UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

Riconoscitore di colori usando Arduino Uno e sensore di colori ISL29125

Relatore: Prof. Meneghini Matteo

Laureando: Carrera Federico

ANNO ACCADEMICO 2022 – 2023

Data di laurea 27/09/2023

Sommario

1. Introduzione

In un mondo sempre più industrializzato come il nostro, l'elettronica svolge un ruolo fondamentale: gli ingegneri sono alla continua ricerca di nuove tecnologie per ottimizzare le varie operazioni, riducendo la manodopera necessaria e soprattutto i costi di produzione.
I sensori, in particolar modo, sono dei componenti indispensabili che permettono di misurare delle grandezze e di avere dei riscontri. Tra questi vi sono i sensori di colore, che svolgono un ruolo importante in diversi ambiti, sia industriali sia scientifici: vengono infatti impiegati dall'automazione al controllo di qualità, dall'industria tessile all'automotive, e molti altri.

In questa tesi, dunque, verrà descritto e realizzato un riconoscitore di colori Bluetooth, prestando particolare attenzione al sensore di colore e agli altri componenti elettronici utilizzati nella realizzazione circuitale ed evidenziandone vantaggi e svantaggi.

È stato deciso di trattare questo argomento in quanto i colori hanno sempre suscitato il mio interesse, non essendo in grado di distinguerli tra loro in quanto affetto da daltonismo. Tale malattia genetica, infatti, è nota anche come "cecità ai colori", e comporta un'alterata percezione di questi.

2. Arduino

2.1. La piattaforma Arduino

Per la realizzazione del progetto si è deciso di utilizzare una scheda Arduino.
Arduino è nato alla Ivrea Interaction Design Institute come un facile strumento per la prototipazione rapida, rivolto a studenti senza esperienza in elettronica e programmazione. Sia gli schemi circuitali sia il software sono open-source, ovvero resi disponibili a chiunque, in modo da permettere di sviluppare e condividere progetti in base alle proprie esigenze. Per tale motivo, viene molto utilizzato nella didattica educativa.

Le schede Arduino sono in grado di leggere input (come luce su un sensore, un dito su un bottone) e trasformarli in output (ad esempio attivare un motore, accendere un LED). Per far ciò, vengono mandate delle istruzioni al microcontrollore sulla scheda tramite il linguaggio di programmazione Arduino e il software Arduino (IDE).

Con il passare degli anni, sempre più persone si sono riunite attorno a questa piattaforma open-source, contribuendo a un'enorme quantità di conoscenza accessibile che può essere estremamente utile sia per principianti sia per esperti.

Fig. 2.1 - Logo Arduino
(Fonte: Wikipedia.org)

2.2. Perché Arduino?

Il software Arduino è facile da usare per i principianti, ma allo stesso tempo abbastanza flessibile per gli utenti più esperti. Infatti, montando il microcontrollore sulla scheda, ne rende la programmazione più semplice, mantenendo allo stesso tempo tutti i pin disponibili. Funziona su Mac, Windows e Linux.

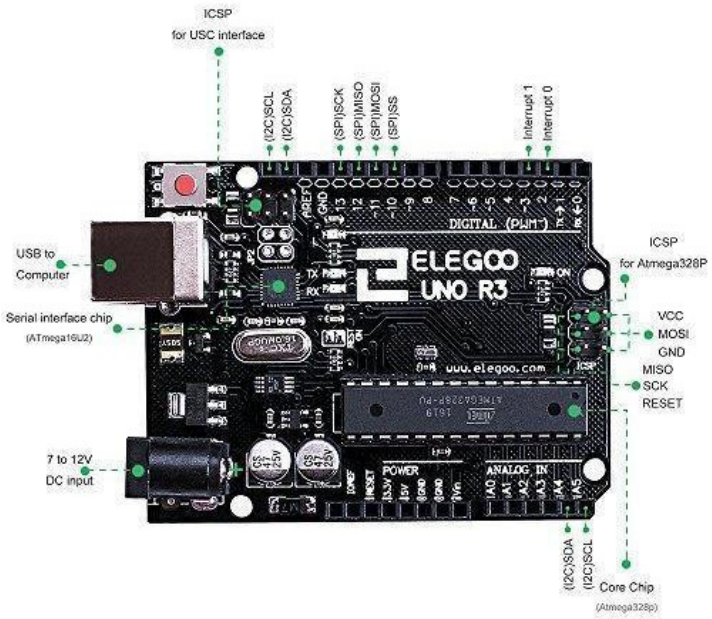
Insegnanti e studenti lo utilizzano per costruire strumenti scientifici economici, per dimostrare principi di chimica e fisica, o ancora per avvicinarsi alla programmazione e alla robotica. Progettisti e architetti costruiscono prototipi interattivi, musicisti e artisti lo usano per installazioni e per sperimentare nuovi strumenti musicali.

Riassumendo, i principali vantaggi che offre Arduino sono:

- **Economico** rispetto ad altre piattaforme a microcontrollori
- **Cross-Platform** – Il software Arduino (IDE) funziona su diversi sistemi operativi
- **Ambiente di programmazione semplice e chiaro** – Semplice per i neofiti, ma tale da permettere agli utenti avanzati di trarne vantaggio
- **Software open-source ed estensibile** – Il software è pubblicato come strumento open-source, in modo da poter essere ampliato da programmatori esperti. Il linguaggio si può ulteriormente espandere usando librerie C++
- **Hardware open-source ed estensibile** – Progettisti di circuiti possono creare la loro versione del modulo, ampliandolo e migliorandolo

2.3. Arduino Uno

Per il progetto si utilizza una scheda Elegoo Uno R3, equivalente all'Arduino Uno R3.



Le sue specifiche sono:

- Microcontrollore ATmega328P
- Tensione di ingresso 7-12V
- 500mA DC dal pin 5V
- 50mA DC dal pin 3.3V
- 40mA DC dai pin I/O
- 14 pin I/O digitali (6 uscite PWM)
- 8 pin di ingresso analogici
- 32kB di memoria flash
- 2kB di SRAM
- 1kB di EEPROM
- Frequenza di clock pari a 16MHz

Fig. 2.2 – Scheda Elegoo Uno R3 (Fonte: elegoo.com)

La scheda può essere alimentata dalla porta USB (dalla quale la scheda viene anche programmata), oppure tramite batteria utilizzando il power jack.

I segnali letti dagli ingressi analogici vengono convertiti in segnali digitali da un convertitore analogico-digitale (DAC) a 10 bit di risoluzione. Ciò significa che si possono avere 1024 diversi valori, da 0 a 1023. Di default il fondoscala è di 5V, ma è possibile abbassare tale valore tramite il pin AREF, in modo tale da avere una quantizzazione più fine. Inoltre, i pin analogici possono anche essere utilizzati come pin I/O digitali.

Alcuni pin I/O digitali hanno delle funzioni aggiuntive:

- Pin 0 (Rx) e 1 (Tx) vengono usati anche per la comunicazione seriale
- Pin 3, 5, 6, 9, 10 e 11 forniscono un'uscita PWM a 8 bit
- Pin 10 (SS), 11 (MOSI), 12(MISO), 13 (SCK) sono usati anche per la comunicazione SPI

Ci sono anche due pin, SDA e SCL, che si usano per la comunicazione I^2C. Al loro posto è possibile usare anche i pin A4 e A5 rispettivamente.

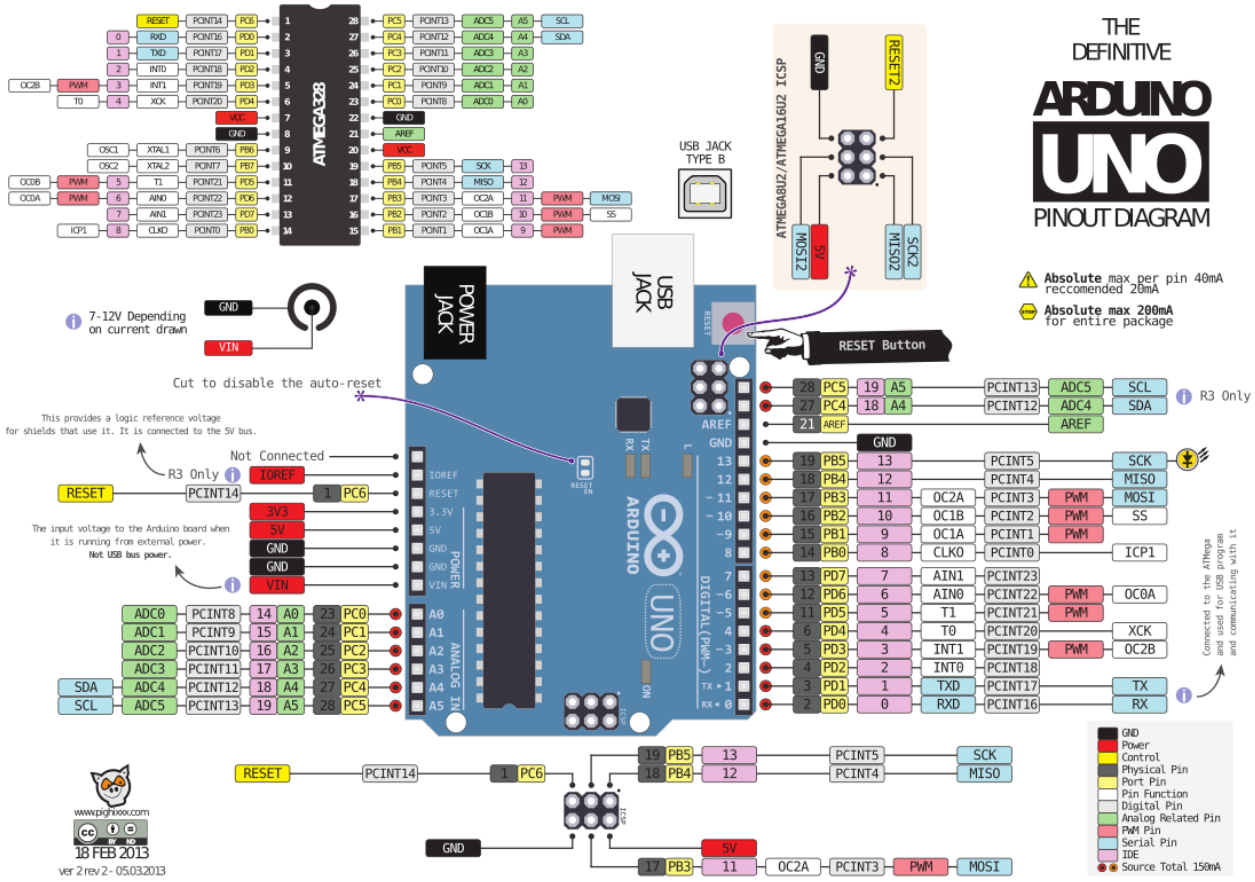


Fig. 2.3 – Diagramma di pinout della scheda Arduino Uno

(Fonte: circuito.io)

3. Sensori di colore

3.1. Principio di funzionamento e applicazioni

Il colore, uno degli attributi più influenti della luce, ha una notevole importanza in diverse applicazioni industriali e scientifiche. Il colore di un materiale può essere usato ad esempio per valutarne le proprietà.
Nelle applicazioni industriali, i sensori di colore spesso vengono utilizzati nel controllo qualità come uno strumento di ispezione visiva. Per esempio, nell'industria alimentare, sono impiegati per monitorare cambiamenti di colore della carne, per verificarne la qualità. Un altro possibile impiego è il controllo della temperatura, per avere una tostatura ideale dei chicchi di caffè.
Ancora, nell'industria tessile, possono determinare se il colore dei tessuti fabbricati è uguale al riferimento. Spesso, infatti, ciò può non essere facile tramite osservazione visiva a causa della luce.

I sensori di colore riconoscono il colore di un materiale in modello RGB (red, green, blue), filtrando le componenti infrarosse e ultraviolette non volute. Tipicamente comprendono anche dei LED bianchi ad alta intensità, che proiettano della luce modulata sull'obiettivo.
La luce bianca è infatti formata dall'unione dei tre colori primari, aventi differenti lunghezze d'onda. Quando essa incide su una superficie, in base alle proprietà del materiale, alcune lunghezze d'onda vengono assorbite e altre riflesse. L'occhio umano, così come il sensore di colore, recepisce quindi le componenti riflesse dalla superficie.

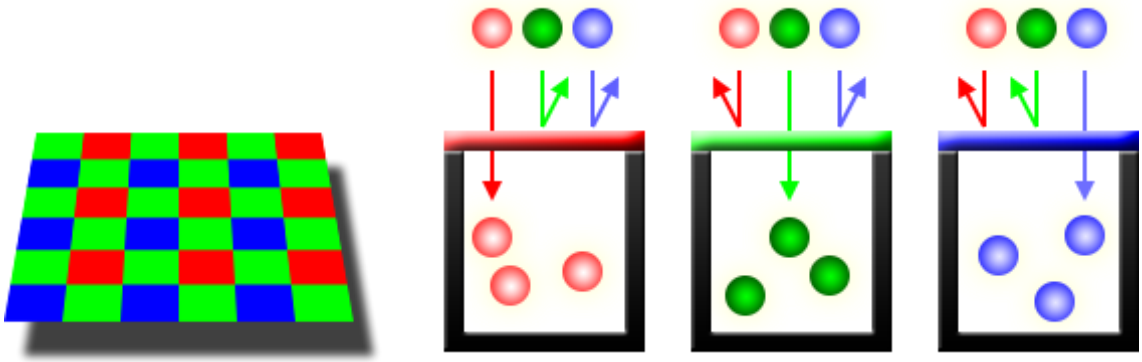


Fig. 3.1 - Color filter array (Fonte: red.com)

La maggior parte dei sensori di colore comprende una schiera di sensori di luce, solitamente fotodiodi o fototransistor. Questi filtrano la luce, in modo tale da accettare solo uno dei tre colori primari. Spesso è anche presente un sensore di luce senza filtri per misurare il livello di luce relativo.

I sensori di luce misurano quindi l'intensità della luce e i risultati sono poi mediati, siccome si hanno più sensori per lo stesso colore. Misurando il livello relativo di rosso, verde e blu si determina il colore dell'oggetto sotto test.

Questi dispositivi non sono perfetti e, siccome fotodiodi e fototransistor non rispondono allo stesso modo per ogni lunghezza d'onda, si hanno delle non-linearità nelle misure dei tre colori primari. Tali non-linearità vengono compensate poi dal dispositivo che processa i risultati.

Esistono diversi tipi di sensori di colore, che differiscono soprattutto per il tipo di output:

- **Output analogico** – Il sensore ha come output una tensione analogica che corrisponde al livello di luce. È il sensore di colore più semplice e al giorno d'oggi viene usato raramente.
- **Output digitale** – Il sensore ha come output una lettura digitale corrispondente all'intensità di ciascun colore.
- **Output a frequenza variabile** – L'uscita è un'onda quadra che cambia la sua frequenza in base all'intensità del colore.

3.2. ISL29125

Il sensore di colori utilizzato per questa tesi è l'ISL29125 prodotto dalla Intersil (Renesas Electronics Corporation), e montato su un modulo prodotto dalla SparkFun.

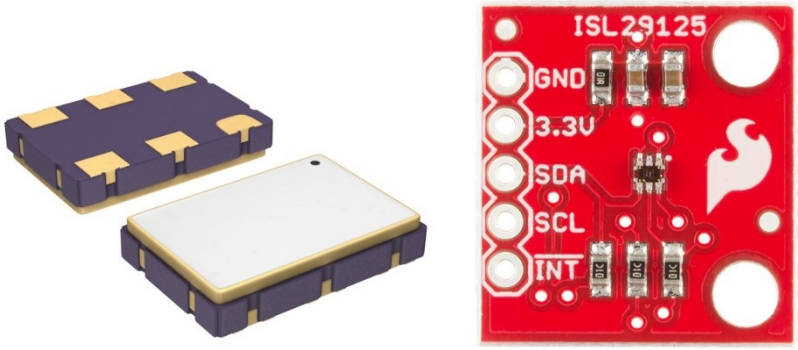


Fig. 3.2 – A sinistra, l'ISL29125 di Intersil. A destra, il componente SparkFun che lo monta (Fonti: digikey.it, sparkfun.com)

L'ISL29125 è un sensore digitale di colori RGB a bassa potenza e ad alta sensibilità, con un'interfaccia I^2C. È progettato in modo tale da filtrare la componente infrarossa (IR) nelle sorgenti luminose, permettendo al dispositivo di operare in ambienti che spaziano dalla luce del sole a camere buie. Il convertitore analogico-digitale (ADC) integrato filtra gli sfarfallii a 50Hz

e 60Hz causati da sorgenti di luce artificiale. Il sensore presenta inoltre due intervalli di sensibilità ottica così da ottimizzarla in base all'applicazione.

Le caratteristiche principali dell'ISL29125 sono:

- Corrente operativa di 56 μA, corrente di spegnimento di 0.5 μA
- Uscita I^2C
- ADC con risoluzione di 16 bit
- Finestre di interrupt programmabili
- Due intervalli di sensibilità ottica:
  - Intervallo 0 = 5.7$m\ lux$ fino a 375 lux
  - Intervallo 1 = 0.152 lux fino a 10$k\ lux$
- Alimentazione operativa da 2.25 a 3.63V
- Alimentazione I^2C da 1.7 a 3.63V

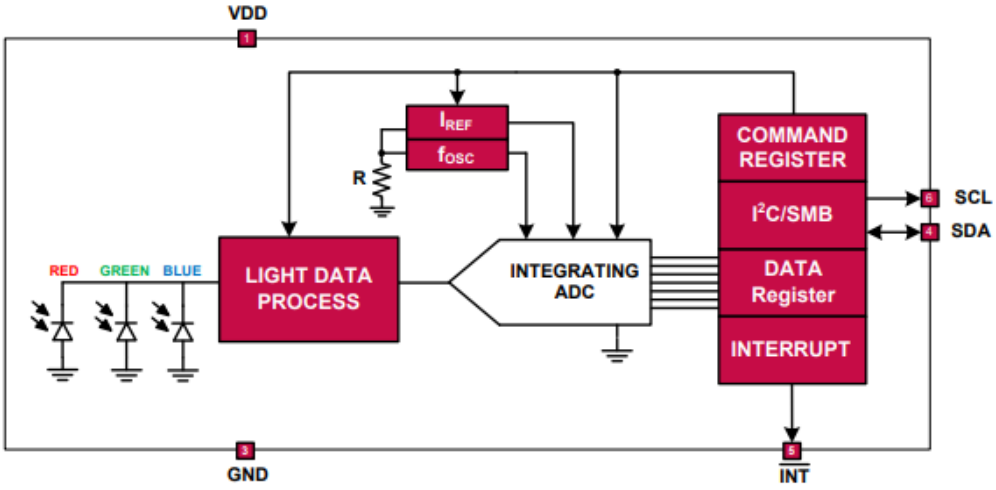


Fig. 3.3 – Diagramma a blocchi dell'ISL29125 (Fonte: ISL29125 datasheet)

L'ISL29125 contiene tre array di fotodiodi, che convertono la luce in corrente. La risposta spettrale per il sensing della luce ambientale si può osservare in figura 3.4:

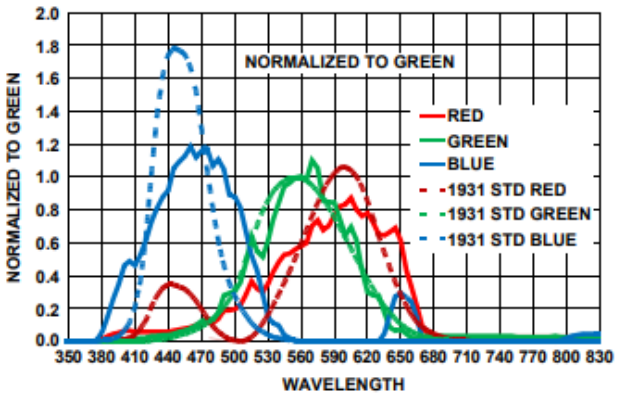


Fig. 3.4 – Risposta spettrale per il sensing della luce ambientale (Fonte: ISL29125 datasheet)

Dopo l'elaborazione del segnale, l'uscita in corrente è convertita a numero digitale dall'ADC integrato, il quale può avere risoluzione di 16 o 12 bit, in base alle esigenze. Il tempo di conversione dell'ADC è quindi inversamente proporzionale alla sua risoluzione.
Il sensore ha inoltre due registri a 8 bit per ogni colore, per salvare al loro interno il byte più significativo e quello meno significativo del valore in uscita dall'ADC. I registri vengono aggiornati dopo ogni ciclo di conversione.

L'ISL29125 presenta anche un pin di interrupt che serve come allarme o per monitoraggio, per determinare se il livello di luce ambientale supera il limite superiore o se scende sotto il limite inferiore. Si noti che l'ADC continua la conversione anche dopo che è stato chiamato l'interrupt. Quindi, se l'utente vuole leggere i valori che hanno attivato l'interruzione, deve essere fatto prima che i registri vengano aggiornati dalla prossima conversione.
Si può inoltre configurare la persistenza del pin di interrupt, per ridurre le possibilità di falsi inneschi dovuti a rumore o picchi improvvisi in condizioni di luce ambientale.

Il sensore supporta il protocollo di trasmissione dati con bus I^2C (Inter-Integrated Circuit), che verrà approfondito nel prossimo capitolo.

4. La comunicazione I^2C

L'I^2C, o TWI (Two Wire Interface), è un protocollo di comunicazione seriale sincrono, ovvero dove:

- viene trasferito un solo bit alla volta (seriale)
- vi è condivisione del clock tra un dispositivo master, il quale controlla il trasferimento dei dati, e uno o più slave, ovvero dispositivi controllati dal master (sincrono)

Non è full-duplex, in quanto non permette ai dispositivi collegati tra loro di inviare dati nello stesso momento. A differenza di altri protocolli, però, ha il vantaggio di funzionare sempre solamente con due fili, anche in presenza di un numero elevato di slave. Infatti l'I^2C utilizza una linea per i dati (SDA, Serial Data) e una linea per il clock (SCL, Serial Clock).
Per far sapere a quale periferica vuole associarsi, il master deve quindi trasmettere sulla linea dati l'indirizzo dello slave desiderato.

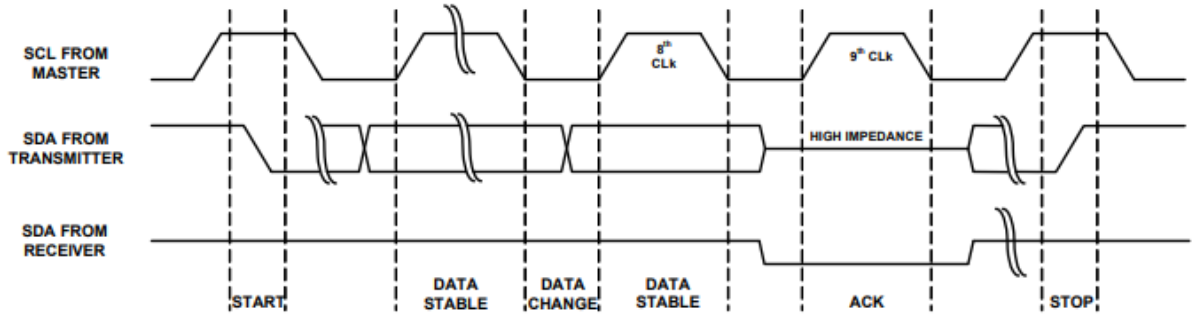


Fig. 4.1 – Condizioni di start e stop, dato stabile e acknowledge
(Fonte: ISL29125 datasheet)

A riposo, solitamente, l'SDA e l'SCL sono a livello logico alto, perché collegate all'alimentazione tramite resistori di pull-up.
Il dispositivo trasmittente abbassa la linea SDA per trasmettere uno "0", e la rilascia per trasmettere un "1". Il master inizia sempre il trasferimento dei dati, solo quando il bus non è occupato, e fornisce il clock sia per le operazioni di trasmissione sia per quelle di ricezione.

Il trasferimento dei dati (in lettura o scrittura) avviene in diversi passaggi:

- **Condizione di start**: l'SDA si abbassa mentre l'SCL resta a livello logico alto. Tutte le periferiche si attivano, in attesa dell'indirizzo del dispositivo desiderato da parte del master.
- Il master manda l'indirizzo a 7 bit della periferica e le altre si spengono. L'ottavo bit viene utilizzato per comunicare allo slave se si vuole fare un'operazione di lettura (bit = "1") o scrittura (bit = "0").

- Durante il nono ciclo di clock, il dispositivo ricevitore abbassa la linea dati per comunicare al trasmittente la corretta ricezione degli 8 bit (**acknowledge**).
- Vengono quindi inviati uno o più byte di dati, ognuno seguito da un acknowledge da parte del ricevitore in caso di corretta ricezione.
- Ogni operazione deve essere terminata da una **condizione di stop**: si ha una transizione da basso ad alto della linea SDA, mentre la linea SCL resta alta.

Fig. 4.2 – Trasferimento di un byte dati con bus I^2C (Fonte: ISL29125 datasheet)

La comunicazione seriale sull'interfaccia I^2C è condotta inviando sempre prima il bit più significativo (MSB, Most Significant Bit) di ciascun byte di dato.

Utilizzando il sensore di colori ISL29125, avendo questo più registri al suo interno, in caso di lettura e scrittura è necessario che il dispositivo master trasmetta anche l'indirizzo del registro nel quale si vogliono leggere o scrivere i dati, dopo aver trasmesso l'indirizzo del dispositivo. L'ISL29125, inoltre, opera come uno slave in tutte le applicazioni.

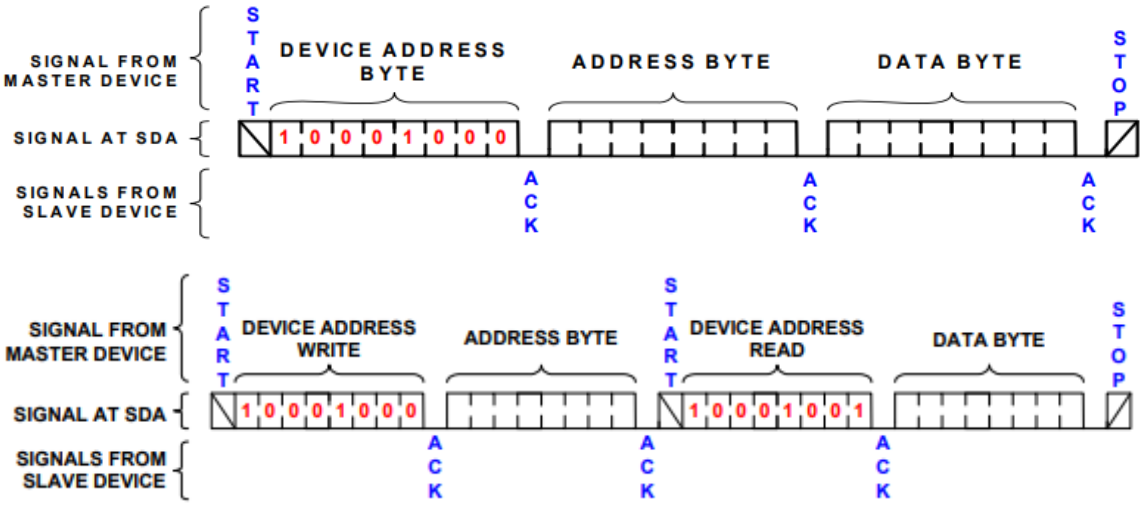


Fig. 4.3 – Operazioni di write (sopra) e read (sotto) nell'ISL29125
(Fonte: ISL29125 datasheet)

5. Bluetooth

5.1. La comunicazione Bluetooth

La comunicazione Bluetooth è uno standard tecnico-industriale di trasmissione dati per reti personali senza fili (WPAN, Wireless Personal Area Network). Fornisce un metodo economico e sicuro per scambiare informazioni tra dispositivi diversi, mettendoli in comunicazione tra loro attraverso una frequenza radio apposita (circa 2.4 GHz) a corto raggio. Tali dispositivi possono essere cellulari, computer, stampanti, console, cuffie e molti altri.

Il Bluetooth si è diffuso da tempo anche nel settore industriale, come ad esempio in strumenti di misura, lettori ottici o simili, per il dialogo con i relativi datalogger.

5.2. Il modulo Bluetooth HC-06

Il modulo Bluetooth usato per questa tesi è l'HC-06, in quanto molto diffuso, economico e di semplice utilizzo. Nello specifico, quello utilizzato in questo progetto è prodotto dalla SunFounder. È un modulo in quanto formato da una serie di componenti montati su scheda.

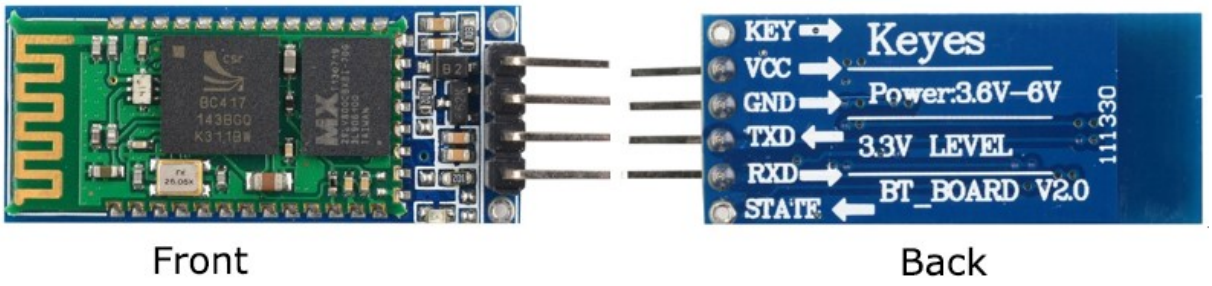


Fig. 5.1 – Modulo HC-06 (Fonte: wiki.sunfounder.cc)

Le sue caratteristiche principali sono:

- Protocollo Bluetooth standard 2.0
- Banda tra 2.40GHz e 2.48GHz
- Tensione di alimentazione tra 3.6V e 6V
- Corrente operativa di 40mA

Il modulo HC-06 può operare in due modalità di funzionamento:

- **Modalità Dati (Data Mode)**: il modulo viene utilizzato per comunicare con altri dispositivi Bluetooth; il trasferimento dei dati avviene in questa modalità
- **Modalità Comando AT (AT Command Mode)**: si può comunicare con il modulo e tramite opportuni comandi AT si possono configurare varie impostazioni e parametri

L’HC-06 presenta sei pin:

- Key: lo stato del pin determina se il modulo funziona in Modalità Comando AT o in Modalità Dati
- Vcc: alimentazione, tra 3.6V e 6V
- GND: connesso a terra
- TXD: i dati seriali sono trasmessi dal modulo attraverso questo pin (3.3V)
- RXD: i dati seriali sono ricevuti dal modulo attraverso questo pin (3.3V)
- State: connesso a un LED per sapere se il modulo è connesso via Bluetooth

L’HC-06 può essere utilizzato esclusivamente come un dispositivo slave. Deve quindi essere collegato a un dispositivo Bluetooth master, come un computer o un cellulare.

Per collegare il modulo a un dispositivo mobile è necessario un terminale Bluetooth. È possibile scaricare gratuitamente una delle numerose applicazioni esistenti che fungono da terminali Bluetooth; oppure, se si vuole realizzarne una propria ma non si hanno particolari conoscenze in programmazione di applicazioni, è possibile progettarla online aiutandosi con degli strumenti specifici.
In particolare, in questo progetto verrà utilizzata l’applicazione “Serial Bluetooth Terminal” sviluppata da Kai Morich.

Il modulo inoltre comunica tramite comunicazione seriale asincrona, ciò significa che non vi è condivisione del segnale di clock. Tuttavia, per poter ricevere o inviare informazioni, è necessario che i dispositivi che comunicano abbiano lo stesso bit rate per avere una trasmissione dei dati corretta.

6. Il progetto

6.1. Il circuito

Il progetto, come accennato nei precedenti capitoli, prevede l'utilizzo del sensore ISL29125 per la rilevazione di un colore tramite modello RGB (Red, Green, Blue). Questo viene poi convertito in un modello differente, chiamato HSV (Hue, Saturation, Value), più intuitivo del modello RGB in quanto utilizza una rappresentazione dei colori basata su quella percepita dall'occhio umano. Il risultato ottenuto viene infine trasferito tramite Bluetooth a un dispositivo mobile, dal quale si potrà verificare il nome e i vari valori del colore misurato.

Di seguito si trova il circuito utilizzato per la realizzazione del progetto:

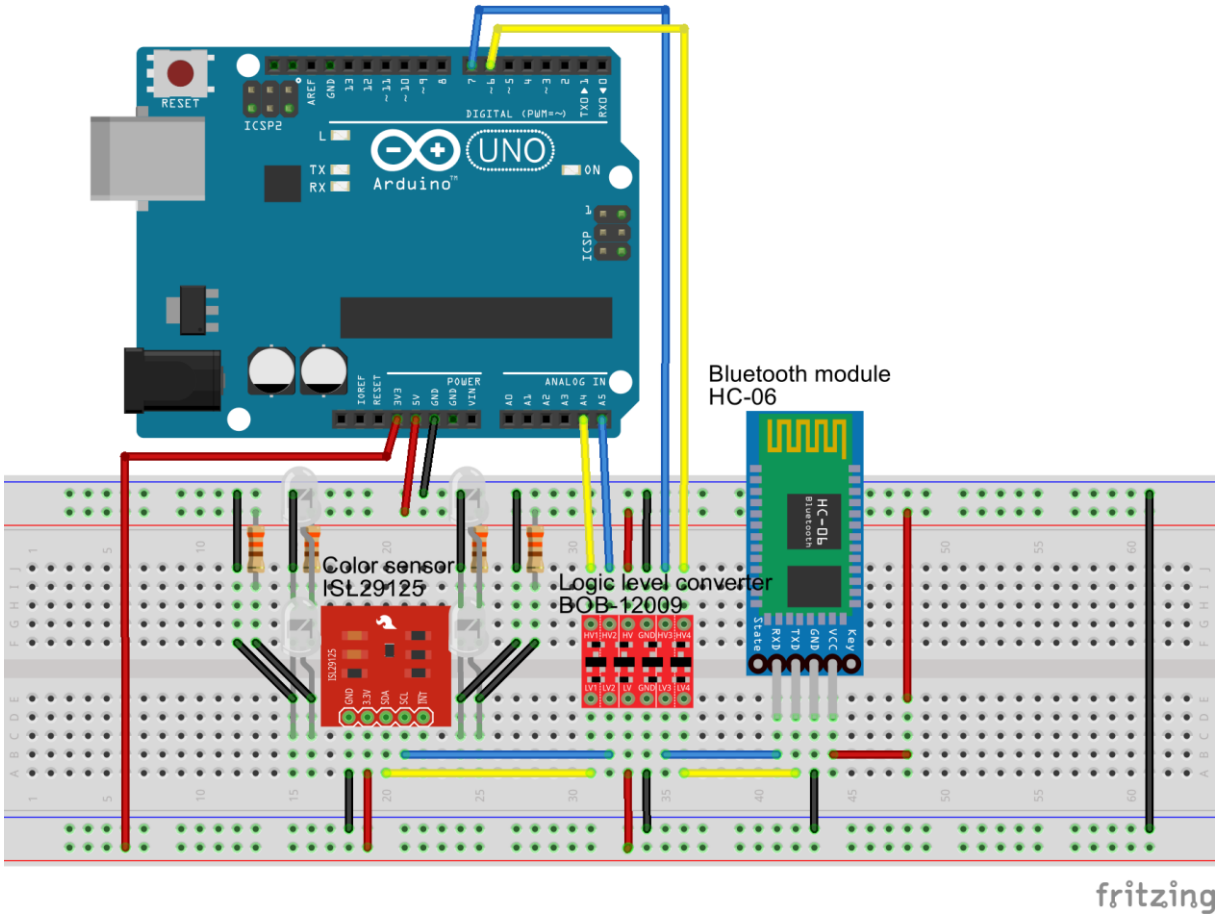


Fig. 6.1 – Il circuito che realizza il progetto di tesi

Come si può notare, oltre ai componenti principali illustrati nei precedenti capitoli, viene utilizzato anche un convertitore logico di livello, in quanto i bus I^2C del sensore di colori, così come i pin TXD e RXD del modulo Bluetooth, non tollerano i 5V in uscita dai pin di Arduino Uno.

Il componente utilizzato è il convertitore logico di livello bidirezionale BOB-12009 prodotto dalla SparkFun.

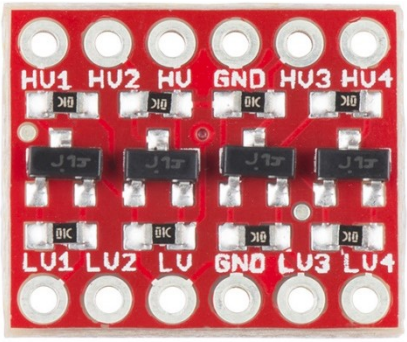


Fig. 6.2 – Convertitore logico di livello BOB-12009 (Fonte: sparkfun.com)

In questa applicazione, la scheda Elegoo R3 viene utilizzata come dispositivo master, mentre il sensore di colori ISL29125 come slave. Inoltre, come illustrato precedentemente, anche il modulo Bluetooth opera come dispositivo slave.

6.2. I modelli RGB e HSV

Il modello RGB è un modello di colori di tipo additivo, nel quale i colori primari della luce rosso, verde e blu sono sommati assieme in modi diversi per riprodurre una vasta gamma di colori.
Il suo scopo principale è il sensing, la rappresentazione e la visualizzazione di immagini in sistemi elettronici, come televisioni e computer.

L'RGB è un modello che dipende dal dispositivo: diversi dispositivi, infatti, rilevano o riproducono un dato valore RGB in modo diverso, in quanto gli elementi di colore (quali fosfori o tinte) e la loro risposta ai singoli colori primari variano da prodotto a prodotto, e anche nel tempo nello stesso dispositivo.

Per formare un colore con il modello RGB, tre raggi di luce (rossa, verde e blu) sono sovrapposti. Ognuno dei raggi è chiamato una componente del colore e ciascuno può avere intensità arbitraria, da completamente spenta ad accesa.

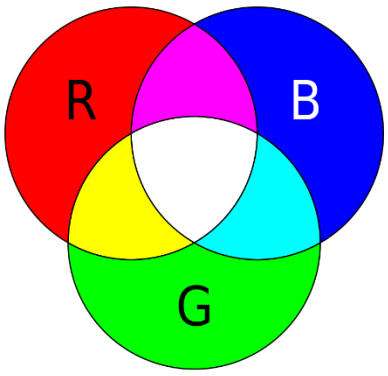


Fig. 6.3 – Colore additivo con modello RGB (Fonte: wikipedia.org)

Intensità nulla per ogni componente dà il colore più scuro (si ha assenza di luce, quindi nero) e intensità massima per ciascuna (tipicamente 2^8-1 = 255) restituisce bianco. Se le tre componenti hanno stessa intensità si ha la scala di grigi.

Quando le intensità sono diverse il risultato è una tonalità colorata, più o meno satura in base alla differenza tra l'intensità più forte e quella più debole dei colori primari.

Se una delle componenti è più intensa delle altre, il colore ha una tonalità simile a quella di quel determinato colore primario. Nel caso in cui due componenti siano prevalenti, la tonalità risultante sarà un colore secondario, dato dalla somma dei due colori primari di intensità simile.

Color	Color name	Hex	(R,G,B)
	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
	Red	#FF0000	(255,0,0)
	Lime	#00FF00	(0,255,0)
	Blue	#0000FF	(0,0,255)
	Yellow	#FFFF00	(255,255,0)
	Cyan	#00FFFF	(0,255,255)
	Magenta	#FF00FF	(255,0,255)

Fig. 6.4 – Principali colori e la loro rappresentazione RGB (Fonte: rapidtables.com)

Spesso, però, descrivere i colori tramite modello RGB risulta non facile e poco intuitivo. Infatti, se consideriamo ad esempio di voler passare da un arancione a uno meno saturo, bisogna diminuire l'intensità del colore primario rosso e allo stesso tempo aumentare quella del verde e del blu, come mostrato nella figura sottostante.

Fig. 6.5 – Variazione di colori tramite modello RGB (Fonte: wikipedia.org)

Per avere una mappatura dei colori più semplice e comprensibile, in questo progetto i valori RGB vengono quindi convertiti in modello HSV. Tale modello prende in considerazione la tonalità, la saturazione e il valore dei colori, rendendo quindi più intuitive le transazioni tra questi.

Per convertire i colori in modello HSV si usano le seguenti formule:

$$H = \begin{cases} 0, & R = G = B \\ cos^{-1}\left(\dfrac{R - \frac{G}{2} - \frac{B}{2}}{\sqrt{R^2 + G^2 + B^2 - R \cdot G - R \cdot B - G \cdot B}}\right), & G > B \ \vee \ G = B \neq R \\ 360° - cos^{-1}\left(\dfrac{R - \frac{G}{2} - \frac{B}{2}}{\sqrt{R^2 + G^2 + B^2 - R \cdot G - R \cdot B - G \cdot B}}\right), & G < B \end{cases}$$

$$S = \begin{cases} 0, & MAX(R,G,B) = 0 \\ 1 - \dfrac{min(R,G,B)}{MAX(R,G,B)}, & MAX(R,G,B) > 0 \end{cases}$$

$$V = \frac{MAX(R,G,B)}{255}$$

dove R, G e B sono le intensità di rosso, verde e blu del colore in modello RGB, e per H, S e V si intende:

- **Tonalità (H, hue)**: attributo che definisce il colore in termine di spettro dei colori, ossia il colore puro.
- **Saturazione (S, saturation)**: intensità o vivacità di un colore, ossia quanto un colore si discosta dalla scala di grigi.
- **Valore (V, value)** o lucentezza (B, brightness): brillantezza di un colore, indica quanto un colore è luminoso o scuro.

Lo spazio dei colori descritto dal modello HSV è cilindrico, con la tonalità che indica a che angolazione attorno all'asse centrale si trova il colore, la saturazione la sua distanza dallo stesso asse e il valore a che altezza si trova nel cilindro.
Per definizione, quindi, la tonalità ha valori compresi tra 0° e 360°, dove a 0° si ha il rosso, a 120° il verde e a 240° il blu; la saturazione e il valore sono invece compresi tra 0 e 1, dove saturazione nulla corrisponde alla scala di grigi e valore nullo al nero.

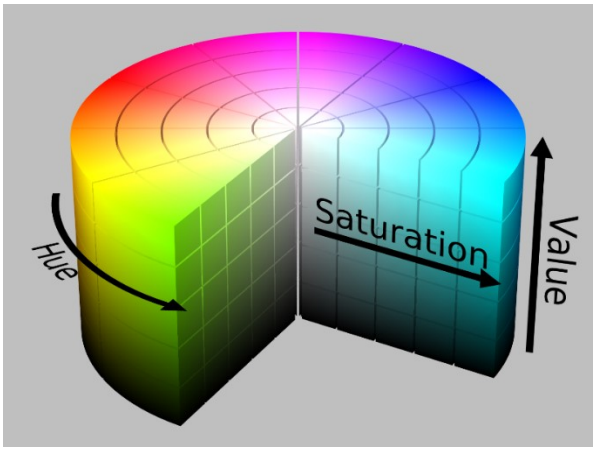


Fig. 6.6 – Spazio dei colori in modello HSV (Fonte: wikipedia.org)

Di seguito si trova una tabella con i principali colori e le loro rappresentazioni in modello RGB e HSV, per evidenziarne le differenze:

Color	Color name	Hex	(R,G,B)	(H,S,V)
	Black	#000000	(0,0,0)	(0°,0%,0%)
	White	#FFFFFF	(255,255,255)	(0°,0%,100%)
	Red	#FF0000	(255,0,0)	(0°,100%,100%)
	Lime	#00FF00	(0,255,0)	(120°,100%,100%)
	Blue	#0000FF	(0,0,255)	(240°,100%,100%)
	Yellow	#FFFF00	(255,255,0)	(60°,100%,100%)
	Cyan	#00FFFF	(0,255,255)	(180°,100%,100%)
	Magenta	#FF00FF	(255,0,255)	(300°,100%,100%)

Fig. 6.7 – Rappresentazione RGB e HSV dei principali colori (Fonte: rapidtables.com)

6.3. Calibrazione e funzionamento

Innanzitutto il sensore di colore deve essere inizializzato. Ciò viene fatto nel setup, tramite la funzione *.init()* della libreria *SparkFunISL29125* del produttore. Tale metodo fa diverse operazioni:

- Inizializza la comunicazione I^2C
- Verifica che il sensore sia collegato attraverso il suo ID
- Resetta tutti i registri e le impostazioni a quelle di default
- Imposta i registri per l'uso più comune: modalità RGB, range di luminosità fino a $10k\ lux$ e compensazione elevata dell'infrarosso

In questo modo il sensore è in modalità di continua acquisizione, quindi aggiorna costantemente i dati all'interno dei suoi registri.

Come illustrato in precedenza, l'ISL29125 presenta sei registri a 8 bit, due per ogni colore primario. Per ciascuna componente RGB si può quindi salvare dati di valore compreso tra 0 e 2^{16}-1 e idealmente gli estremi dovrebbero rappresentare rispettivamente la totale assenza o presenza di un dato colore primario.
Questi però non verranno mai raggiunti, in quanto difficilmente si può avere veramente totale riflessione e totale assorbimento dei colori. Il sensore richiede quindi una calibrazione, in modo tale che gli vengano forniti i riferimenti di bianco e nero per poter analizzare correttamente il

colore sotto test. Tali riferimenti si ottengono ricavando i valori più bassi e più alti misurabili dei tre colori primari: per ricavare i valori minimi leggo i dati salvati nei registri mentre valuto un oggetto di colore nero; per i valori massimi si ripete la misura con un oggetto bianco. Tali valori ottenuti saranno quindi rispettivamente lo 0 e il 255 per i colori primari.

Per misurare i valori RGB di un colore basta quindi leggere i dati salvati nei registri del sensore e, tramite la funzione *map* di Arduino, fare una proporzione con i valori di bianco e nero ottenuti in precedenza.
Ciò però non basta ad assicurare che l'intensità misurata sia compresa tra 0 e 255, in quanto i riferimenti di bianco e nero utilizzati non sono universali. Potrebbe quindi capitare che, per colori con una o più componenti particolarmente assenti o presenti, le intensità cadano leggermente all'infuori del range di valori RGB. Occorre quindi utilizzare una funzione di Arduino chiamata *constrain* che permette di impostare due limiti e, in caso la variabile sotto test abbia valore inferiore al minimo o superiore al massimo, il metodo le assegna rispettivamente il valore minimo o quello massimo, garantendo quindi che cada sempre all'interno dell'intervallo desiderato.
La funzione che ricava i valori RGB sarà allora come segue:

```
void RGB(){
  //Leggo valori dai registri (16 bit)
  unsigned int red = RGB_sensor.readRed();
  unsigned int green = RGB_sensor.readGreen();
  unsigned int blue = RGB_sensor.readBlue();

  //Converto in valori RGB (Nota: può non essere compreso tra 0 e 255)
  int redVal = map(red, redLow, redHigh, 0, 255);
  int greenVal = map(green, greenLow, greenHigh, 0, 255);
  int blueVal = map(blue, blueLow, blueHigh, 0, 255);

  //Limito i valori RBG tra 0 e 255
  redRGB = constrain(redVal, 0, 255);
  greenRGB = constrain(greenVal, 0, 255);
  blueRGB = constrain(blueVal, 0, 255);
}
```

Per ottenere i valori del modello HSV si utilizzano semplicemente le formule riportate nel precedente sottocapitolo.

In seguito, da tonalità, saturazione e valore non è complesso ricavare il nome del colore sotto test. È sufficiente, infatti, definire una struttura per i colori, in modo tale che i valori HSV ricadano all'interno degli intervalli creati per ogni colore. La struttura *ColorMapping* è stata definita nel seguente modo:

```
struct ColorMapping{
    int hueMin;
    int hueMax;
    float satMin;
    float satMax;
    float valMin;
    float valMax;
    String colorName;
        };
```

Si crea quindi un array di *ColorMapping*, con una mappatura manuale dei vari colori. In tal modo, se i valori di tonalità, saturazione e valore ricadono negli intervalli di uno di questi, a quei valori verrà associato il *colorName* di quel dato *ColorMapping*, come segue:

```
int numMappings = sizeof(colorMappings)/sizeof(colorMappings[0]); //n°colori
for(int i=0; i<numMappings; i++){
  if(hue>=colorMappings[i].hueMin && hue<=colorMappings[i].hueMax &&
     saturation>=colorMappings[i].satMin &&
     saturation<=colorMappings[i].satMax &&
     value>=colorMappings[i].valMin && value<=colorMappings[i].valMax)
    return colorMappings[i].colorName;
}
return "Unknown";
```

Per inviare i dati misurati a un dispositivo mobile attraverso Bluetooth si utilizza la libreria *SoftwareSerial*, che permette di dichiarare l'oggetto modulo Bluetooth e indicare i pin di Arduino utilizzati per la comunicazione seriale, oltre a leggere e scrivere dati.
Il dispositivo mobile necessita di un terminale Bluetooth per ricevere o inviare messaggi. Da questo si può comunicare all'Arduino le operazioni che si vogliono effettuare tramite i seguenti comandi:

- ***calibrate white*** e ***calibrate black*** permettono di calibrare il sensore con i riferimenti di bianco e nero rispettivamente;
- ***rgb*** restituisce i valori RGB del colore sotto test;
- ***hsv*** restituisce i valori HSV del colore sotto test;
- ***name*** restituisce il nome del colore sotto test;
- ***complete analysis*** restituisce i valori RGB, HSV e il nome del colore sotto test.

Nel messaggio contenente il comando è stato deciso di utilizzare un carattere di inizio pacchetto e uno terminatore, in questo caso il cancelletto '**#**' ed il punto e virgola '**;**' rispettivamente. Infatti, utilizzando un baud rate relativamente basso, senza questi si rischia che la parte di messaggio salvata venga eliminata prima che questo venga letto completamente, non portando l'operazione a buon fine.

Utilizzando un baud rate più alto ciò non avviene, però ci sono delle interferenze che sporadicamente non permettono la corretta ricezione del messaggio, in quanto viene ricevuto dall'Arduino un carattere errato all'interno del comando.

Con i caratteri di inizio pacchetto e terminatore si garantisce quindi il corretto invio dei comandi tramite Bluetooth:

Fig. 6.8 – Esempio di comando su terminale Bluetooth

Nel dispositivo mobile si avrà una schermata simile alla seguente durante l'utilizzo:

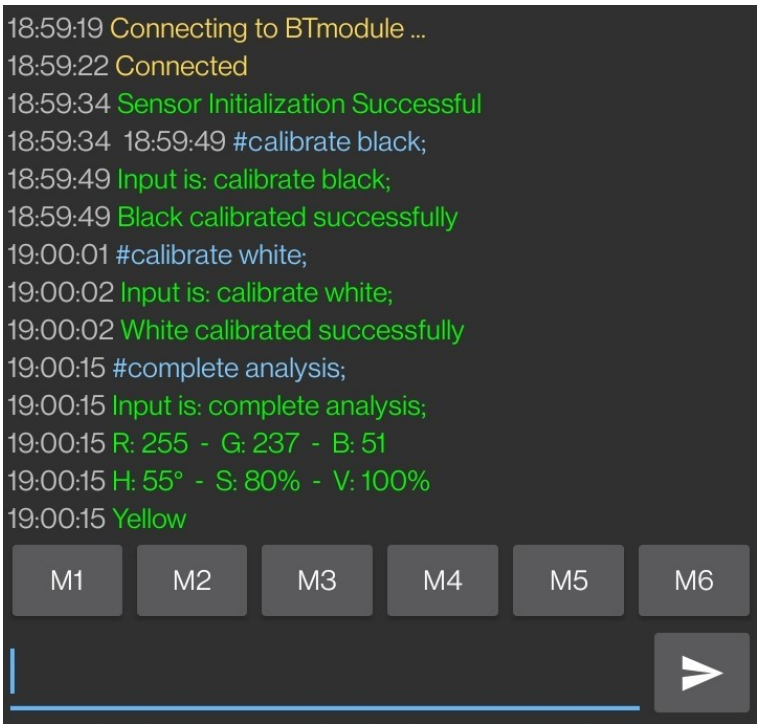


Fig. 6.9 – Tipica schermata su terminale Bluetooth

Per maggiori dettagli sul codice Arduino utilizzato si rimanda a pagina 23, dove è riportato il codice completo e commentato utilizzato per la realizzazione del progetto.

7. Considerazioni e conclusioni

Il progetto realizzato è molto versatile e applicabile a diverse situazioni. A causa del ridotto costo dei componenti utilizzati, però, presenta dei difetti.

Innanzitutto, nonostante faccia misurazioni molto più stabili rispetto ad altri sensori dello stesso prezzo, l'ISL29125 non sempre restituisce la tonalità precisa del colore. Infatti, come accennato in precedenza, il sensore ha una diversa risposta spettrale ai diversi colori primari, in quanto di lunghezze d'onda (e quindi di frequenze) differenti:

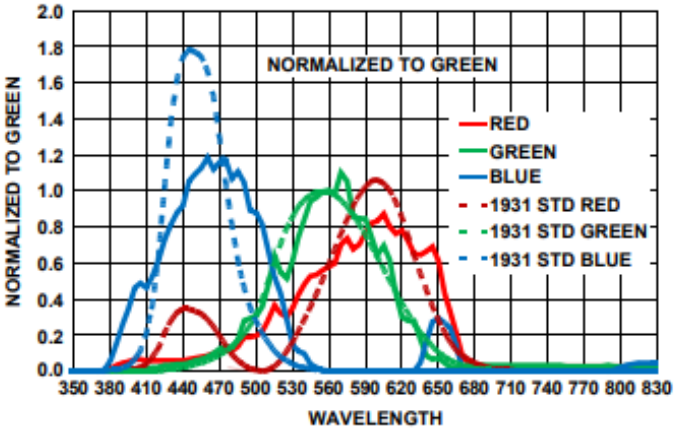


Fig. 7.1 - Risposta spettrale per il sensing della luce ambientale (Fonte: ISL29125 datasheet)

Come si può notare, il sensore è più sensibile alla componente blu, seguita dal verde e infine dal rosso. Ciò si può notare anche nella fase di calibrazione, in quanto nei riferimenti di bianco e nero le componenti blu e verdi hanno valori più elevati rispetto al rosso.
La differente risposta ai colori primari si traduce anche in una minore precisione nella misura del rosso, in quanto questo potrà spaziare un numero ridotto di valori.
Inoltre bisogna tener conto che, se si effettuano delle misure su oggetti lucidi, questi riflettono meglio la luce, dando quindi anche in questo caso una misura meno accurata.

Un'ulteriore non-idealità è data dai LED bianchi utilizzati per illuminare l'oggetto sotto test. Infatti non esistono LED in grado di produrre una luce veramente bianca, la quale risulterà di una tonalità bluastra.
I LED bianchi vengono prodotti con diverse tecnologie, ma la più utilizzata è quella di conversione ai fosfori. Questi vengono ottenuti a partire da LED blu in GaN (Nitruro di Gallio) che vengono ricoperti da fosfori, i quali emettono luce gialla se illuminati da fotoni con frequenza nel blu.

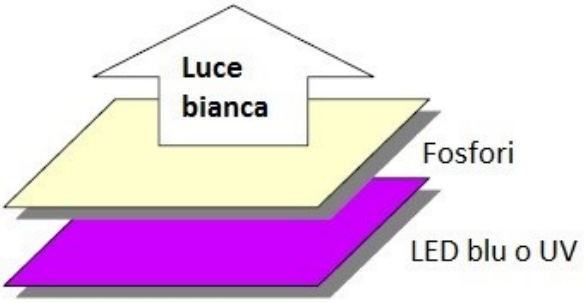


Fig. 7.2 – LED a conversione ai fosfori (Fonte: progettazioneottica.it)

Anche in fase di calibrazione è possibile che venga introdotto qualche piccolo errore. Infatti per i riferimenti di bianco e nero non vengono utilizzati dei riferimenti "universali", bensì i colori più vicini possibile alla concezione di bianco e nero.

In conclusione, il progetto è utilizzabile per diverse applicazioni in numerosi ambiti e, modificandolo ulteriormente o rendendolo più complesso, è possibile anche adattarlo secondo le proprie esigenze.
Nonostante qualche imprecisione, il riconoscitore di colori svolge egregiamente il suo scopo. Nel caso fossero necessarie delle misure particolarmente accurate, però, è consigliabile ovviare in qualche modo agli svantaggi sopra descritti, oltre ad adottare una mappatura dei colori più fine.

8. Codice completo per il progetto

```
#include <Wire.h>
#include <SparkFunISL29125.h>
#include <math.h>
#include <SoftwareSerial.h>

//Dichiaro l'oggetto sensore RGB
SFE_ISL29125 RGB_sensor;
//Dichiaro l'oggetto modulo Bluetooth
//Uso i digital pin 6 e 7 come Tx e Rx
SoftwareSerial BTserial(6, 7);

//Inizializzo la stringa per i comandi
String command;
char data;

//Valori per la calibrazione del sensore
unsigned int redLow, redHigh, greenLow, greenHigh, blueLow, blueHigh;

//Dichiaro le variabili per i valori RGB
unsigned int redRGB, blueRGB, greenRGB;

//Dichiaro le variabili per i valori HSV
int H;
float S, V;

void setup() {
  //Inizializzo comunicazione seriale
  BTserial.begin(38400);

  //Inizializzo l'ISL29125
  if (RGB_sensor.init())
  {
    BTserial.write("Sensor Initialization Successful\n\r");
  }
}

void loop() {
  //Se ho dati dal dispositivo mobile
  while(BTserial.available()){
    data = (char)BTserial.read();

    //Se ho il carattere di inizio pacchetto
    if(data == 35)
      //Resetto la stringa per i comandi
      command = "";

    else{
    command += data;
```

```
//Se ho il carattere di fine comando
if(data == ';'){
BTserial.println("Input is: " + command);

  //Comandi:
  //Calibro il bianco (valori massimi)
  if(command.equals("calibrate white;")){
    redHigh = RGB_sensor.readRed();
    greenHigh = RGB_sensor.readGreen();
    blueHigh = RGB_sensor.readBlue();
    BTserial.println("White calibrated successfully");
  }

  //Calibro il nero (valori minimi)
  else if(command.equals("calibrate black;")){
    redLow = RGB_sensor.readRed();
    greenLow = RGB_sensor.readGreen();
    blueLow = RGB_sensor.readBlue();
    BTserial.println("Black calibrated successfully");
  }

  //Analisi completa (RGB, HSV, nome)
  else if(command.equals("complete analysis;")){
    //Calcolo e stampo a schermo i valori RGB
    RGB();
    BTserial.print("R: "); BTserial.print(redRGB);
    BTserial.print("  -  G: "); BTserial.print(greenRGB);
    BTserial.print("  -  B: "); BTserial.println(blueRGB);
    //Calcolo e stampo a schermo i valori HSV
    RGBtoHSV(redRGB, greenRGB, blueRGB);
    BTserial.print("H: "); BTserial.print(String(H) + "°");
    BTserial.print("  -  S: "); BTserial.print(String((S*100),0) + "%");
    BTserial.print("  -  V: "); BTserial.println(String((V*100),0) + "%");
    //Stampo a schermo il nome del colore
    BTserial.println(getColorName(H, S, V));
  }

  //Stampo i valori RGB
  else if(command.equals("rgb;")){
    RGB();
    BTserial.print("R: "); BTserial.print(redRGB);
    BTserial.print("  -  G: "); BTserial.print(greenRGB);
    BTserial.print("  -  B: "); BTserial.println(blueRGB);
  }

  //Stampo i valori HSV
  else if(command.equals("hsv;")){
    RGB();
    RGBtoHSV(redRGB, greenRGB, blueRGB);
    BTserial.print("H: "); BTserial.print(String(H) + "°");
    BTserial.print("  -  S: "); BTserial.print(String((S*100),0) + "%");
```

```
        BTserial.print("  -  V: "); BTserial.println(String((V*100),0) + "%");
      }

      //Stampo il nome del colore
      else if(command.equals("name;")){
        RGB();
        RGBtoHSV(redRGB, greenRGB, blueRGB);
        BTserial.println(getColorName(H, S, V));
      }

      //Comando non riconosciuto
      else
        BTserial.println("Not a command");
    }
  }
 }
}

//Calcola i valori RGB
void RGB(){
  //Leggo valori dai registri (16 bit)
  unsigned int red = RGB_sensor.readRed();
  unsigned int green = RGB_sensor.readGreen();
  unsigned int blue = RGB_sensor.readBlue();

  //Converto in valori RGB (Nota: può non essere compreso tra 0 e 255)
  int redVal = map(red, redLow, redHigh, 0, 255);
  int greenVal = map(green, greenLow, greenHigh, 0, 255);
  int blueVal = map(blue, blueLow, blueHigh, 0, 255);

  //Limito i valori RBG tra 0 e 255
  redRGB = constrain(redVal, 0, 255);
  greenRGB = constrain(greenVal, 0, 255);
  blueRGB = constrain(blueVal, 0, 255);
}

//Converte i valori RGB in HSV
void RGBtoHSV(long R, long G, long B){
  float M = max(max(R, G), B);
  float m = min(min(R, G), B);

  //hue (tonalità)
  if (G==B && G==R)
    H = 0;
  else if(G >= B)
    H = RAD_TO_DEG * acos((R - (float)G/2 - (float)B/2)/sqrt(pow(R,2) +
        pow(G,2) + pow(B,2) - R*G - R*B - G*B));
  else
    H = 360 - RAD_TO_DEG * acos((R - (float)G/2 - (float)B/2)/sqrt(pow(R,2) +
        pow(G,2) + pow(B,2) - R*G - R*B - G*B));
```

```
  //saturation (saturazione)
  if(M>0)
    S = 1-(m/M);
  else S=0;

  //value (valore; o brightness, luminosità)
  V = M/255;
}

//Associa ai valori HSV un nome di colore
String getColorName(int hue, float saturation, float value){
  //Definisco la struttura per la mappatura del colore
  struct ColorMapping{
    int hueMin;
    int hueMax;
    float satMin;
    float satMax;
    float valMin;
    float valMax;
    String colorName;
  };

  //Colori
  ColorMapping colorMappings[] = {
    {0, 360, 0, 1.0, 0, 0.2, "Black"},
    {0, 360, 0, 0.15, 0.8, 1.0, "White"},
    {0, 360, 0, 0.15, 0.2, 0.8, "Gray"},
    {0, 12, 0.7, 1.0, 0.2, 1.0, "Red"},
    {0, 12, 0.15, 0.7, 0.2, 1.0, "Pink"},
    {13, 41, 0.15, 1.0, 0.75, 1.0, "Orange"},
    {13, 41, 0.15, 1.0, 0.2, 0.75, "Brown"},
    {42, 69, 0.15, 1.0, 0.2, 1.0, "Yellow"},
    {70, 137, 0.15, 1.0, 0.2, 1.0, "Green"},
    {138, 166, 0.15, 1.0, 0.6, 1.0, "Spring Green"},
    {167, 187, 0.15, 1.0, 0.6, 1.0, "Aqua"},
    {138, 187, 0.15, 1.0, 0.2, 0.6, "Green"},
    {188, 217, 0.15, 1.0, 0.2, 1.0, "Azure"},
    {218, 251, 0.15, 1.0, 0.2, 1.0, "Blue"},
    {252, 299, 0.15, 1.0, 0.2, 1.0, "Purple"},
    {300, 326, 0.15, 1.0, 0.2, 1.0, "Magenta"},
    {327, 337, 0.15, 1.0, 0.2, 1.0, "Pink"},
    {338, 360, 0.15, 0.7, 0.2, 1.0, "Pink"},
    {338, 360, 0.7, 1.0, 0.2, 1.0, "Red"}
  };

  int numMappings = sizeof(colorMappings)/sizeof(colorMappings[0]);//n° colori
  for(int i=0; i<numMappings; i++){
    if(hue>=colorMappings[i].hueMin && hue<=colorMappings[i].hueMax &&
       saturation>=colorMappings[i].satMin &&
       saturation<=colorMappings[i].satMax &&
       value>=colorMappings[i].valMin && value<=colorMappings[i].valMax)
```

```
        return colorMappings[i].colorName;
    }
    return "Unknown";
}
```

9. Riferimenti

Testo:

- https://www.bestech.com.au/blogs/understanding-colour-sensors-working-principle-and-applications
- https://www.arduino.cc/en/Guide/Introduction#why-arduino
- https://www.elegoo.com/en-it/products/elegoo-uno-r3-board
- https://www.circuito.io/blog/arduino-uno-pinout/
- https://dronebotworkshop.com/arduino-color-sense/
- ISL29125 datasheet
- https://www.makerslab.it/i-moduli-hc-05-e-hc-06-con-arduino/
- https://www.utmel.com/components/hc-06-bluetooth-module-pinout-datasheet-pdf-and-arduino-connection?id=884
- https://en.wikipedia.org/wiki/RGB_color_model
- https://en.wikipedia.org/wiki/HSL_and_HSV
- github.com/sparkfun/SparkFun_ISL29125_Breakout_Arduino_Library/tree/V_1.0.1
- https://www.progettazioneottica.it/tecnologia-led-illuminazione/556
- https://www.pcprofessionale.it/news/magazine/attenti-alla-luce-blu/

Immagini:

- Fig. 2.1: https://it.wikipedia.org/wiki/Arduino_%28hardware%29
- Fig. 2.2: https://www.elegoo.com/en-it/products/elegoo-uno-r3-board
- Fig. 2.3: https://www.circuito.io/blog/arduino-uno-pinout/
- Fig. 3.1: https://www.red.com/red-101/bayer-sensor-strategy
- Fig. 3.2: https://www.sparkfun.com/products/12829
  https://www.digikey.it/it/product-highlight/i/intersil/isl29125-rgb-sensors
- Fig. 3.3: ISL29125 datasheet
- Fig. 3.4: ISL29125 datasheet
- Fig. 4.1: ISL29125 datasheet
- Fig. 4.2: ISL29125 datasheet
- Fig. 4.3: ISL29125 datasheet
- Fig. 5.1: wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06

- Fig. 6.2: https://www.sparkfun.com/products/12009
- Fig. 6.3: https://en.wikipedia.org/wiki/RGB_color_model
- Fig. 6.4: https://www.rapidtables.com/convert/color/rgb-to-hsv.html
- Fig. 6.5: https://en.wikipedia.org/wiki/HSL_and_HSV
- Fig. 6.6: https://en.wikipedia.org/wiki/HSL_and_HSV
- Fig. 6.7: https://www.rapidtables.com/convert/color/rgb-to-hsv.html
- Fig. 7.1: ISL29125 datasheet
- Fig. 7.2: https://www.progettazioneottica.it/tecnologia-led-illuminazione/556