convertitore AC-DC, ovvero un apparato elettrico, semplice o composto, il cui utilizzo è raddrizzare in uscita la tensione elettrica in ingresso (da alternata AC a continua DC).

Ciò è necessario per fornire energia elettrica adattandola all'uso di altre apparecchiature elettriche, modificando eventualmente anche i livelli di tensione e corrente, e dunque potenza, in uscita attraverso un trasformatore.

Per rendere operativi i led abbiamo utilizzato un alimentatore industriale da 5V e 100W seguendo le specifiche necessarie per alimentare tutta la strip led.

Ho optato per il modello PU100-05 di “Alphatronics”, in figura 2.15, con le seguenti specifiche:

Tensione d'ingresso	220-230V
Tensione d'uscita	5V
Corrente d'uscita	16000 mA
Potenza	80W

Il modello d'alimentatore era del tipo "100W" ma erogava realmente 80W.

Questa potenza è risultata sufficiente poiché anche se i led che avrebbe dovuto alimentare richiedevano circa 90W (controllare capitolo 2.6.2), collegando entrambe le estremità della strip led all'alimentatore il problema era stato risolto.



Figura 2.15: Alimentatore PU100-05.



## 2.6 Striscia led

La striscia led (o strip led), in figura 2.16, è una scheda flessibile popolata da diodi luminosi montati in superficie (SMD LEDs) [47] e altri componenti.

Di solito è dotata di supporto adesivo per un montaggio adatto a qualsiasi genere di superficie.



Figura 2.16: Striscia led.

Esistono vari tipi di strip led che si differenziano a seconda:

- del colore;
- del grado di protezione;
- della tensione di pilotaggio.

**Colore [10]:**

Singolo colore, non indirizzabile	Ogni LED è di un singolo colore bianco, tipicamente compreso tra 2700K e 6500K nella temperatura del colore, oppure di uno qualsiasi dei diversi colori monocromatici che coprono l'intervallo dello spettro visibile.
Multicolore, non indirizzabile	Ogni LED è in grado di visualizzare rosso, verde, blu o tutti e tre (bianco), guidati da tre guide d'ingresso. Tutti i LED mostrano lo stesso colore in qualsiasi momento, ma il colore può essere manipolato variando la tensione applicata a ciascuno dei tre ingressi di potenza.
RGB, indirizzabile	Colori e indirizzi multipli. Ogni LED ha un proprio chip che può essere attivato individualmente per cambiare il colore o modificare la propria intensità luminosa.

Per il nostro progetto ho scelto di utilizzare una striscia led RGB indirizzabile (figura 2.17), in modo da poter controllare qualsiasi dettaglio della luce, dalla sua intensità al suo colore.

Infatti con questa tipologia di led ho potuto decidere una qualsiasi quantità di rosso (R), verde (G) e blu (B) per scegliere diverse tipologie di luce.

**Grado di protezione [48]:**

Il grado di protezione IP è indicato con due cifre caratteristiche.

La prima cifra indica il grado di protezione contro la penetrazione di corpi solidi estranei:

IP	Significato
0	nessuna protezione
1	protetto contro corpi solidi superiori a 50 mm di diametro
2	protetto contro corpi solidi superiori a 12 mm di diametro
3	protetto contro corpi solidi superiori a 2,5 mm di diametro
4	protetto contro corpi solidi superiori a 1 mm di diametro
5	protetto contro le polveri (nessun deposito nocivo)
6	totalmente protetto contro le polveri

La seconda cifra indica il grado di protezione contro la penetrazione di liquidi:

IP	Significato
0	nessuna protezione
1	protetto contro le cadute verticali di gocce d'acqua
2	protetto contro le cadute di gocce d'acqua o pioggia fino a 15° dalla verticale
3	protetto contro le cadute di gocce d'acqua o pioggia fino a 60° dalla verticale
4	protetto contro gli spruzzi d'acqua da tutte le direzioni
5	protetto contro i getti d'acqua
6	protetto contro i getti d'acqua potenti
7	protetto contro gli effetti delle immersioni temporanee
8	protetto contro gli effetti delle immersioni continue

Nella scelta della strip led ho optato per scegliere il grado di protezione IP30, poiché tra le varie opzioni era quella che si adattava maggiormente al nostro caso, dato che non dovevo esporre il dispositivo all'acqua o a particolari condizioni.

**Tensione di pilotaggio:**

Tensione	Dispositivi per l'alimentazione
5V	Alimentatore, o cavo USB
12V	Alimentatore
24V	Alimentatore

Per il dispositivo ho scelto una strip led che funzionasse con 5V in modo tale da renderla compatibile con Arduino, che sarebbe stato utilizzato con la funzione di controllore.

### 2.6.1 Scelta

Per le ragioni e le considerazioni espresse in precedenza ho deciso di utilizzare delle strisce led RGB da 5 metri del tipo WS2812B [49] con 60leds/m.

Queste strisce, rappresentate nella figura 2.17, sono facilmente programmabili con Arduino e montano dei led della categoria 5050 SMD.



Figura 2.17: Striscia led RGB.

I led SMD (Surface Mounting Device) [50] vengono montati tramite una tecnica utilizzata in elettronica per l'assemblaggio di un circuito stampato prevedente l'applicazione dei componenti elettronici sulla sua superficie senza la necessità di praticare dei fori.

Le ridotte dimensioni dei LED SMD e la flessibilità nel loro impiego li ha resi ideali al mio scopo, dato che la striscia led doveva essere molto elastica per poter essere disposta sul congegno.

Questa tipologia di Led è illustrata di seguito in figura 2.18 e 2.19.

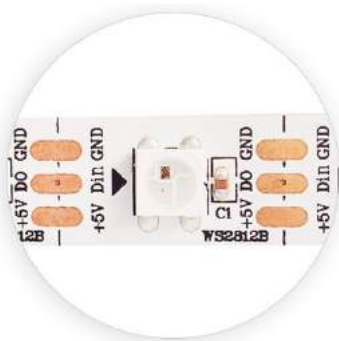


Figura 2.18: Singolo led RGB.

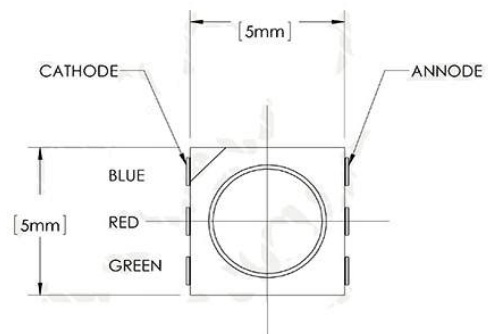


Figura 2.19: Schema singolo led RGB.

Inoltre le strisce dispongono di un adesivo, come mostrato in figura 2.20, per attaccare facilmente il tutto al dispositivo in modo saldo.



Figura 2.20: Adesivo striscia led.

## 2.6.2 Caratteristiche led scelti

Modello	WS2812B strip 5050 RGB
Voltaggi di d'ingresso	5V DC
Potenza	18W/m
Lunghezza	5m
Angolo di illuminazione	120°

La striscia WS2812B lavora con 5V DC, ciascun LED è indirizzabile singolarmente, con 8 bit di dati rossi, verdi e blu utilizzabili per colori a 24 bit.

Le strisce prendono i dati in ordine verde, rosso e blu.

Tutta la luce si diffonde in maniera molto uniforme.

Ogni singolo LED può essere tagliato arbitrariamente, la striscia ha una buona flessibilità per essere piegata a seconda delle proprie necessità.

Questo tipo di Led è di alta qualità infatti la durata della sua vita è maggiore di 50.000 ore. In totale si hanno 300 led e ognuno di essi per essere alimentato necessita di 0,3 W di alimentazione.

Infatti le strisce led per essere illuminate completamente richiedono:

$$(18\text{W/m}) * 5\text{m} = 90\text{W}$$

Ho provato ad alimentarle con un alimentatore da 80W, ma i led tendevano con il passare dei secondi a un colore rosso acceso come mostrato in figura 2.21 e 2.22.

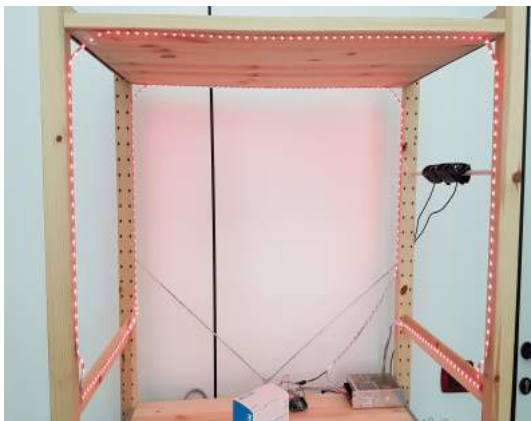


Figura 2.21: Test accensione dei led montati sul dispositivo, senza collegare entrambe le estremità all'alimentatore.



Figura 2.22: Test accensione dei led alla massima potenza, senza collegare entrambe le estremità all'alimentatore.

Ciò era causato da una caduta di potenziale, perciò è stato necessario collegare entrambe le estremità della strip led all'alimentatore.

Ciò è rappresentato con uno schema in figura 2.23.

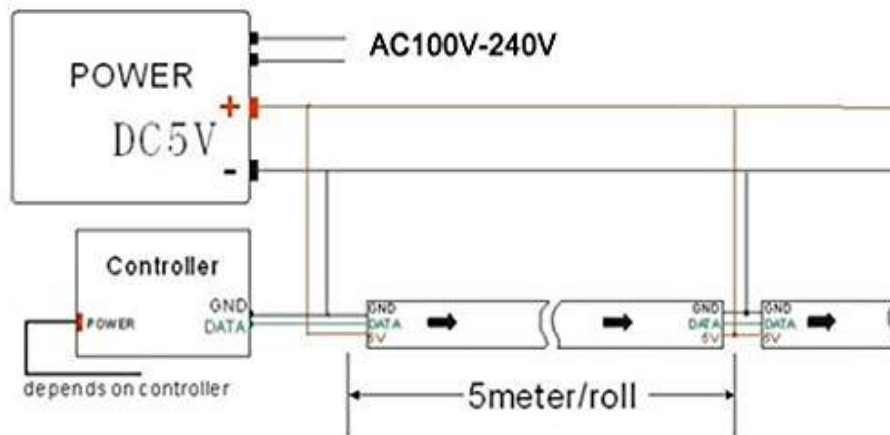


Figura 2.23: Schema collegamenti striscia led RGB.

Dopo questi test ho contattato il produttore delle suddette strisce, constatando che erano necessari anche meno watt (W).

Infatti ho utilizzato un alimentatore della categoria 100W ma che erogava effettivamente 80W.

Si è così verificato il corretto funzionamento delle strisce led, come è mostrato nella figura 3.27.

Questi led sono stati scelti appositamente poiché compatibili con la libreria NeoPixel [44], poiché servivano dei led programmabili in modo veloce e duttile.

Le informazioni sulla libreria e su come installarla sono presenti al capitolo 2.2.4.

## 2.7 Fotoresistore

La fotoresistenza (o fotoresistore) [6], rappresentato in figura 2.24, è un componente elettronico la cui resistenza è inversamente proporzionale alla quantità di luce che lo colpisce.

Si comporta come un tradizionale resistore, ma il suo valore in ohm diminuisce a mano a mano che aumenta l'intensità della luce che la colpisce.

Ciò comporta che la corrente elettrica che transita attraverso tale componente è proporzionale all'intensità di una sorgente luminosa.

In tale maniera si può rilevare l'intensità luminosa nell'ambiente.



Figura 2.24: Fotoresistore utilizzato.

Ho usato il fotoresistore come sensore di luminosità per riuscire a rilevare l'intensità luminosa presente all'interno del congegno in una scala che va da 0 a 1023.

Più alto era il numero più intensità luminosa era presente.

Tramite queste rilevazioni sono state fatte alcune considerazioni per rendere più efficiente la programmazione delle strip led per la gestione real-time della luminosità dentro al dispositivo.

Queste analisi verranno trattate successivamente nel capitolo 3.3.2.

Nella figura 2.26 viene descritto lo schema del fotoresistore in figura 2.25.



Figura 2.25: Fotoresistore.

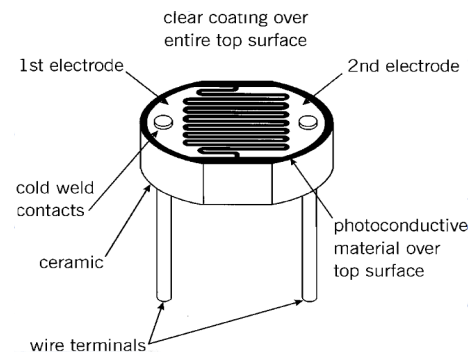


Figura 2.26: Schema del fotoresistore.



## 2.8 Spettrofotometro

### 2.8.1 Caratteristiche

La spettrofotometria [51] è una tecnica analitica, qualitativa e quantitativa che permette il riconoscimento e la quantizzazione di una sostanza in base al suo spettro di assorbimento della luce.

L'analisi spettrofotometrica è applicabile, con determinati accorgimenti a tutte le sostanze; se la sostanza in esame è di per sé fotoassorbente alla lunghezza d'onda scelta è possibile una misura diretta.

Per effettuare tali misurazioni si ricorre ad un dispositivo di rilevazione chiamato spettrofotometro.

### 2.8.2 Scelta

Ho deciso di utilizzare uno spettrofotometro per poter rilevare la luce ambientale. Infatti grazie a questo dispositivo abbiamo ottenuto i lux [9] presenti all'interno del nostro congegno in modo da poter basare poi i calcoli sui risultati ottenuti da tali rilevazioni.

Ciò è stato necessario per poter programmare correttamente le strisce led.

Tutti i calcoli e le specifiche seguite sono disponibili nel capitolo 3.3.2.



Figura 2.27: Kit di calibrazione i1Basic Pro 2.

Lo spettrofotometro utilizzato è l' i1Basic Pro 2 della “X-Rite” [52] mostrato in figura 2.28.



Figura 2.28: Spettrofotometro i1Basic Pro 2.

Il dispositivo permette una misurazione e una calibrazione del colore particolarmente precise.

Lo spettrofotometro i1Pro 2 [53] introduce un nuovo livello di precisione cromatica, garantendo una maggiore semplicità d'uso e un maggior numero di funzionalità di livello professionale.

Esso permette la riduzione del tempo garantendo la sicurezza che il flusso digitale dell'utente sia sempre calibrato e profilato perfettamente.

Per poter effettuare le rilevazioni e utilizzare lo spettrofotometro è stato necessario scaricare il software i1Profiler [54] (figura 2.29) per il suo utilizzo.

La versione utilizzata per le rilevazioni di questo progetto è la 1.7.

Il programma è scaricabile dal sito ufficiale:

[http://www.xrite.com/service-support/downloads/i/i1profiler-i1publish\\_v1\\_7\\_0](http://www.xrite.com/service-support/downloads/i/i1profiler-i1publish_v1_7_0)

Con l'ultima versione del software i1Profiler v 1.7 (figura 2.29), X-Rite ha ulteriormente potenziato questa soluzione professionale, offrendo un livello di flessibilità e presentando diverse funzionalità esclusive di gestione del colore, quali l'ottimizzazione dei profili e la creazione di profili con misurazioni della luce ambiente. Gli utenti possono scegliere tra un'interfaccia “di base” con configurazione guidata, oppure un'interfaccia “avanzata” configurabile dall'utente per creare profili colore di alta qualità, precisi e personalizzati per monitor, proiettori, stampanti e macchine da stampa.

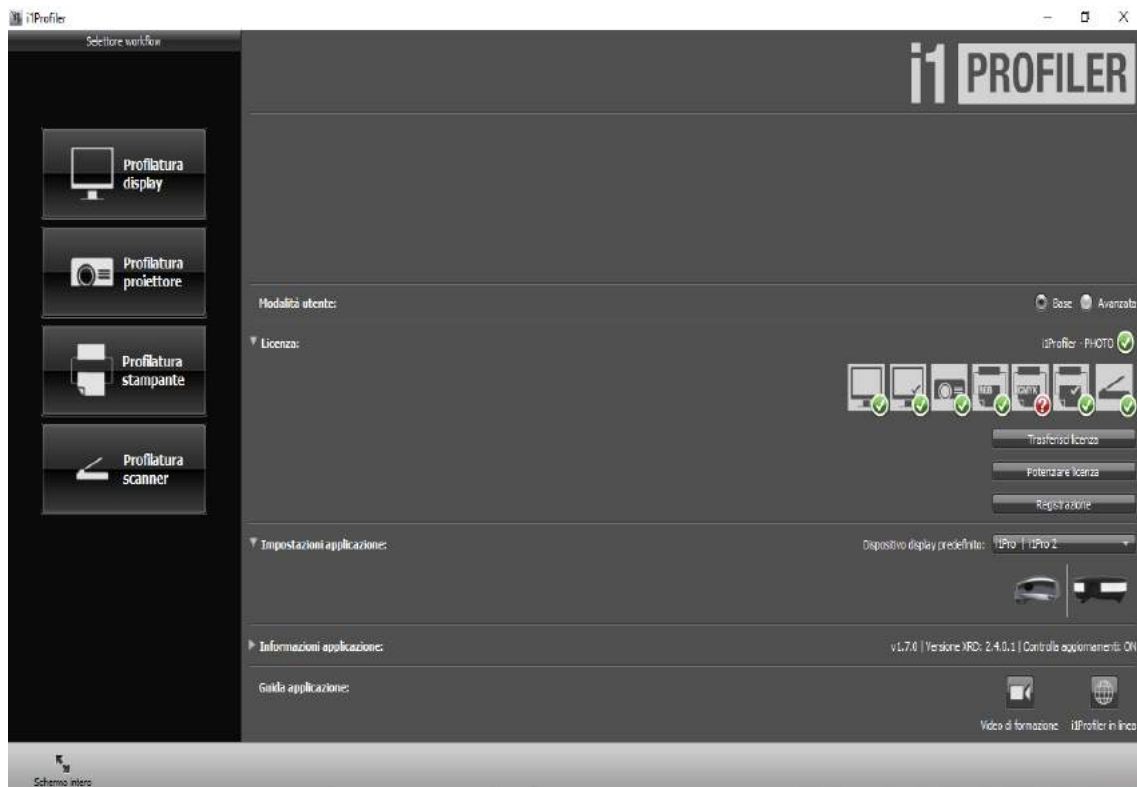


Figura 2.29: Schermata i1Profiler v 1.7.

## 2.9 Cablaggio, resistenze, breadboard e singolo led

### 2.9.1 Cablaggio

Come cablaggio si è scelto di utilizzare dei cavi di rame, chiamati Jumper (figura 2.30), presi al metro in modo tale da poter scegliere volta per volta le lunghezze necessarie ai collegamenti.

Sono stati presi cavi di diverso colore per semplificare i collegamenti, dividendoli per categorie: ad es. nero per la messa a terra (GND) e rosso per l'alimentazione (5V).

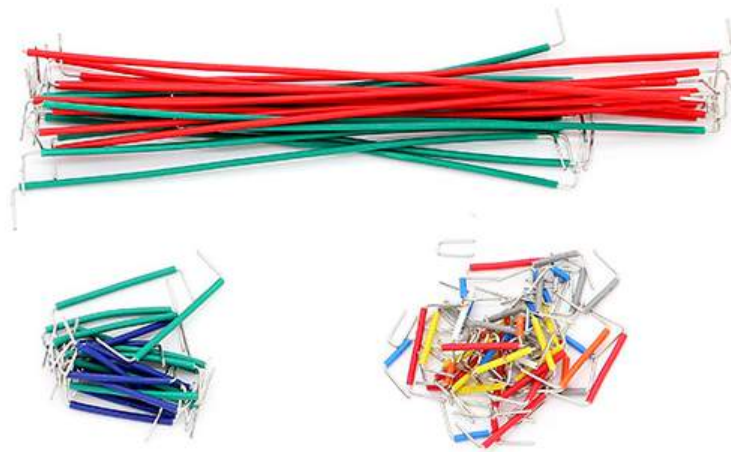


Figura 2.30: Cavi Jumper.

### 2.9.2 Resistenze

La resistenza [23] è una forza fisica che, se sottoposta ad una tensione, si oppone al passaggio della corrente elettrica in un materiale semiconduttore.

Questa forza fisica può variare a seconda di molte caratteristiche, come ad esempio il materiale con cui è realizzata, la potenza che riesce a sviluppare, le sue dimensioni e soprattutto dalla sua temperatura di esercizio.

E' importante quindi dosare il flusso della corrente elettrica in un circuito, sia per controllare il flusso della corrente stessa, e sia per bloccare eventuali sbalzi di corrente che possano danneggiare il circuito.

L'unità di misura delle resistenze prende il nome di "ohm".

Per il nostro progetto abbiamo utilizzato due resistenze entrambe da  $10\text{k}\Omega$ , come in figura 2.31, poiché ritenuto che tale resistenza bastasse per mantenere stabile il circuito.



Figura 2.31: Resistenza  $10\text{k}\Omega$ .

### 2.9.3 Breadboard

Una breadboard [55] (figura 2.32) è uno strumento utilizzato per creare prototipi di circuiti elettrici.

A differenza della basetta millefori, che è un circuito stampato su cui vengono saldati i componenti e i collegamenti che formano il prototipo, la breadboard non richiede saldature ed è completamente riutilizzabile.

Sebbene venga usata normalmente per la prototipazione di circuiti semplici, può essere usata anche per testare interi calcolatori.

Tutte le breadboard hanno generalmente una struttura simile composta da linee di trasmissione (strips) che consistono in collegamenti elettrici tra i fori.

Le linee di alimentazione, che sono generalmente poste ai lati e collegate lungo tutto l'asse, e le linee dedicate ai componenti, collegate in posizione perpendicolare alle linee d'alimentazione e più corte.

Spesso le breadboard sono componibili, cioè è possibile collegare più basette tra di loro per ampliare le linee a disposizione.

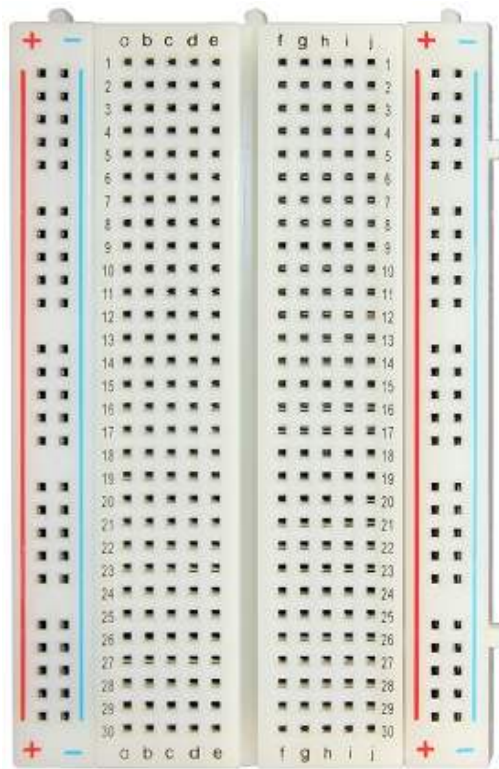


Figura 2.32: Breadboard.

### 2.9.4 Singolo led

Diodo a semiconduttore che al passaggio di corrente elettrica emette radiazioni luminose.

Il singolo led è rappresentato in figura 2.33.



Figura 2.33: Singolo led.

Abbiamo utilizzato questo singolo led come segnale di ON/OFF.

Quando il sensore di prossimità rilevava un oggetto nel suo raggio d'azione allora il led si accendeva, altrimenti rimaneva spento.

Ciò rendeva più semplice per chi utilizzava il dispositivo controllare se l'oggetto appena inserito venisse rilevato correttamente.

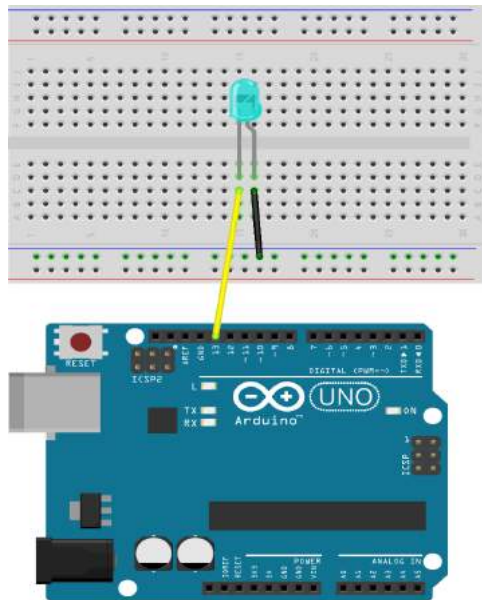


Figura 2.34: Rappresentazione singolo led.

## 2.10 Fotocamere

Per il nostro progetto ho pensato a diverse soluzioni e alternative per quanto riguarda l'utilizzo e il tipo di camera da usare.

Alla fine la scelta è ricaduta su due fotocamere USB, in modo da rendere facile il loro utilizzo grazie al plug and play.

Il modello di camere USB utilizzato è il c525 della "Logitech" [56] (figura 2.36).

In seguito nel capitolo 4.2 si parlerà delle migliorie pensate per il congegno per quanto riguarda le fotocamere, tra cui utilizzo di camere RAW [12] e camere a bus CSI [22].



Figura 2.35: Fotocamere c525 posizionate sul dispositivo.



Figura 2.36: Vista fotocamere c525.



## Capitolo 3

# Funzionamento

Ora andremo ad analizzare nei dettagli il funzionamento del dispositivo.

Ho cercato di rendere questo congegno il più “user-friendly” possibile, poiché l’interesse primario era rendere comprensibile il suo funzionamento facendo in modo che il suo utilizzo fosse semplice e veloce.

Abbiamo creato un diagramma di flusso (o Flow chart) che spiegasse in modo riassuntivo tutti i passi che il dispositivo andrà a compiere.

L’obiettivo era di rendere comprensibile, in modo rapido, il nostro lavoro a chiunque volesse testarlo.

In questo capitolo si presuppone che si siano già seguiti i passi del capitolo 2, perciò si tiene conto che se si volesse provare o riprodurre il dispositivo, il software e l’hardware siano stati già correttamente settati e preparati.

## 3.1 Diagramma di flusso

Nella figura 3.1 mostrerò il diagramma di flusso nella sua interezza, ma successivamente, per una spiegazione più efficace, andrò ad analizzarlo diviso in due macro sezioni:

- Sezione A:
  - sensore di prossimità;
  - scatto manuale/automatico ed acquisizione;
  - preprocessing;
  - creazione modello 3D.
- Sezione B:
  - sensore di luminosità;
  - led RGB;
  - gestione luce in real-time.

La parte di elaborazione con immagini RAW, di demosaicing e di color mapping completa non sono state inserite in questo diagramma di flusso poichè utilizzando camere USB e non camere RAW non era possibile procedere in questa direzione.

Abbiamo utilizzato per il preprocessing un algoritmo di White Balancing.

Il flow chart completo anche con la parte delle immagini RAW, di demosaicing e di color mapping si trova nel capitolo 4.3 quando parleremo degli aggiornamenti disponibili.

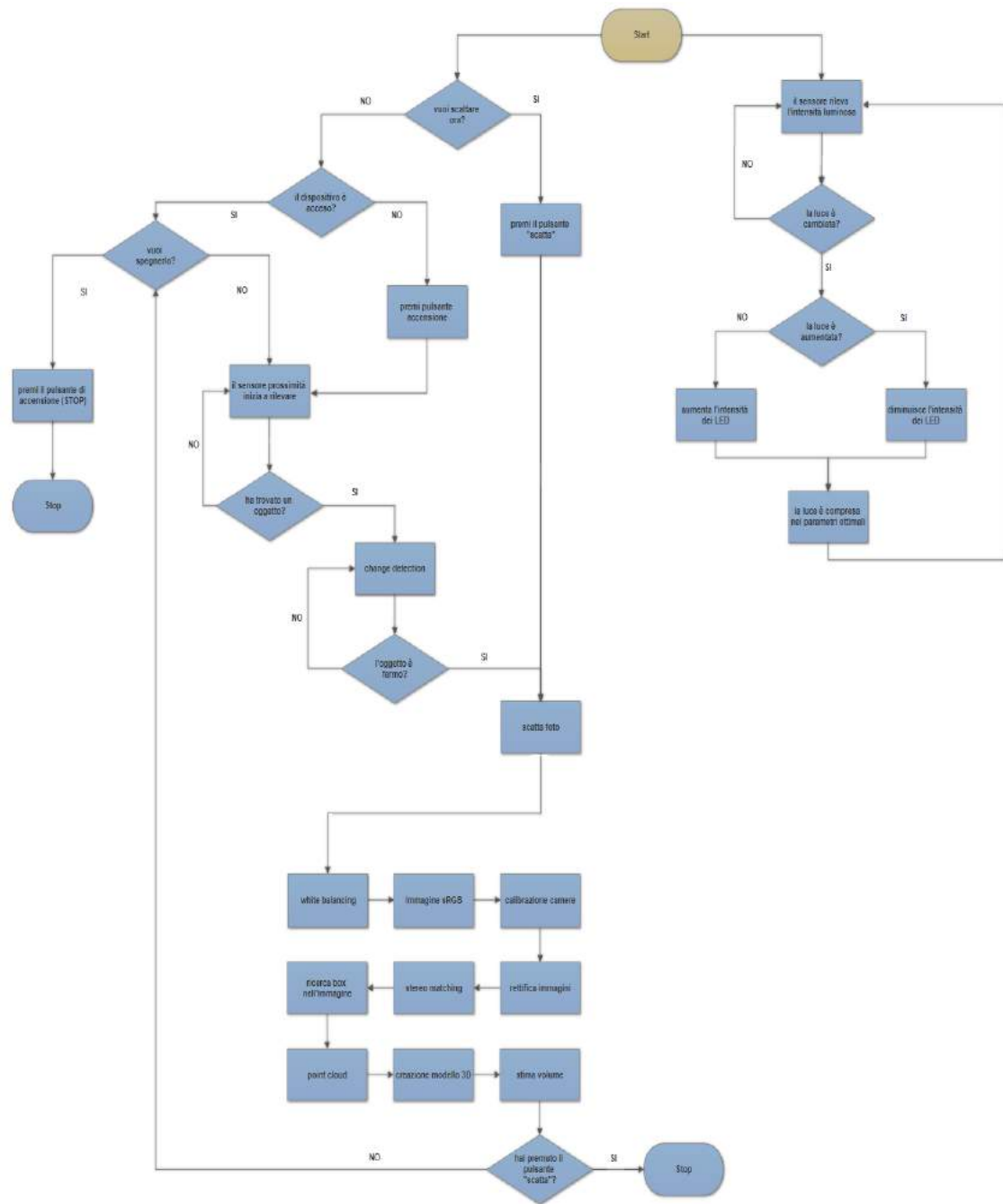


Figura 3.1: Flow chart per fotocamere USB.

## 3.2 Sezione A

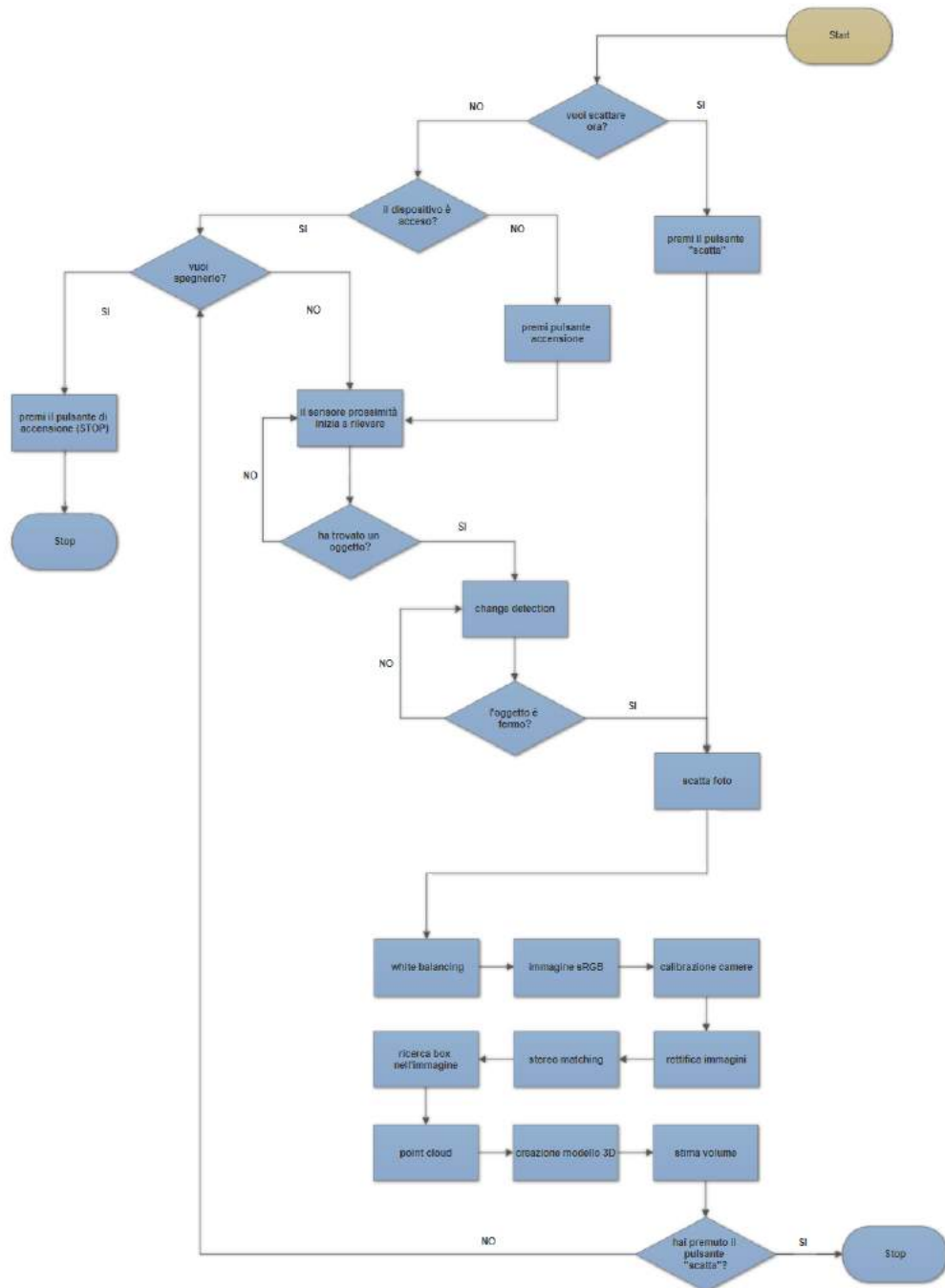


Figura 3.2: Flow chart sezione A.

In questa sezione andrò ad analizzare per passi il funzionamento del dispositivo, in figura 3.3, per quanto riguarda l'utilizzo del sensore di prossimità, lo scatto e ciò che ne consegue.



Figura 3.3: Vista del dispositivo completo.

### 3.2.1 Costruzione e collegamenti

I collegamenti per questa sezione sono molto semplici e non mi dilungherò. Si tratta di:

- collegare le due fotocamere USB alla Jetson TK1 (abbiamo usato un HUB USB a 10 porte poichè la Jetson ha solo un ingresso USB);
- collegare Arduino (e quindi la parte relativa al sensore di prossimità) tramite il proprio cavo USB alla Jetson TK1.

### 3.2.2 Avvio e modalità

Una volta accesa la Jetson l'Arduino relativo al sensore di prossimità sarà automaticamente collegato e attivato (se ciò non avvenisse consultare il capitolo 3.4).

Ovviamente su questo Arduino sarà necessario caricare la prima volta che lo si utilizza (poi sarà sempre salvato e quindi utilizzabile senza alcun caricamento) il programma:

```
acquisizione.ino
```

Per rendere funzionante il dispositivo è necessario eseguire dal terminale (prompt dei comandi) della Jetson TK1 il programma:

```
comunicaFinale.py
```

Per attivarlo bisognerà entrare nella corretta directory (quella dove si trovano i programmi) e utilizzare il comando:

```
python comunicaFinale.py
```

I programmi a cui si fa riferimento sono consultabili al capitolo 4.3

Una volta avviato il programma avremo due possibili scelte per poter utilizzare il dispositivo:

- Modalità automatica;
- Modalità manuale.

Tutto il processo, dal momento della rilevazione fino al termine e quindi alla ricostruzione del modello 3D, impiega poco meno di 1 minuto.

### 3.2.3 Modalità automatica

Questa opzione permette di attivare il sensore di prossimità per far sì che si possa rilevare un oggetto automaticamente, scattare due fotografie (una per la Camera SX e una per la Camera DX) e inviarle alla Jetson TK1 che elaborerà poi il tutto. Per attivare il sensore di prossimità occorrerà premere il pulsante specifico, denominato come pulsante “A” (spiegazione al capitolo 2.3).

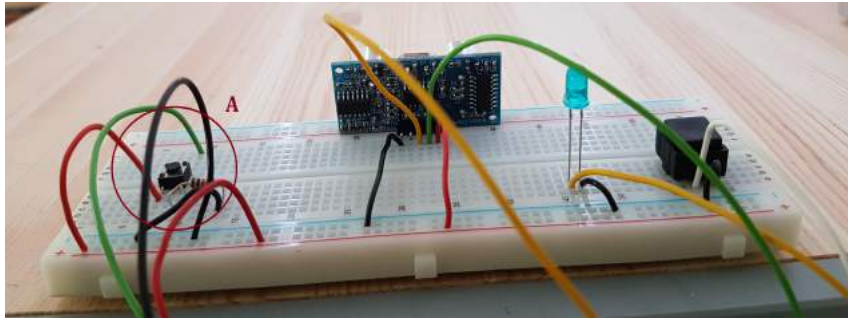


Figura 3.4: Pulsante A sulla breadboard.

Una volta premuto il pulsante, il sensore di prossimità SRF-05 sarà operativo e pronto alla rilevazione in real-time.

In questo momento il congegno è in attesa che accada un evento, ad esempio il posizionamento di un oggetto all'interno del dispositivo come in figura 3.5.

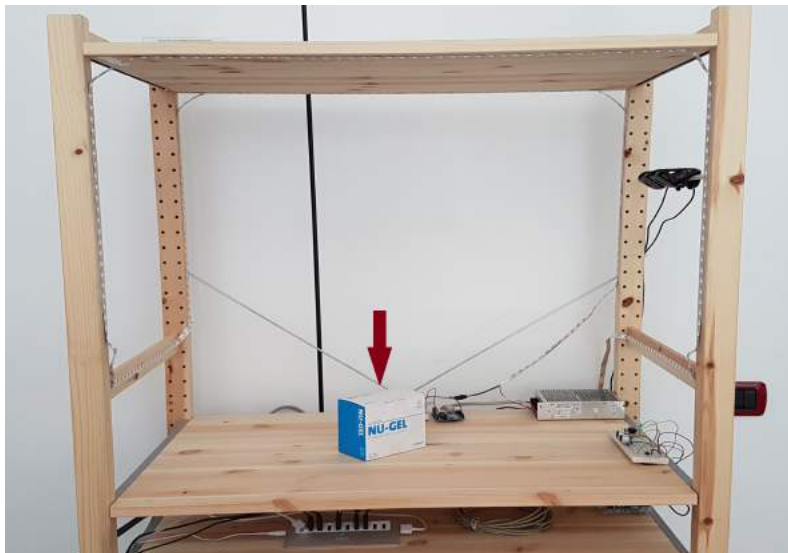


Figura 3.5: Oggetto posizionato nel macchinario.

Il dispositivo resterà in attesa fino a che un oggetto non entrerà nel campo di rilevazione del sensore di prossimità.

Per essere rilevato è necessario che almeno un bordo dell'oggetto debba toccare anche minimamente l'angolo del raggio d'azione del sensore.

Un esempio del meccanismo di rilevazione è rappresentato in figura 3.6.

E' stata settata come massima distanza di rilevazione del sensore 60 cm, in modo tale da rilevare solo gli oggetti nel congegno senza ricevere interferenze da ciò che è al di fuori del dispositivo.

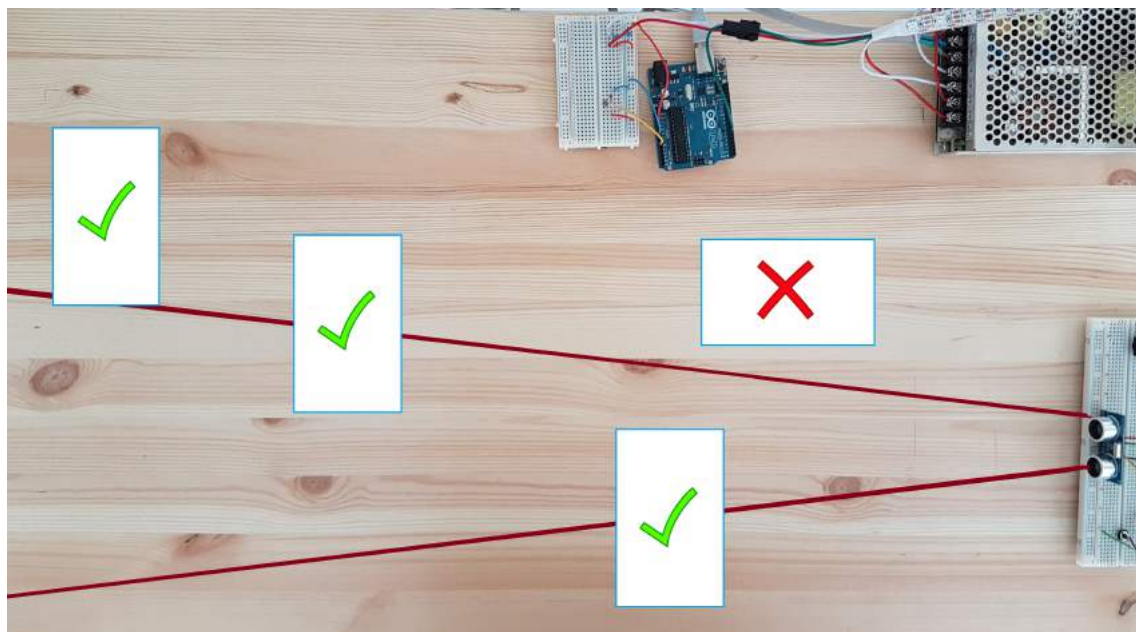


Figura 3.6: Esempio raggio di rilevazione dispositivo.



Figura 3.7: Rilevato dal sensore di prossimità.



Figura 3.8: Non rilevato dal sensore di prossimità.



Una volta rilevato l'oggetto, un led blu posizionato accanto al sensore si illuminerà. Ciò è mostrato in figura 3.9 e permette di controllare comodamente ad occhio nudo se l'oggetto è stato individuato.

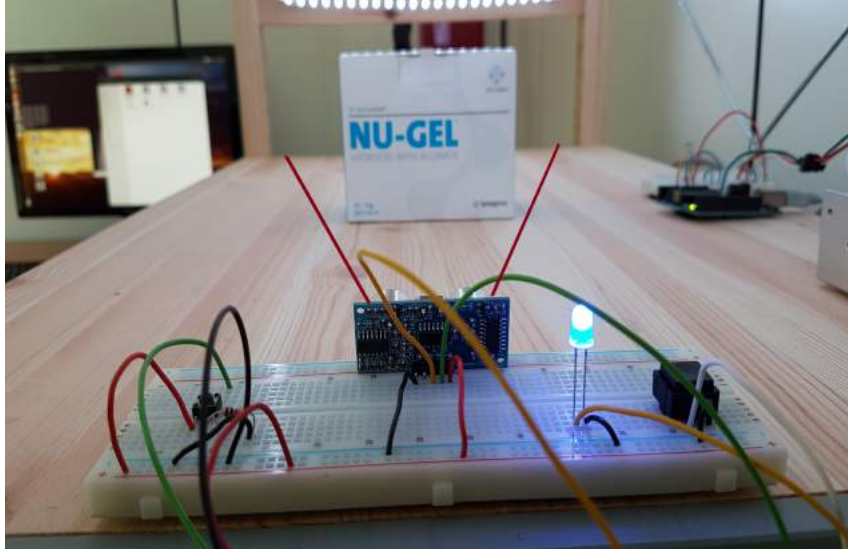


Figura 3.9: Accensione del led per l'avvenuta rilevazione dell'oggetto.

Dopo questa fase Arduino va in stand-by e viene inviato alla Jetson TK1 un segnale di avvenuta rilevazione.

Dopodiché la scheda esegue rapidamente un algoritmo di change detection [57].

**Change Detection:** rilevamento dei cambiamenti in immagini della stessa scena acquisite in istanti differenti.

- Input: due o più immagini della scena;
- Output: immagine binaria, detta “Change Mask”, che ad ogni pixel associa uno fra due valori c (“changed”) o u (“unchanged”), a seconda che il pixel presenti “cambiamenti significativi” o meno.

Vengono presi continuamente frame dalle foto scattate dalle due fotocamere.

Ovviamente sarà effettuato un controllo alle camere per vedere se sono rilevate e attive.

Dopodiché i frame estrapolati sono portati in scala di grigi, per semplificare il confronto tra il frame appena ottenuto e quello precedente.

Se l'algoritmo non rileva una differenza significativa tra i frame delle immagini confrontate allora notifica attraverso un flag l'avvenimento (poiché vuol dire che l'oggetto è immobile e pronto ad essere acquisito), altrimenti continua a confrontare i frame.

Nell'immagine 3.10 a sinistra si nota il frame precedente, mentre a destra il nuovo. I frame vengono trasformati in scala di grigi e confrontati tramite change detection. Il risultato sarà che l'oggetto si trova in movimento e quindi il dispositivo non scatterà ancora alcuna foto.



Figura 3.10: Change detection, trasformazione in scala di grigi.

Ciò è stato necessario per evitare che le camere scattassero istantaneamente appena l'oggetto fosse rilevato, perché esisteva il forte rischio che l'oggetto fosse ancora in movimento, oppure che mentre lo si posizionasse lo scatto immediato immortalasse anche le mani e il braccio di chi stava posando l'oggetto come in figura 3.11.



Figura 3.11: Errore scatto, mano all'interno dell'immagine.

Una volta che l'algoritmo di change detection notifica che l'oggetto è fermo nella sua posizione, si ha il via libera per poter far scattare le due fotocamere.

Ottenute le due immagini, si prosegue con il preprocessing, attuando prima di tutto un bilanciamento del bianco (White balancing) [14] spiegato brevemente nella parte di color constancy del capitolo 1.1.2.

Si può notare nella figura 3.12 lo scatto della fotocamera sinistra, mentre nella figura 3.13 lo scatto della fotocamera destra.

In entrambe le immagini a sinistra si trova l'immagine originale, a destra quella dopo il white balancing.

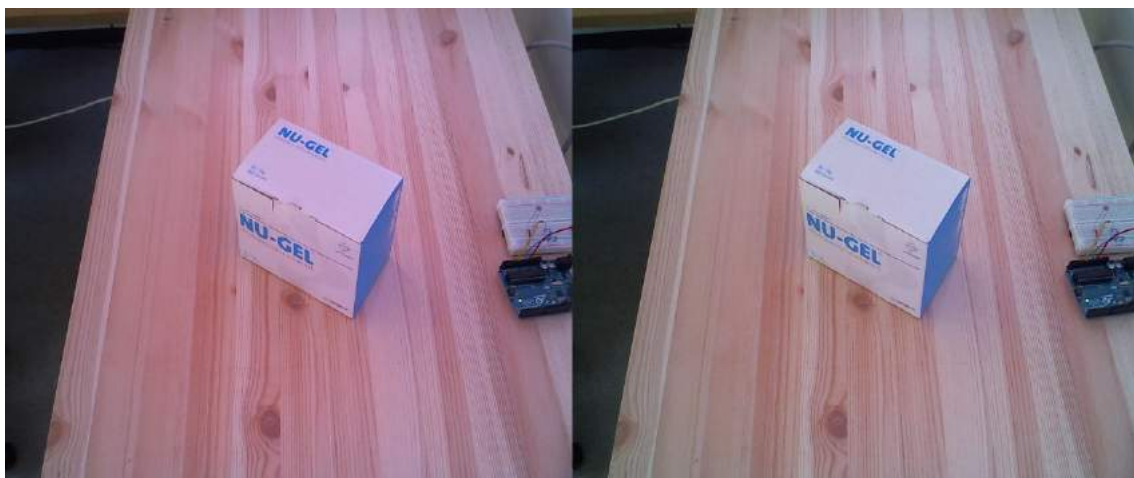


Figura 3.12: Applicazione white balancing scatto SX.

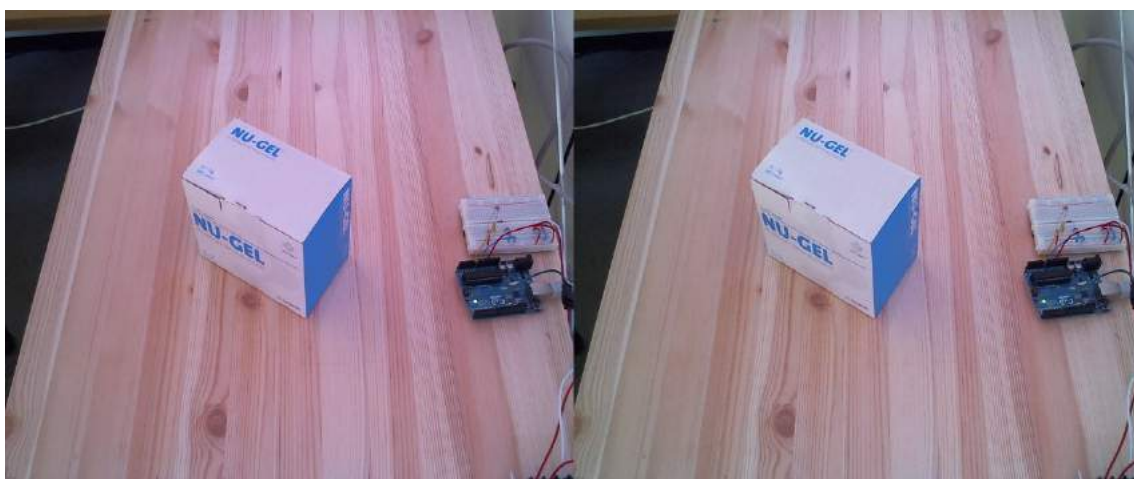


Figura 3.13: Applicazione white balancing scatto DX.

Concluso il preprocessing si procedeva alla ricostruzione del modello 3D e al salvataggio delle immagini in una cartella sulla Jetson.

Al termine di tutte le fasi Arduino, che si trovava in stand-by, tornava nuovamente in modalità di ricezione, pronto a iniziare nuovamente il processo.

Era importante però risolvere una questione maturata durante queste prove, poiché se l'oggetto veniva rilevato, una volta completata l'acquisizione e l'elaborazione delle immagini, si poteva creare questa situazione:

- se l'oggetto non era rimosso dal dispositivo, esso veniva rilevato ulteriormente come se fosse stato appena inserito.

Bisognava creare una distinzione tra gli oggetti che rimanevano fermi sul congegno e quelli nuovi, appena inseriti.

Perciò per risolvere il problema si è rivelato necessario ragionare per “stati”:

- se l'oggetto era posizionato nel dispositivo e non veniva mai rimosso o spostato, una volta rilevato ed elaborate le sue immagini, il congegno rimaneva in attesa, senza ripetere inutilmente il processo;
- se l'oggetto veniva spostato o rimosso e reinserito uno nuovo (o il medesimo) allora ripartiva la fase di acquisizione.

## Console di controllo

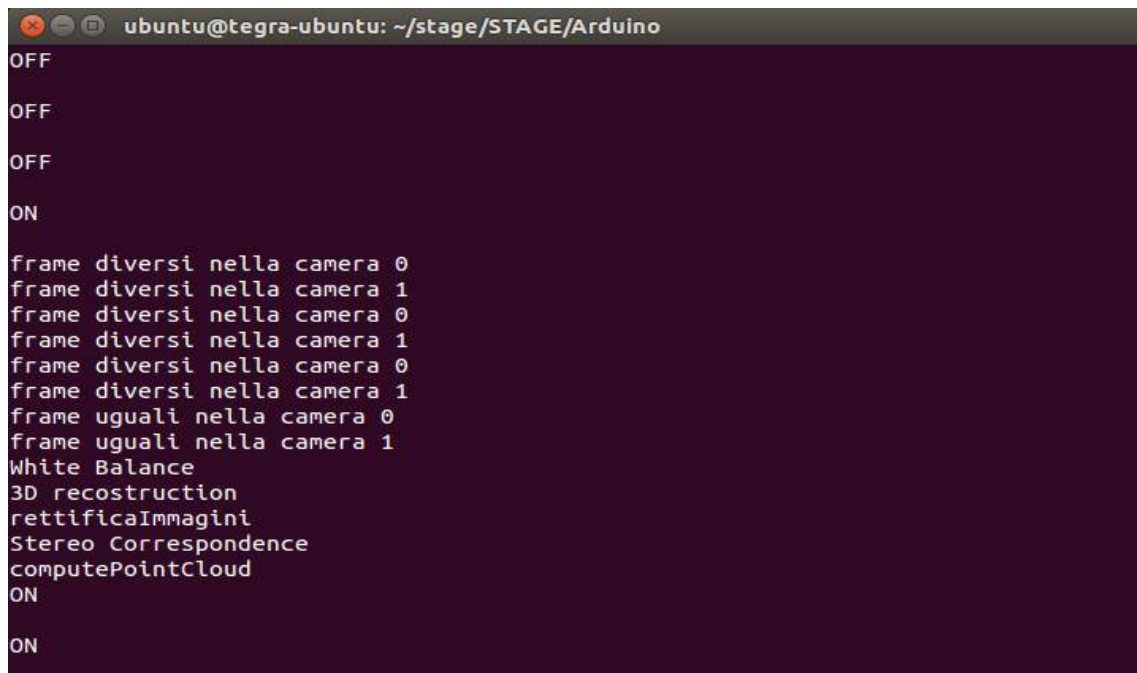
Questo è ciò che accade sul monitor, mentre si utilizza la modalità automatica del dispositivo:

Dopo avere avviato il programma e premuto il pulsante di accensione del sensore di prossimità, sul monitor saranno mostrati una serie di OFF (scanditi nel tempo ogni mezzo secondo circa), che mostreranno che il sensore è acceso, ma non rileva nulla.

Quando il sensore rileverà un oggetto verrà mostrato a monitor ON e saranno svolti tutti i passaggi dalla change detection, al preprocessing, per concludersi dopo la ricostruzione del modello 3D.

Al termine di questi passaggi il sensore mostrerà ON, se l'oggetto non è stato rimosso, OFF altrimenti e il congegno sarà ancora in modalità di ricezione.

Questi passi sono mostrati nell'esempio della figura 3.14.



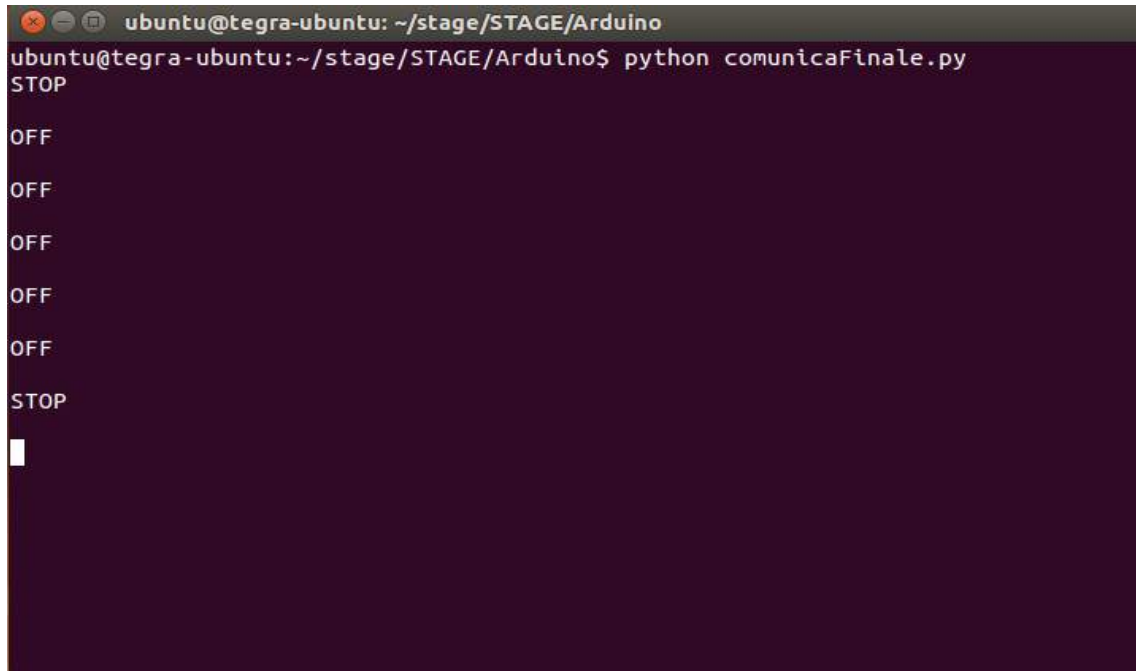
```
ubuntu@tegra-ubuntu: ~/stage/STAGE/Arduino
OFF
OFF
OFF
ON
frame diversi nella camera 0
frame diversi nella camera 1
frame diversi nella camera 0
frame diversi nella camera 1
frame diversi nella camera 0
frame diversi nella camera 1
frame uguali nella camera 0
frame uguali nella camera 1
White Balance
3D reconstruction
rettificaImmagini
Stereo Correspondence
computePointCloud
ON
ON
```

Figura 3.14: Console durante la modalità automatica.

Se si volesse spegnere il sensore di prossimità e bloccare la ricezione, sarà sufficiente premere nuovamente il bottone di accensione.

A monitor sarà visualizzata una scritta STOP, segno che il sensore è stato spento correttamente.

Questi passi sono mostrati nell'esempio della figura 3.15.



```
ubuntu@tegra-ubuntu: ~/stage/STAGE/Arduino
ubuntu@tegra-ubuntu:~/stage/STAGE/Arduino$ python comunicaFinale.py
STOP
OFF
OFF
OFF
OFF
OFF
STOP
█
```

Figura 3.15: Console al momento dello stop.

### 3.2.4 Modalità manuale

Questa opzione permette di acquisire immagini in modo manuale, scattare due fotografie (una per camera) e inviarle alla Jetson TK1 che elaborerà poi il tutto. Per scattare in qualsiasi momento sarà necessario premere il pulsante specifico, denominato come pulsante “B” (spiegazione al capitolo 2.3).

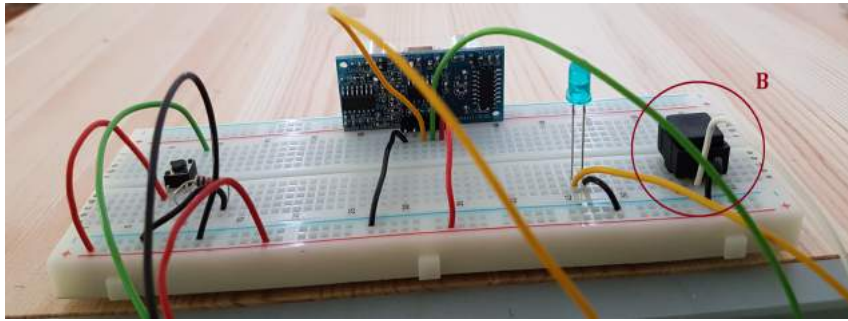


Figura 3.16: Pulsante B sulla breadboard.

Quando il pulsante sarà premuto, Arduino invierà un segnale alla Jetson TK1 per comunicare alle due camere di scattare.

Questa operazione sarà disponibile in qualsiasi momento, a meno che Arduino sia in stand-by, ed è stata pensata come un'opzione aggiuntiva, anche se la sua funzione primaria era quella di meccanismo di “emergenza”.

Questo meccanismo era attivabile nel caso si volesse scattare e il sensore di prossimità non funzionasse o non rilevasse l'oggetto.

Una volta acquisite le due immagini, si passerà al preprocessing, alla ricostruzione del modello 3D e al salvataggio delle immagini in una cartella sulla Jetson con le medesime fasi della modalità automatica.



### Console di controllo

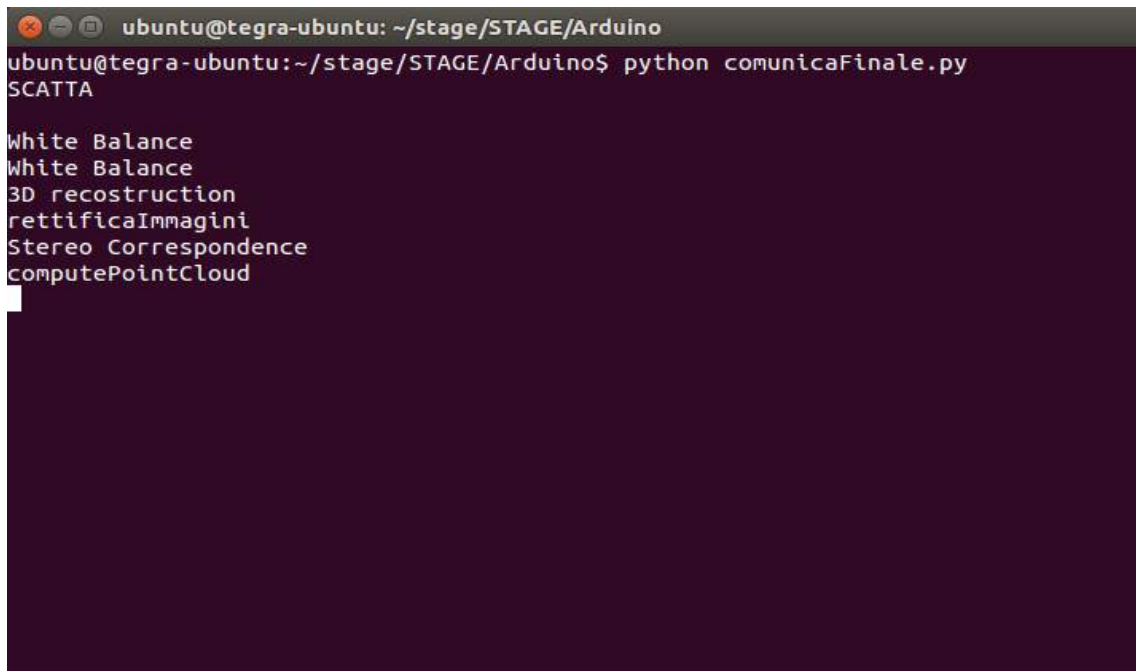
Questo è ciò che accade sul monitor, mentre si utilizza la modalità manuale del dispositivo:

Dopo avere avviato il programma e premuto il pulsante per lo scatto manuale sul monitor sarà mostrata la scritta SCATTA.

In seguito a questo avvenimento saranno svolti tutti i passaggi del preprocessing, per concludersi dopo la ricostruzione del modello 3D.

Al termine di questi passaggi sul monitor non sarà mostrato altro e il programma resterà in attesa di altri avvenimenti.

Questi passi sono mostrati nell'esempio della figura 3.17.



```
ubuntu@tegra-ubuntu: ~/stage/STAGE/Arduino
ubuntu@tegra-ubuntu:~/stage/STAGE/Arduino$ python comunicaFinale.py
SCATTA

White Balance
White Balance
3D reconstruction
rettificaImmagini
Stereo Correspondence
computePointCloud
```

Figura 3.17: Console durante la modalità manuale.



### 3.2.5 Schema elettrico

In seguito nella figura 3.18 viene mostrato la rappresentazione del circuito, creato sulla breadboard, per poter ricostruire la parte della sezione A.

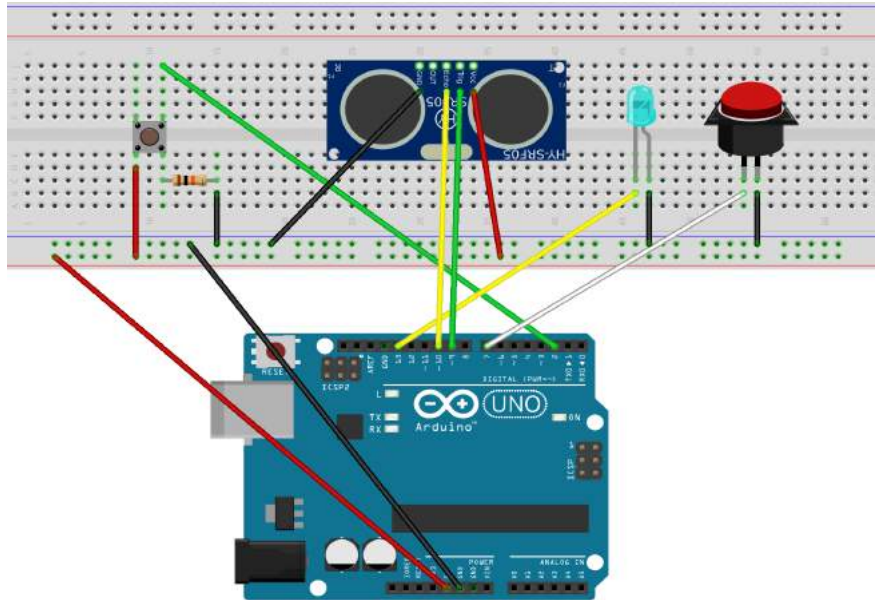


Figura 3.18: Rappresentazione del circuito della sezione A.

Nella figura 3.19 si nota lo schema elettrico della sezione A.

Nella figura 3.20 viene mostrato lo schema elettrico da utilizzare per poter passare dalla breadboard ad un circuito stampato su una scheda.

Come scheda si potrebbe utilizzare una basetta di rame PCB [58].

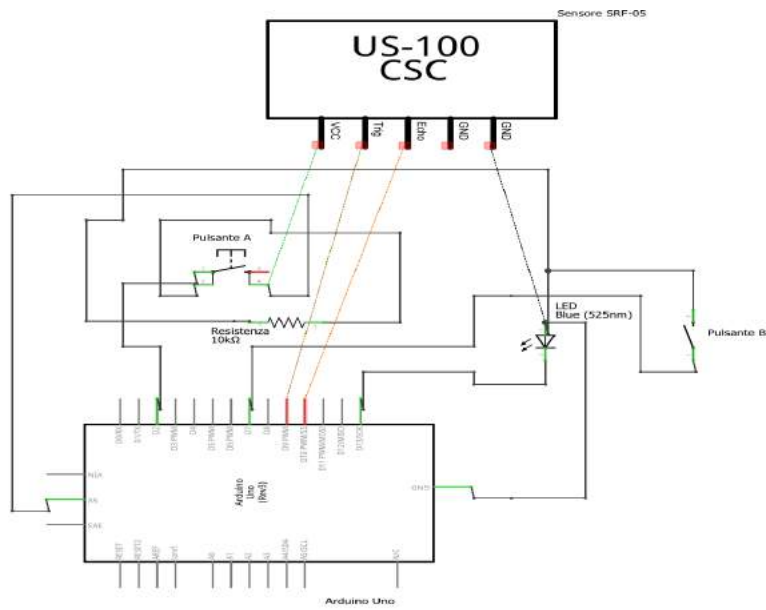


Figura 3.19: Schema elettrico del circuito della sezione A.

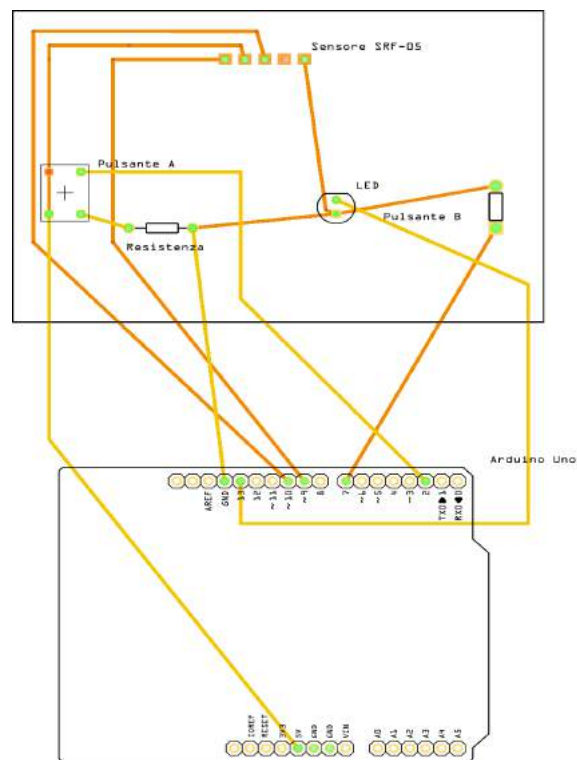


Figura 3.20: Circuito stampato per la sezione A.

### 3.3 Sezione B

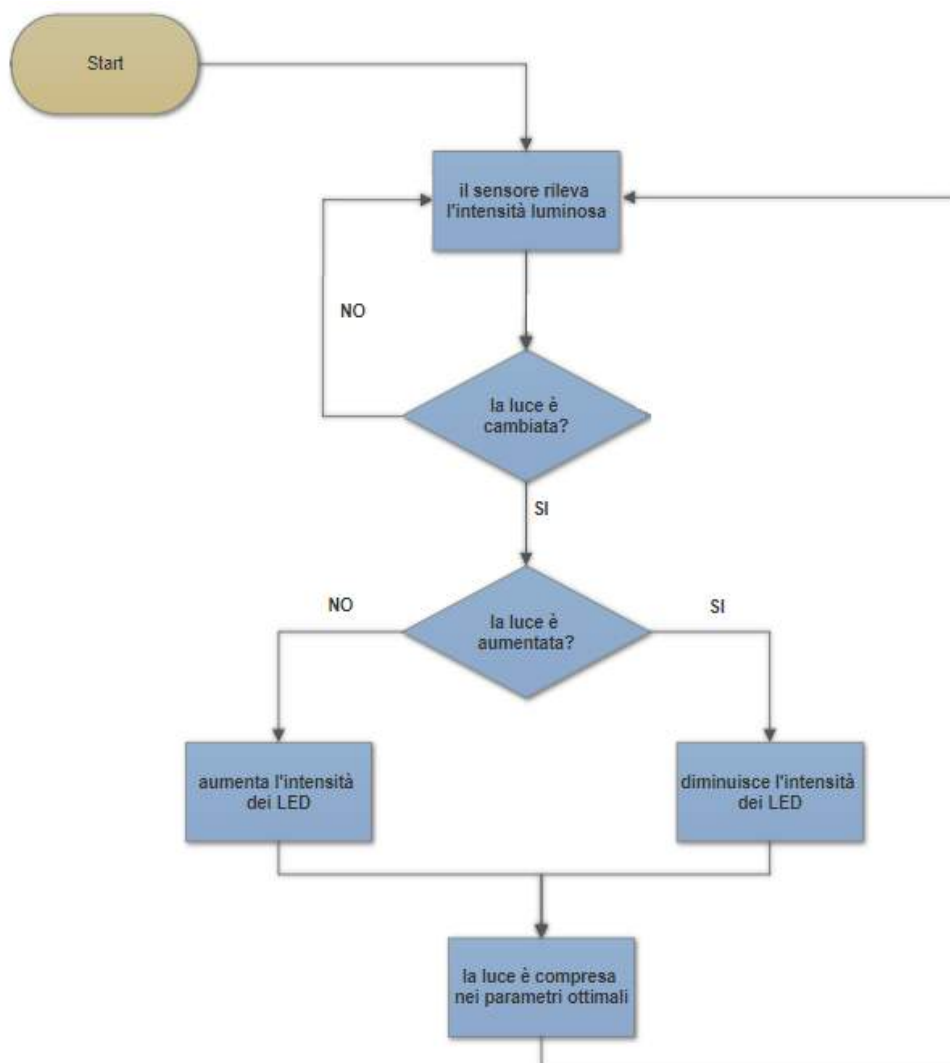


Figura 3.21: Flow chart sezione B.

In questa sezione andrò ad analizzare per passi il funzionamento del dispositivo per quanto riguarda il sensore di luminosità e la gestione della luce all'interno del dispositivo.

### 3.3.1 Costruzione e collegamenti

Come introdotto nel capitolo 2.6.1 la scelta dei led è ricaduta su led RGB programmabili.

Ho scelto di acquistare 5 metri di questi led, poiché studiando la loro potenza e la disposizione che avrebbero assunto nel dispositivo, ho ritenuto fossero sufficienti per un'illuminazione uniforme e intensa.

Per illuminare in modo omogeneo il congegno ho pensato alla disposizione mostrata in figura 3.22.

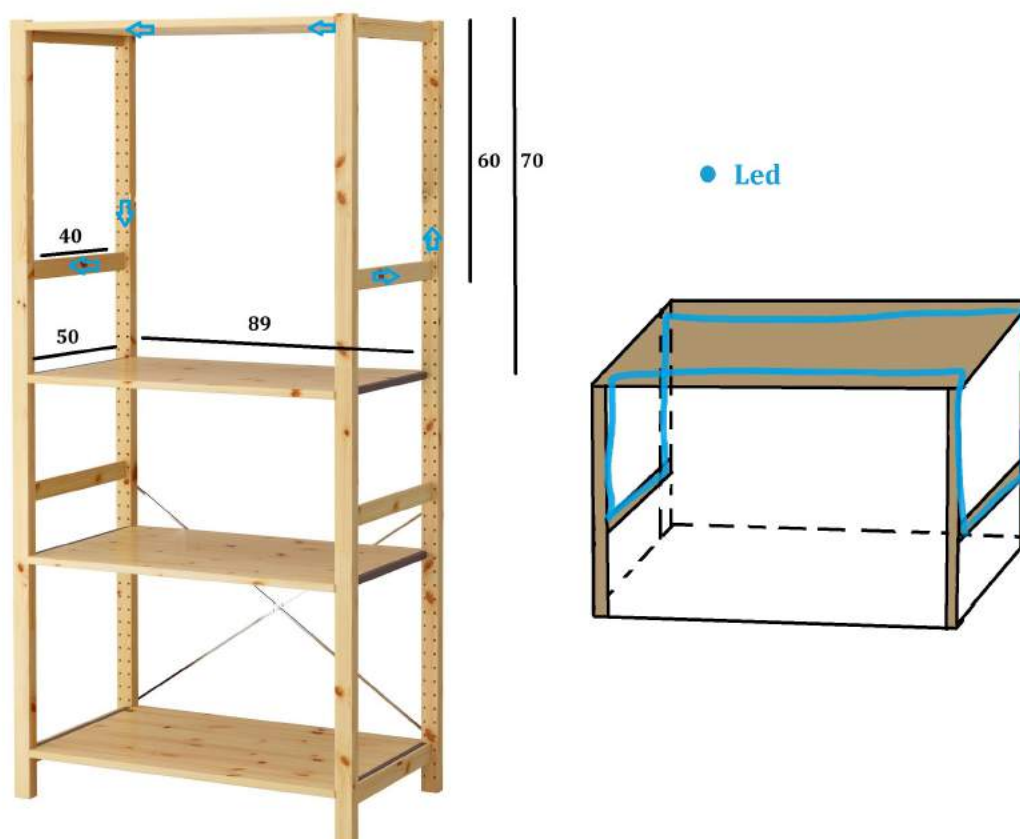


Figura 3.22: Disposizione dei led sull'IVAR.

Sono stati effettuati diversi test, alcuni come i risultati al capitolo 2.6.2, hanno mostrato che fosse necessario collegare all'alimentatore entrambe le estremità della striscia led, con la rispettiva alimentazione (VCC) e messa a terra (GND).

L'Arduino Uno utilizzato per questa sezione era collegato a:

- sensore di luminosità;
- striscia led RGB.

Arduino è stato collegato alle striscia led, seguendo lo schema della figura 3.23.

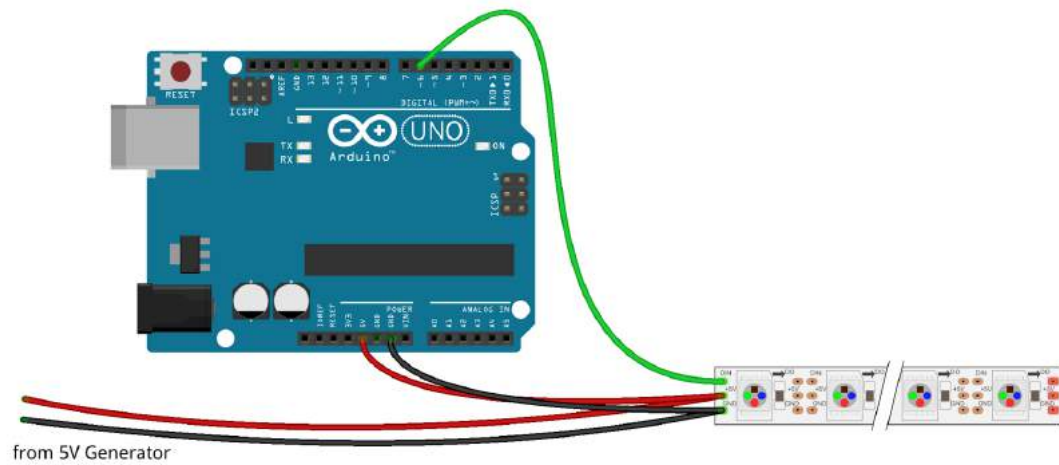


Figura 3.23: Schema collegamento striscia led - Arduino.

Tenendo conto di collegare il cavo per la comunicazione (verde) a un pin digitale PWM [43] di Arduino (io ho utilizzato il pin 6), in modo tale da poter programmare i led.

E' stato necessario collegare sia il sensore che i led al medesimo Arduino così che la scheda potesse ricevere le informazioni e in real-time gestire la luce all'interno del dispositivo.

Il collegamento è stato effettuato come mostrato nella figura 3.24.

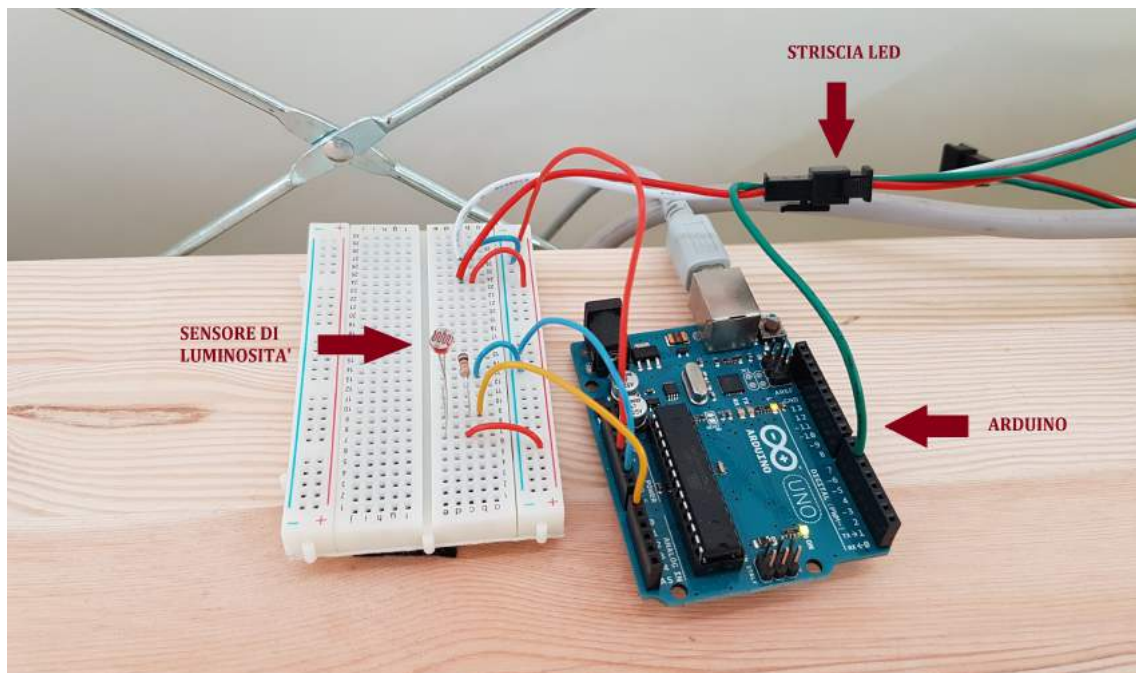


Figura 3.24: Collegamento sensore luminosità - Arduino - striscia led.

### 3.3.2 Rilevazione Luce

I led utilizzati erano programmabili per:

- colore;
- intensità luminosa.

Mi sono soffermato prima sul colore.

#### Colore led

Ho effettuato diversi test, dove si è tenuto conto che queste strisce led RGB in realtà prendono in ordine, come espresso nel capitolo 2.6.2, verde (G) rosso (R) e blu (B).

Insieme al mio collega che si occupava del preprocessing, abbiamo testato varie soluzioni e varietà di colori.

Abbiamo cercato di semplificare il lavoro successivo bilanciando in principio alcune tonalità della luce.

Siamo partiti da una luce bianca (255,255,255), ma abbiamo notato che le strisce led tendevano leggermente al colore blu.

Per queste ragioni dopo varie prove abbiamo diminuito il blu da 255 a 200, mentre il resto dei colori è stato lasciato invariato.

Colore finale:

- Verde: 255;
- Rosso: 255;
- Blu: 200.

Quindi il colore delle strisce led definitivo è **(255,255,200)**, che ho confrontato con alcuni test di intensità luminosa, mostrati al seguente paragrafo.

## Intensità luminosa led

Per rilevare l'intensità luminosa mi sono servito di uno spettrofotometro (spiegazione al capitolo 2.8), grazie al quale riuscivo a rilevare l'intensità luminosa in lux.

Il lux (simbolo lx) è l'unità di misura per l'illuminamento, è relativa alla luce visibile e pertanto dipendente dalle caratteristiche dell'occhio umano.

Dai vincoli iniziali, sapevo di dover mantenere una intensità luminosa che si aggirasse tra i 400 - 500 lx, decidendo se essa fosse adeguata alla nostra esigenza. Perciò utilizzando lo spettrofotometro i1Pro2 della X-Rite e il programma iProfiler sono riuscito, aumentando o diminuendo l'intensità luminosa dei led, ad avere sufficienti dati per arrivare ad alcune conclusioni.

Mostro in figura 3.25 la schermata del programma iProfiler, durante una rilevazione dell'intensità luminosa.

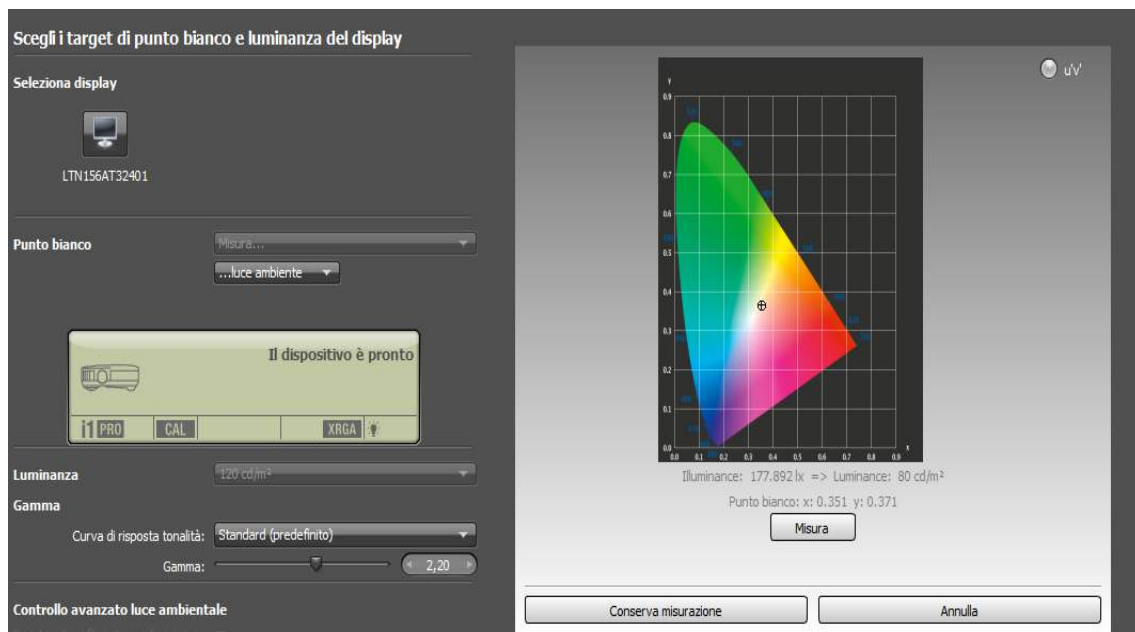


Figura 3.25: Schermata iProfiler durante rilevazione luce ambiente.



E' importante sapere che il sensore di luminosità comunicava il valore di intensità luminosa ad Arduino in una scala da 0 a 1023, dove 0 era il più basso valore e 1023 il massimo della luce ambientale rilevabile.

Bisogna tenere conto che nella stanza dove si sono effettuati i test c'erano alcune fonti per avere una maggiore o minore quantità di luce ambientale come:

- due finestre con tapparella (luce naturale);
- luce della stanza (luce artificiale).

In seguito saranno mostrati i risultati raccolti durante i test per la scelta dell'intensità luminosa con colore dei led.

Mi sono concentrato su:

- 500 lx;
- 400 lx.

di cui farò un confronto al termine del test.

Il colore dei led utilizzati, come espresso in precedenza è (255,255,200)

Modalità	Led	Intensità	Lux
luci spente/tapparelle aperte	spenti	0	500
luci spente/tapparelle aperte	accesi	255	1100

Non necessario tener conto del sensore di luminosità, lux già sufficienti.

Modalità	Led	Intensità	Lux
luci accese/tapparelle chiuse	spenti	0	300
luci accese/tapparelle chiuse	accesi	65	500
luci accese/tapparelle chiuse	accesi	35	400

Sensore di luminosità (a 500 lx): 870

Sensore di luminosità (a 400 lx): 850

Da questo momento in poi nei test le luci artificiali della stanza saranno sempre spente.

Modalità	Led	Intensità	Lux
tapparelle chiuse	spenti	0	113
tapparelle chiuse	accesi	255	740
tapparelle chiuse	accesi	235	650
tapparelle chiuse	accesi	215	630
tapparelle chiuse	accesi	175	500
tapparelle chiuse	accesi	140	440
tapparelle chiuse	accesi	120	400

Sensore di luminosità (a 500 lx): 880

Sensore di luminosità (a 400 lx): 855

Modalità	Led	Intensità	Lux
tapparelle chiusa 1 e aperta 2 (a metà)	spenti	0	184
tapparelle chiusa 1 e aperta 2 (a metà)	accesi	165	630
tapparelle chiusa 1 e aperta 2 (a metà)	accesi	125	500
tapparelle chiusa 1 e aperta 2 (a metà)	accesi	90	400

Sensore di luminosità (a 500 lx): 875

Sensore di luminosità (a 400 lx): 860

Modalità	Led	Intensità	Lux
tapparelle chiusa 1 e aperta 2	spenti	0	265
tapparelle chiusa 1 e aperta 2	accesi	100	500
tapparelle chiusa 1 e aperta 2	accesi	90	470
tapparelle chiusa 1 e aperta 2	accesi	65	400

Sensore di luminosità (a 500 lx): 875

Sensore di luminosità (a 400 lx): 860

Modalità	Led	Intensità	Lux
tapparelle aperta 1 e chiusa 2	spenti	0	320
tapparelle aperta 1 e aperta 2	accesi	85	500
tapparelle chiusa 1 e aperta 2	accesi	55	400

Sensore di luminosità (a 500 lx): 875

Sensore di luminosità (a 400 lx): 860

Modalità	Led	Intensità	Lux
tapparelle chiusa 1 e aperta 2	spenti	0	265
tapparelle chiusa 1 e aperta 2	accesi	100	500
tapparelle chiusa 1 e aperta 2	accesi	90	470
tapparelle chiusa 1 e aperta 2	accesi	65	400

Sensore di luminosità (a 500 lx): 875

Sensore di luminosità (a 400 lx): 860

Alla fine dei test e dei calcoli abbiamo constatato che:

- il range utilizzato 870:880 produce 500 lx;
- il range utilizzato 860:870 produce 400 lx.

Dopo una serie di consultazioni con i Tutor, ho deciso che 500 lx era un valore troppo alto e ho utilizzato come luminosità da mantenere all'interno del dispositivo **400 lx**, con il relativo range del sensore di luminosità **860:870**.

### 3.3.3 Avvio e modalità

Una volta accesa la Jetson l'Arduino relativo alla gestione delle luci sarà automaticamente collegato e attivato (se ciò non avvenisse consultare il 3.4).

Ovviamente su questo Arduino sarà necessario caricare la prima volta che lo si utilizza (poi sarà sempre salvato e quindi utilizzabile senza alcun caricamento) il programma:

gestioneLuci.ino

Una volta fatto ciò, ad ogni avvio saranno attivati in automatico i led, che si auto-regoleranno in real-time.

Il programma segue questi passi in una continua ripetizione:

1. controlla il sensore di luminosità;
2. se il valore è compreso tra 860:870 o l'intensità è uguale a 0, si lascia invariata l'intensità luminosa;
3. se il valore è minore di 855 si aumenta l'intensità luminosa (fino ad un massimo di 255);
4. se il valore è maggiore di 875 si diminuisce l'intensità luminosa (fino ad un minimo di 0).

Il range originale era di 860:870 ma è risultato necessario per gli aumenti e le diminuzioni dell'intensità luminosa, lasciare una piccola soglia di errore del sensore (stimata a  $\pm 5$ ) per assicurare un funzionamento fluido e corretto.

In questo modo utilizzando i range del test, siamo sicuri che qualsiasi condizione di luce sia presente nel dispositivo e nella stanza che lo circonda, esso al suo interno avrà sempre un ambiente controllato a 400 lx che permetterà di far scattare fotografie sempre in condizioni di luce ottimale.

Se la luce nella stanza sarà ritenuta sufficiente i led resteranno spenti, ma sempre pronti ad azionarsi qualora queste condizioni cambiassero.

Per non ricevere delle interferenze alcuni led sono stati singolarmente spenti:

- i led da 1 a 16;
- i led da 277 a 300.

Cioè i primi led della parte iniziale e finale, poichè alla fine del progetto sono stati ritenuti non necessari.

In figura 3.26 è mostrato il dispositivo in presenza di luce naturale, perciò con i led spenti, mentre nella figura 3.27 si nota il dispositivo in una situazione di scarsa luce in cui i led si accendono per riportare le condizioni di luce nel macchinario nella situazione ottimale.



Figura 3.26: Dispositivo con luce naturale sufficiente.



Figura 3.27: Dispositivo con i led attivi.

### 3.3.4 Schema elettrico

In seguito nella figura 3.28 viene mostrato la rappresentazione del circuito, creato sulla breadboard, per poter ricostruire la parte della sezione B.

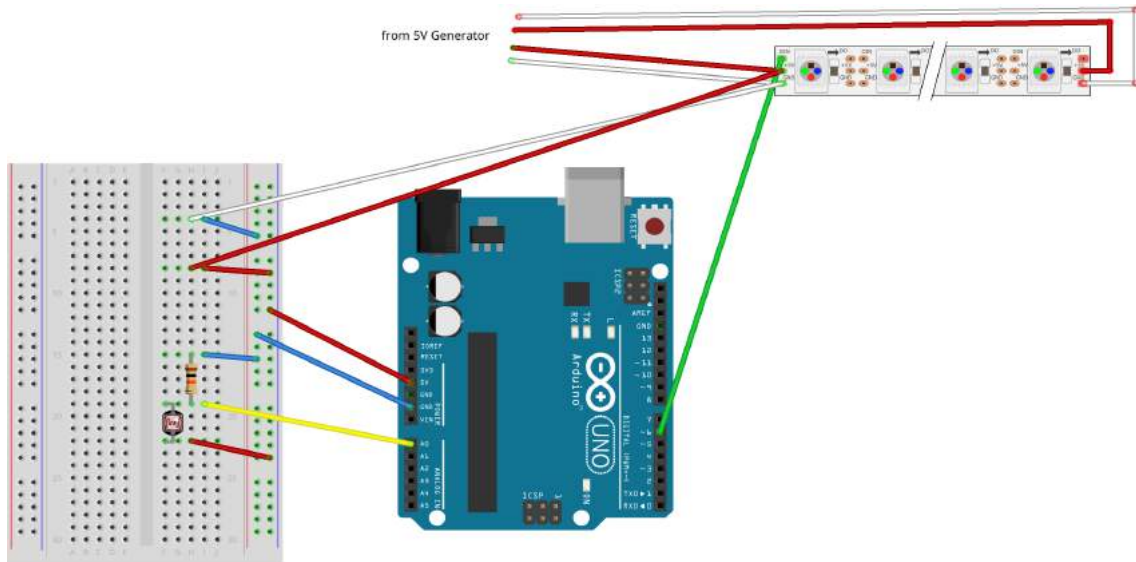


Figura 3.28: Rappresentazione del circuito della sezione B.

Nella figura 3.29 si nota lo schema elettrico della sezione B.

Nella figura 3.30 viene mostrato lo schema elettrico da utilizzare per poter passare dalla breadboard ad un circuito stampato su una scheda.

Come scheda si potrebbe utilizzare una basetta di rame PCB [58].

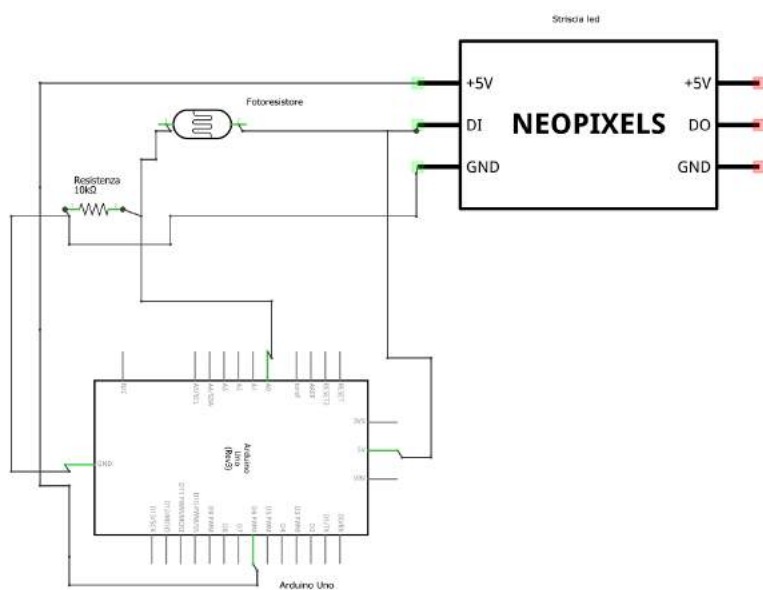


Figura 3.29: Schema elettrico del circuito della sezione B.

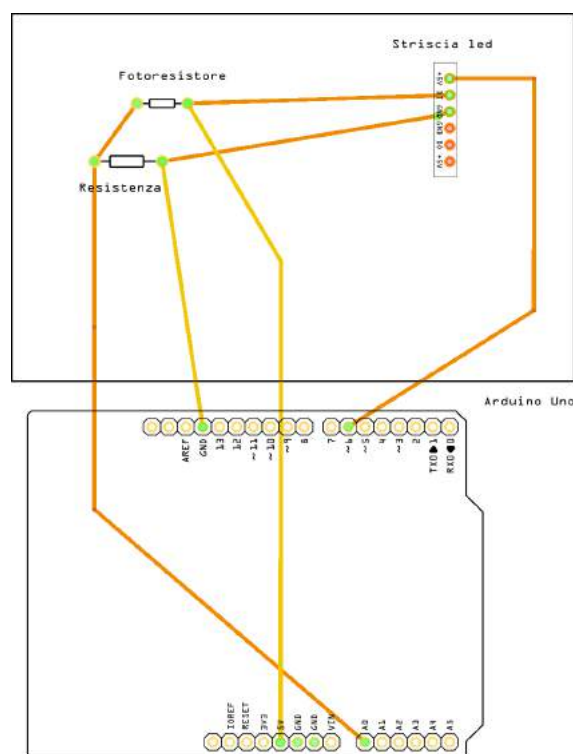


Figura 3.30: Circuito stampato per la sezione B.

## 3.4 Malfunzionamenti collegamenti Arduino e Sincronizzazione

In questo paragrafo verrà spiegato come collegare Arduino e sincronizzarlo con la Jetson TK1 in caso di malfunzionamenti.

Questa eventualità è molto rara e non dovrebbe mai verificarsi, ma di seguito elencherò una serie di passaggi per risolvere eventuali errori o problemi sui collegamenti di Arduino.

Se una volta accesa la Jetson TK1, si riscontrassero uno di questi problemi:

- errore di riconoscimento di Arduino che gestisce il sensore di prossimità;
- errore di riconoscimento di Arduino che gestisce il sensore di luminosità;
- errore al momento dell'esecuzione di `comunicaFinale.py`.

Aprire l'IDE di Arduino, andare in "Strumenti>Porta seriale" e controllare che:

- l'Arduino che controlla il sensore di prossimità sia collegato alla porta `"/dev/ttyACM0"`;
- l'Arduino che controlla il sensore di luminosità sia collegato alla porta `"/dev/ttyACM1"`.

Per rapidità di selezione aprire l'IDE [\[42\]](#) di Arduino direttamente dal programma:

- `"acquisizione.ino"` per il sensore di prossimità;
- `"gestioneLuci.ino"` per il sensore di luminosità.

Inoltre controllare per una maggiore sicurezza che in "Strumenti>Tipo di Arduino" sia selezionato:

- Arduino Uno.

I programmi `"acquisizione.ino"` e `"gestioneLuci.ino"` sono consultabili al capitolo 4.3

Seguiti tutti questi passaggi gli errori dovrebbero essere stati risolti e i due Arduino pronti all'uso e sincronizzati in via seriale con la Jetson TK1.



## Capitolo 4

# Fase finale e aggiornamenti

Alla fine del progetto, ho iniziato a pensare a degli aggiornamenti possibili per perfezionare la resa del dispositivo e ne ho trovati due di maggior rilevanza:

1. Rendere funzionante il dispositivo con la Jetson TX1;
2. Pensare a dei miglioramenti per le fotocamere.

### 4.1 Jetson TX1

La Jetson TX1 [8] (figura 4.1) sempre creata dall’NVIDIA è una piccola scheda basata su un sistema Linux-on-module, destinato a richiedere applicazioni embedded nel Visual computing.



Figura 4.1: Jetson TX1.

Con Jetson TX1 Developer Kit [59], si utilizzano molti dei software che utilizzava la Jetson TK1 ma ottimizzati e aggiornati alle ultime versioni.

Alcuni dei software utilizzati contenuti sempre nel pacchetto Jetpack 3.1 [32] sono:

- Linux For Tegra R28.1;
- CUDA Toolkit 8;
- cuDNN v6.0;
- OpenGL 4.4;
- VisionWorks 1.6.

Concepito dagli sviluppatori interessati a Vision computing, l'impronta simile a una carta di credito della Jetson TX1 e un basso consumo energetico portano a pensare che questa scheda sia destinata all'implementazione di sistemi on board embedded con dimensioni, peso e potenza limitati.

Jetson fornisce un'efficienza superiore mantenendo un ambiente favorevole agli sviluppatori per prototipi agili.

Il modulo Jetson TX1 consente agli sviluppatori di distribuire Tegra [25] in applicazioni embedded che vanno dalla navigazione autonoma alle analisi basate sull'apprendimento.

Andrò ora a fare un confronto tra alcune specifiche tra la Jetson TK1 (analizzata nel capitolo 2.1) e la Jetson TX1.

Le due schede sono mostrate in figura 4.2



Figura 4.2: Jetson TX1 - Jetson TK1.

Caratteristiche principali [60]:

	<b>TX1</b>	<b>TK1</b>
CPU	quad-core ARM Cortex-A57	quad-core ARM Cortex-A15
GPU	256-core Maxwell	NVIDIA Kepler "GK20a"
Memory	4GB	2GB
Storage	16GB eMMC 5.1	16GB eMMC 4.51
USB	USB 3.0 + USB 2.0	USB 3.0 + USB 2.0
Power	19V DC	12V DC

Ho cambiato la scheda Jetson TK1 con la scheda TX1, installando esattamente gli stessi software con la versione aggiornata e modificando alcune righe del codice `comunicaFinale.py` (spiegazione capitolo 4.3) ho testato il dispositivo.

Tenendo in considerazione che il tempo con la Jetson TK1 era di quasi un minuto, ho notato con la Jetson TX1 una diminuzione di qualche decina di secondi per elaborare le immagini, velocizzando tutto il processo.

Come ultimo aggiornamento in futuro si potrebbe aumentare ulteriormente la potenza del tutto utilizzando una Jetson TX2 [61], da poco uscita sul mercato, ma di cui al momento non disponevo.

## 4.2 Miglioramenti fotocamere

Ho utilizzato per la Jetson TK1 delle fotocamere USB.

Ma per la Jetson TX1 è possibile utilizzare delle fotocamere a bus CSI [22], con maggiori megapixel (MP) e la possibilità di cambiare l'obiettivo e le lenti per ogni necessità.

Oltretutto queste camere consentivano di ottenere immagini RAW [12], ottime per ottenere un preprocessing completo e dettagliato.

Il kit Jetson TX1 disponeva del modulo camera fornito nel Developer kit, come mostrato in figura 4.3.

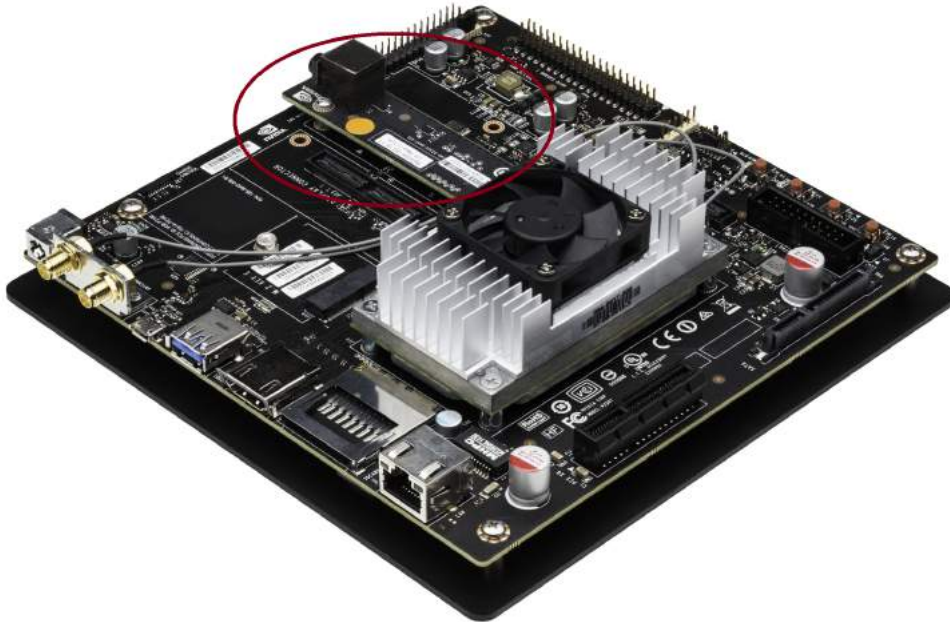


Figura 4.3: Modulo camera Jetson TX1.

Ma per riuscire ad utilizzare 6 camere per Raspberry Pi [62] del tipo OV5647 [63] (figura 4.4 e 4.5) o simili è necessario servirsi della scheda J20 dell'Auvideo [64] (figura 4.6 e 4.7) inserendola al posto del modulo camera sulla Jetson TX1.



Figura 4.4: Camera OV5647 con lente intercambiabile

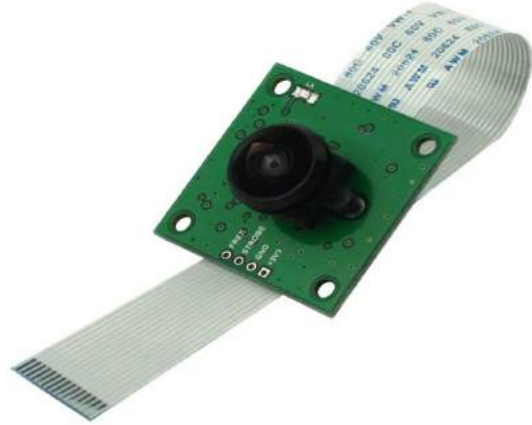


Figura 4.5: Camera OV5647 con bus CSI.

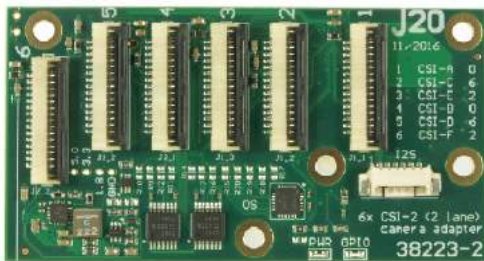


Figura 4.6: Parte anteriore dell'Auvideo J20.

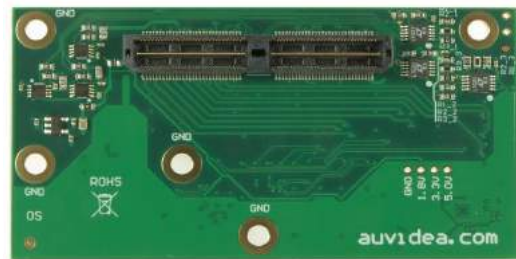


Figura 4.7: Parte posteriore dell'Auvideo J20.



Nella figura 4.8 è mostrato il prototipo finale.

Esso è composta da sei camere OV5647 collegate alla Jetson TX1 e supportate dalla scheda Auvideo J20.

Per poterla collegare al dispositivo è necessario acquistare dei bus di lunghezza maggior per poterli posizionare in modo corretto.

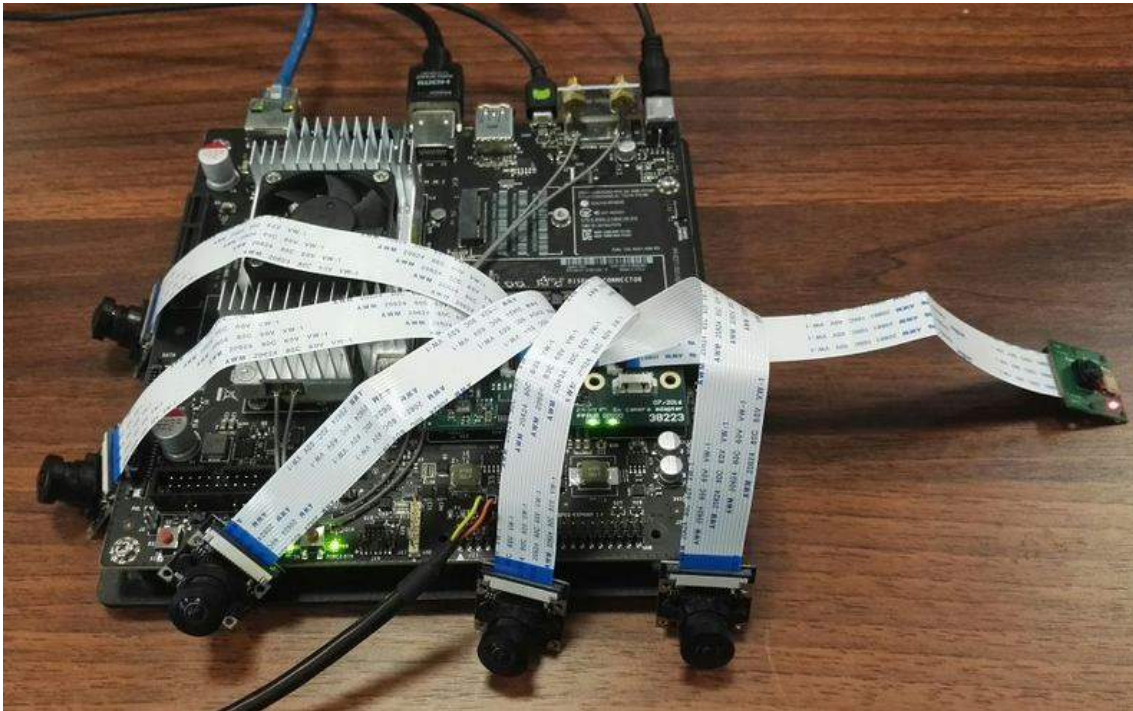


Figura 4.8: Prototipo Jetson TX1 con 6 camere.

Il materiale è stato acquistato e conservato perchè questi miglioramenti sono stati pensanti per un aggiornamento futuro.

Allego, se si volesse procedere con questo miglioramento, il sito dell'auvidea e il manuale della J20.



Figura 4.9: Logo Auvidea.

Sito auvidea con spiegazione:

<https://auvidea.com/j20/>

Manuale:

<https://auvidea.com/manuals/>

### 4.3 Diagramma di flusso aggiornamenti

Nella figura 4.10 è mostrato il diagramma di flusso completo, nel caso si decidesse di aggiornare il dispositivo.

Questo flow chart comprende la parte delle immagini RAW, di demosaicing e di color mapping.

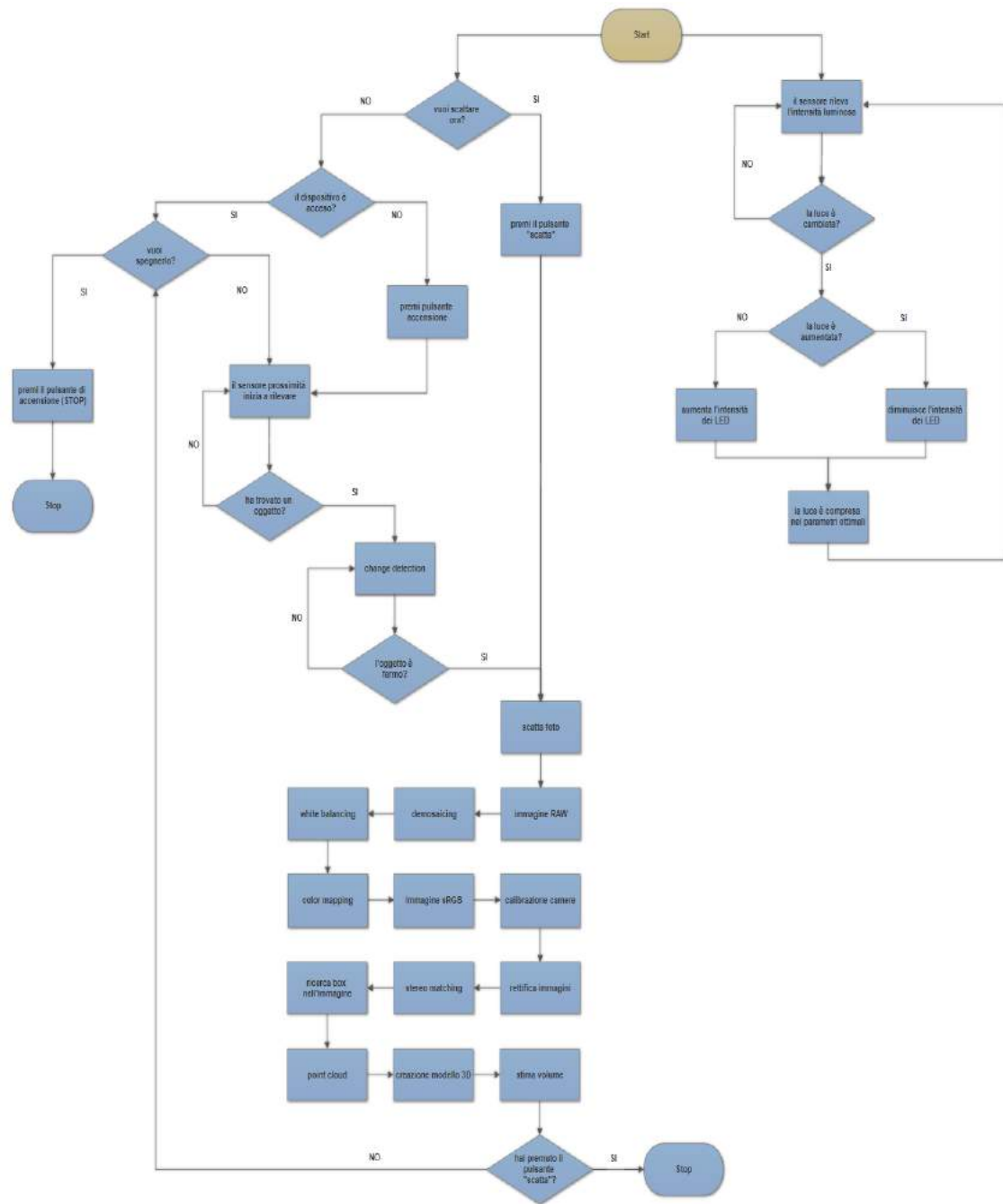


Figura 4.10: Flow chart completo prototipo finale.



# Conclusioni

Al termine del progetto, sono state stilate le seguenti conclusioni.

L'idea iniziale era ricreare un ambiente controllato in cui acquisire delle immagini e l'obiettivo è stato ampiamente raggiunto.

Ciò è stato possibile grazie alla gestione degli spazi in cui rilevare gli oggetti e alla gestione delle luci.

Gli spazi sono stati verificati mediante un sensore di prossimità, controllato da un Arduino Uno.

La gestione della luce è stata controllata mediante un sensore di luminosità e una striscia Led RGB gestiti entrambi da un altro Arduino Uno.

Perciò si aveva un dispositivo che in contemporanea gestiva, mediante i due Arduino, la ricezione di oggetti automatica o manuale, l'acquisizione delle immagini e la regolazione costante dell'intensità luminosa.

La scelta della Jetson TK1 come computer per elaborare le immagini si è rivelata corretta e l'aggiornamento con la Jetson TX1 ha migliorato ulteriormente il prototipo, diminuendo i tempi di esecuzione.

Infatti tutto il processo del dispositivo si svolge in poche decine di secondi, non andando mai oltre al minuto.

La fase iniziale si è rivelata molto importante, poichè avendo scelto i giusti componenti per il nostro progetto, sono riuscito a portare a termine la mia parte nei tempi e nei termini concordati.

E' stato fondamentale lavorare con miei colleghi come un vero gruppo, poichè riuscendo a collaborare abbiamo semplificato notevolmente il lavoro, diminuendo le difficoltà.

I miei colleghi hanno anch'essi portato a compimento gli obiettivi della loro parte del progetto, concludendo la parte di preprocessing e ricostruzione del modello 3D. Alla fine unendo le tre parti siamo riusciti a creare un dispositivo funzionante, che elaborasse le immagini in poco tempo e in modo ottimale.

Perciò al termine di questa esperienza tutti gli obiettivi stilati nella fase di preparazione sono stati portati a termine con successo.

# Codice

In fondo alla pagina sarà inserito in link alla repository dove sarà presente il codice dei seguenti programmi:

- `comunicaFinale.py`;
- `acquisizione.ino`;
- `gestioneLuci.ino`.

Inoltre nella repository si troverà anche il codice dei seguenti programmi scritti dai miei due colleghi:

- `E.py` (per la parte di preprocessing);
- `pointCloud.py` (per la parte di ricostruzione del modello 3D);
- `functions.py` (che viene richiamato da `pointCloud.py`).

Saranno presenti due cartelle con i codici adattati per la:

- Jetson TK1;
- Jetson TX1.

Le differenze si trovano nel programma `comunicaFinale.py`, i restanti programmi non hanno subito variazioni.

Si sono create due cartelle semplicemente per comodità di download dei file.

**Link alla repository:**

[https://github.com/MirkoRima/ProgettoDispositivo\\_Tesi](https://github.com/MirkoRima/ProgettoDispositivo_Tesi)

# Bibliografia

- [1] Bianco Simone. «Color Correction Algorithms for Digital Cameras». Thesis. Università degli Studi di Milano-Bicocca, 2008/2009. URL: [https://boa.unimib.it/retrieve/handle/10281/7819/9245/phd\\_unimib\\_033211.pdf](https://boa.unimib.it/retrieve/handle/10281/7819/9245/phd_unimib_033211.pdf).
- [2] Wikipedia. *Modello 3D* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Modello\\_3D&oldid=88251328](http://it.wikipedia.org/w/index.php?title=Modello_3D&oldid=88251328).
- [3] *Microcontrollore*. [Online; in data 20-settembre-2017 ]. 2017. URL: <http://www.elemania.altervista.org/pic/pic1.html>.
- [4] *Sensore ad ultrasuoni srf05*. [Online; in data 20-settembre-2017 ]. 2017. URL: <http://www.mauroalfieri.it/elettronica/sensore-ad-ultrasuoni-srf05.html>.
- [5] Gianni Forcolini. *Illuminazione LED*. Vol. 2<sup>a</sup> edizione. Hoepli, 2011.
- [6] Wikipedia. *Fotoresistenza* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://it.wikipedia.org/w/index.php?title=Fotoresistenza&oldid=87339775>.
- [7] Wikipedia. *Alimentatore elettrico* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Alimentatore\\_elettrico&oldid=90785123](http://it.wikipedia.org/w/index.php?title=Alimentatore_elettrico&oldid=90785123).
- [8] *Jetson TX1*. [Online; in data 20-settembre-2017 ]. 2017. URL: <https://devblogs.nvidia.com/parallelforall/nvidia-jetson-tx1-supercomputer-on-module-drives-next-wave-of-autonomous-machines/>.
- [9] Wikipedia. *Lux* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2016. URL: <http://it.wikipedia.org/w/index.php?title=Lux&oldid=82978786>.
- [10] Wikipedia. *LED strip light* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=LED\\_strip\\_light&oldid=795058679](https://en.wikipedia.org/w/index.php?title=LED_strip_light&oldid=795058679).

- [11] Wikipedia. *Demosaicizzazione* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://it.wikipedia.org/w/index.php?title=Demosaicizzazione&oldid=86994652>.
- [12] Wikipedia. *Raw image format* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Raw\\_image\\_format&oldid=799890986](https://en.wikipedia.org/w/index.php?title=Raw_image_format&oldid=799890986).
- [13] *Color Constancy*. [Online; in data 20-settembre-2017 ]. 2017. URL: [http://colorconstancy.com/?page\\_id=9](http://colorconstancy.com/?page_id=9).
- [14] Wikipedia. *Bilanciamento del colore* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Bilanciamento\\_del\\_colore&oldid=88903882](http://it.wikipedia.org/w/index.php?title=Bilanciamento_del_colore&oldid=88903882).
- [15] Wikipedia. *Color mapping* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Color\\_mapping&oldid=794925038](https://en.wikipedia.org/w/index.php?title=Color_mapping&oldid=794925038).
- [16] Samuele Salti e Luigi Di Stefano. *Modello della Telecamera, Calibrazione e Rettificazione*. Report. Computer Vision Laboratory, 2010/2011. URL: [www.t3lab.it/wp-content/.../02%20-%20Camera\\_Model-Calibration-Rectification.pdf](http://www.t3lab.it/wp-content/.../02%20-%20Camera_Model-Calibration-Rectification.pdf).
- [17] *Stereo matching*. [Online; in data 21-settembre-2017 ]. 1997. URL: [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/OWENS/LECT11/node5.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT11/node5.html).
- [18] Wikipedia. *Nuvola di punti* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Nuvola\\_di\\_punti&oldid=87484038](http://it.wikipedia.org/w/index.php?title=Nuvola_di_punti&oldid=87484038).
- [19] Donà Marco. «Stima di modelli tridimensionali da immagini bidimensionali». Thesis. Università degli Studi di Padova, 2012/2013. URL: [http://tesi.cab.unipd.it/40199/1/Stima\\_di\\_modelli\\_tridimensionali\\_da\\_immagini\\_bidimensionali.pdf](http://tesi.cab.unipd.it/40199/1/Stima_di_modelli_tridimensionali_da_immagini_bidimensionali.pdf).
- [20] *Arduino Uno*. [Online; in data 19-settembre-2017 ]. 2017. URL: <https://store.arduino.cc/usa/arduino-uno-rev3>.
- [21] *Jetson TK1*. [Online; in data 20-settembre-2017 ]. 2014. URL: <https://hardware.hdblog.it/2014/03/26/nvidia-jetson-tk1-il-primo-supercomputer-mobile-per-sistemi-embedded/>.
- [22] Wikipedia. *Camera Serial Interface* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Camera\\_Serial\\_Interface&oldid=801438807](https://en.wikipedia.org/w/index.php?title=Camera_Serial_Interface&oldid=801438807).

- [23] *Resistenza*. [Online; in data 20-settembre-2017 ]. 2015. URL: <http://www.elettronicaincorso.it/definizione-resistenza-elettrica.html>.
- [24] Autori vari. *NVIDIA Jetson TK1 developer kit*. A cura di NVIDIA. Technical manual. 2017. URL: <http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html>.
- [25] V. P. Nikolskiy, V. V. Stegailov e V. S. Vecher. «Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics». In: *2016 International Conference on High Performance Computing Simulation (HPCS)*. 2016, pp. 682–689. DOI: [10.1109/HPCSim.2016.7568401](https://doi.org/10.1109/HPCSim.2016.7568401).
- [26] Wikipedia. *OpenGL* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://it.wikipedia.org/w/index.php?title=OpenGL&oldid=90881450>.
- [27] Autori vari. *CUDA C Programming Guide*. A cura di Nvidia. Technical manual. Ver. 4.2. 2012. URL: [developer.download.nvidia.com/compute/.../C/doc/CUDA\\_C\\_Programming\\_Guide.pdf](http://developer.download.nvidia.com/compute/.../C/doc/CUDA_C_Programming_Guide.pdf).
- [28] *VisionWorks*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://developer.nvidia.com/embedded/>.
- [29] Wikipedia. *Computer vision* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Computer\\_vision&oldid=801426556](https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=801426556).
- [30] Bucci Giacomo. *Architettura ARM (Advanced RISC Machine)*. Report. Università degli Studi di Firenze, 2011. URL: [www.dsi.unifi.it/~bucci/Teaching/Magistrale/Arm.pdf](http://www.dsi.unifi.it/~bucci/Teaching/Magistrale/Arm.pdf).
- [31] Wikipedia. *Image processing* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Image\\_processing&oldid=792699223](https://en.wikipedia.org/w/index.php?title=Image_processing&oldid=792699223).
- [32] *JetPack*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://developer.nvidia.com/embedded/jetpack>.
- [33] Wikipedia. *Artificial intelligence* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://en.wikipedia.org/w/index.php?title=Artificial\\_intelligence&oldid=801526880](https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=801526880).
- [34] Gary Bradski e Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008. URL: [www-cs.ccny.cuny.edu/~wolberg/capstone/opencv/LearningOpenCV.pdf](http://www-cs.ccny.cuny.edu/~wolberg/capstone/opencv/LearningOpenCV.pdf).
- [35] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006. URL: [csc.ucdavis.edu/~chaos/courses/nlp/Software/NumPyBook.pdf](http://csc.ucdavis.edu/~chaos/courses/nlp/Software/NumPyBook.pdf).

- [36] Wikipedia. *SciPy* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://it.wikipedia.org/w/index.php?title=SciPy&oldid=87398134>.
- [37] *Matplotlib*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://matplotlib.org/>.
- [38] *CSV*. [Online; in data 20-settembre-2017 ]. 2013. URL: <https://docs.python.org/2/library/csv.html>.
- [39] *time*. [Online; in data 21-settembre-2017 ]. 2017. URL: [https://www.tutorialspoint.com/python/python\\_date\\_time.htm](https://www.tutorialspoint.com/python/python_date_time.htm).
- [40] C Liechti. *PySerial*. Technical report. 2001-2016. URL: [readthedocs.org/projects/pyserial/downloads/pdf/latest/](http://readthedocs.org/projects/pyserial/downloads/pdf/latest/).
- [41] Wikipedia. *Arduino (hardware)* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Arduino\\_\(hardware\)&oldid=91034729](http://it.wikipedia.org/w/index.php?title=Arduino_(hardware)&oldid=91034729).
- [42] Wikipedia. *Arduino (software)* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Arduino\\_\(software\)&oldid=91277689](http://it.wikipedia.org/w/index.php?title=Arduino_(software)&oldid=91277689).
- [43] *PWM*. [Online; in data 20-settembre-2017 ]. 2017. URL: <https://www.arduino.cc/en/Tutorial/PWM>.
- [44] *Adafruit NeoPixel (Arduino Library Use)*. [Online; in data 21-settembre-2017 ]. 2016. URL: <https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library>.
- [45] *Pushbutton*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://www.arduino.cc/en/Tutorial/Pushbutton>.
- [46] *Sensore di distanza ad Ultrasuoni SRF05*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://www.robot-italy.com/it/low-cost-ultrasonic-range-finder-1.html>.
- [47] *SMD LED*. [Online; in data 21-settembre-2017 ]. 2014. URL: <https://www.illuminazione-a-led.eu/smd-led-2/>.
- [48] *Gradi di protezione IP*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://www.oppo.it/normative/protezione\\_ip.htm](http://www.oppo.it/normative/protezione_ip.htm).
- [49] Autori vari. *WS2812B (Intelligent control LED integrated light source)*. A cura di Worldsemi. Technical manual. 2017. URL: <http://www.seeedstudio.com/document/pdf/WS2812B%20Datasheet.pdf>.
- [50] Wikipedia. *Surface mount technology* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Surface\\_mount\\_technology&oldid=86349516](http://it.wikipedia.org/w/index.php?title=Surface_mount_technology&oldid=86349516).

- [51] Wikipedia. *Spettrofotometria* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://it.wikipedia.org/w/index.php?title=Spettrofotometria&oldid=90794958>.
- [52] Autori vari. *i1Pro User Manual*. A cura di x rite. Technical manual. 2017. URL: [my.xrite.com/documents/apps/public/Manuals/E02UV-QSG\\_i1Pro\\_UVcut\\_User\\_Manual.pdf](http://my.xrite.com/documents/apps/public/Manuals/E02UV-QSG_i1Pro_UVcut_User_Manual.pdf).
- [53] In: *X-Rite annuncia le nuove soluzioni i1Pro 2 per la gestione professionale* (2012). A cura di Expostampa. URL: <http://expostampa.it/novita/x-rite-annuncia-le-nuove-soluzioni-i1pro-2-per-la-gestione-professionale/>.
- [54] *X-Rite i1Profiler*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://www.boscarol.com/blog/?page\\_id=18649](http://www.boscarol.com/blog/?page_id=18649).
- [55] Wikipedia. *Breadboard* — *Wikipedia, The Free Encyclopedia*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Breadboard&oldid=791179452>.
- [56] *c525 Specifications*. [Online; in data 19-settembre-2017 ]. 2017. URL: [http://support.logitech.com/en\\_us/product/hd-webcam-c525/specs](http://support.logitech.com/en_us/product/hd-webcam-c525/specs).
- [57] Schettini Raimondo. *Identificazione di oggetti in moto (Change Detection)*. Report. Università degli Studi di Milano-Bicocca, 2016/2017.
- [58] Wikipedia. *Circuito stampato* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Circuito\\_stampato&oldid=87482802](http://it.wikipedia.org/w/index.php?title=Circuito_stampato&oldid=87482802).
- [59] Autori vari. *Unleash Your Potential with the Jetson TX1 Developer Kit*. A cura di Nvidia. Technical manual. 2017. URL: <https://developer.nvidia.com/embedded/buy/jetson-tx1-devkit>.
- [60] *Developer Kits Key Features*. [Online; in data 21-settembre-2017 ]. 2017. URL: <http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html>.
- [61] *NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge*. [Online; in data 21-settembre-2017 ]. 2017. URL: <https://devblogs.nvidia.com/parallelforall/jetson-tx2-delivers-twice-intelligence-edge/>.
- [62] Wikipedia. *Raspberry Pi* — *Wikipedia, L'enciclopedia libera*. [Online; in data 21-settembre-2017 ]. 2017. URL: [http://it.wikipedia.org/w/index.php?title=Raspberry\\_Pi&oldid=89230951](http://it.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=89230951).
- [63] *Rev.C OV5647 Camera for Raspberry Pi*. [Online; in data 21-settembre-2017 ]. 2015. URL: <http://www.arducam.com/raspberry-pi-camera-rev-c-improves-optical-performance/>.

- [64] Autori vari. *J20 technical reference manual*. A cura di Auvideo. Technical manual. Ver. 1.0. 2017. URL: <https://auvideo.com/manuals/>.