Francesco Bafumo

Marco La Martina

Screen calibration

on bare-metal

Prof. Daniele Peri

Sistemi Embedded

Ingegneria Informatica LM

Palermo, 11/04/2020

Sommario

Introduzione

Il corso di Sistemi Embedded del primo anno del corso di laurea in Ingegneria Informatica Magistrale presso l'Università degli Studi di Palermo richiede agli studenti un progetto che consiste nello sviluppo di un'applicazione embedded funzionante e documentata, che dovrà essere discussa durante la prova orale, insieme a tutti gli argomenti trattati durante il corso in seguito riportati.

Argomenti principali del corso:

- Embedded systems architectures
- ISA level programming
- Machine language and assembly (ARM)
- BCM2835/7 Architecture
- STM32-F446 architecture
- Forth programming
- Buses (I2C, SPI, UART e USART)

L'applicazione in questione inoltre dovrà essere realizzata su un qualsiasi hardware a scelta dello studente ma su un ambiente di sviluppo specifico, ovvero pijFORTHos.
Il seguente documento descrive l'architettura e l'obiettivo del sistema sviluppato. Sono dunque presentate tutte le caratteristiche sia dell'applicazione embedded realizzata che degli strumenti adottati a tal fine. Saranno descritti inizialmente tutte le componenti hardware e i meccanismi di comunicazione impiegati per procedere poi con una discussione sulla struttura e sull'obiettivo dell'applicazione. Quest'ultima consiste nella visualizzazione su un display di una serie di colori di intensità variabile e la loro rilevazione attraverso un apposito sensore di colore. Il fine è quello di mostrare le differenze tra i valori dei colori mostrati a schermo e quelli rispettivamente rilevati dal sensore attraverso l'impiego di grafici bidimensionali, noti come tabelle di look-up (LUT). Inoltre, il sistema comprenderà un'ulteriore funzionalità che consiste nel riconoscimento di oggetti attraverso il loro colore; questa operazione, che sarà effettuata ovviamente per mezzo del sensore, dovrà includere anche il calcolo del numero di occorrenze degli oggetti dello stesso colore e mostrarlo sul display.

Hardware

In questa sezione sarà descritto e analizzato ogni componente hardware che è stato impiegato per la realizzazione del progetto.

Raspberry PI 4B

La macchina target scelta per lo sviluppo dell'applicazione è il Raspberry PI 4B. Il Raspberry Pi è un *single-board computer* sviluppato nel Regno Unito dalla Raspberry Pi Foundation. La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux o RISC OS. La versione 4B, annunciata il 24 giugno 2019, sostituisce la precedente 3B+. Basato su Broadcom Chip BCM2837 [1], presenta le seguenti specifiche:

- Una CPU ARM Cortex-A72 quad-core a 64 bit da 1,5 GHz
- 2 GB di LPDDR4 SDRAM
- Bluetooth 5.0
- Gigabit Ethernet a pieno rendimento
- Grafica VideoCore VI, che supporta OpenGL ES 3.x
- Due porte USB 3.0 e due porte USB 2.0
- Decodifica hardware 4Kp60 del video HEVC
- Rete wireless dual-band 802.11ac
- Supporto per doppio monitor, con risoluzioni fino a 4K
- Completa compatibilità con i precedenti prodotti Raspberry Pi

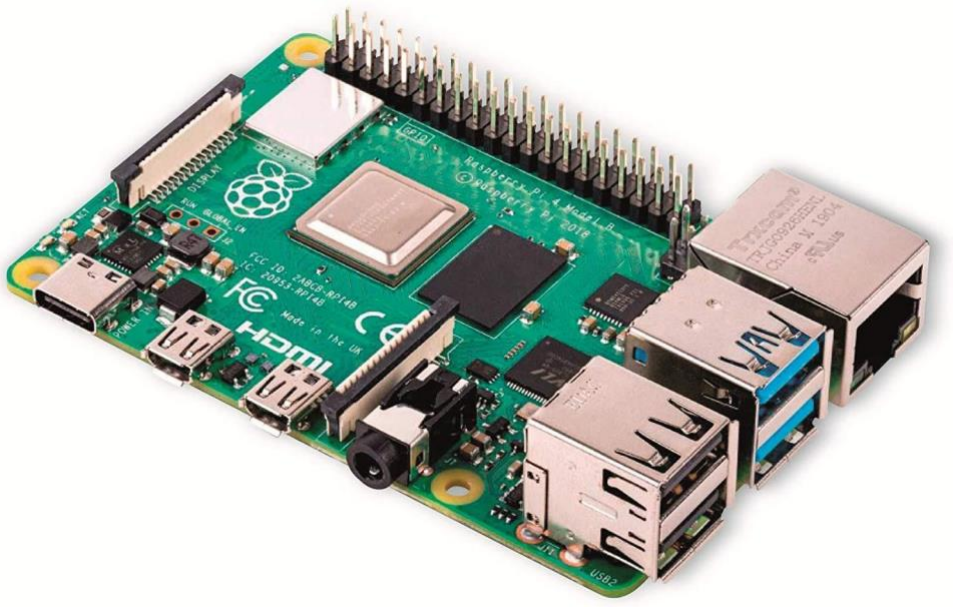

Figura 1. Raspberry PI model 4B

Il dispositivo può comunicare con altre periferiche attraverso il GPIO. Un GPIO (GENERAL PURPOSE INPUT-OUTPUT) è un'interfaccia che consente di collegare fisicamente il Raspberry ad altri devices. È costituito nel caso in questione da 40 pin esterni che, oltre agli ingressi e alle uscite digitali, possono essere commutati anche a funzioni alternative (AF) fornite da periferiche interne. La selezione delle funzioni per i pin GPIO è gestita attraverso una matrice di routing configurabile tramite i registri. Precisamente, è stato utilizzato il registro GPFSEL0 per assegnare ai pin GPIO2 e GPIO3 l'alternate function adatta per il bus I2C.

FTDI FT232RL

Lo FTDI FT232RL [2] è un'interfaccia seriale da mini USB a UART. Lo Universal Asynchronous Receiver-Transmitter (UART) è un dispositivo che ha lo scopo di trasformare flussi di bit da formato seriale asincrono a parallelo e viceversa. Una variante del UART è lo USART che è in grado di gestire comunicazioni seriali sincrone. Gli UART sono gestiti tramite periferiche interne. Sostanzialmente, tutti i sistemi embedded includono almeno una periferica tra UART e USART, soprattutto la prima. Permettono l'utilizzo di diversi protocolli di trasmissione grazie alla loro versatilità, e comprendono efficaci servizi come il controllo di flusso e la correzioni degli errori.

Figura 2. FTDI FT232RL

Le specifiche elettriche possono variare ed è possibile collegare due dispositivi solo se richiedono le stesse specifiche. UART e USART sono stati inclusi nei PC fino all'avvento di USB. Ora, sono necessari adattatori (ad es. dispositivi USB) per collegare un PC a sistemi forniti da UART.

TCS34725

Il TCS34725 [3] è un sensore che fornisce un ritorno digitale dei valori di rilevamento della luce rossa, verde, blu (RGB) e chiara. L'alta sensibilità, ampia gamma dinamica e filtro IR rendono il TCS34725 un sensore di colore adatto a condizioni di illuminazione variabili. Un filtro IR integrato sul chip riduce al minimo la componente spettrale IR della luce in entrata e consente di eseguire misurazioni del colore in modo accurato. Il sensore di colore TCS34725 ha una vasta gamma di applicazioni tra cui controllo della retroilluminazione a LED RGB, illuminazione a stato solido, controlli di processo industriali e apparecchiature diagnostiche mediche. Inoltre, il filtro di blocco IR consente al TCS34725 di eseguire il rilevamento della luce ambientale (ALS).

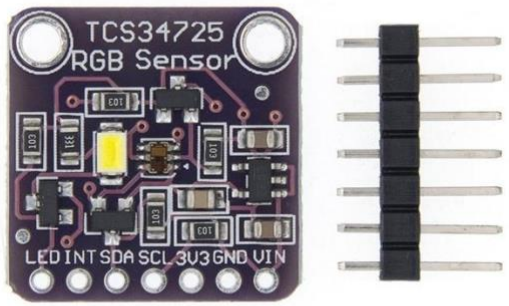

Figura 3. TCS34725

Il rilevamento della luce ambientale è ampiamente utilizzato nei prodotti basati su display come telefoni cellulari, notebook e TV per rilevare l'ambiente di illuminazione e consentire la luminosità automatica del display per una visione ottimale e un risparmio energetico. La comunicazione dei dati TCS34725 avviene tramite un bus seriale I2C che arriva fino a 400 kHz. Lo standard industriale I2C facilita la connessione diretta a microcontrollori e processori integrati. Oltre al bus I2C, il TCS34725 fornisce un'uscita del segnale di interrupt separata. Quando gli interrupt sono abilitati e vengono superate le soglie definite dall'utente, viene generato un interrupt che manterrà il sensore in stato di sleeping fino a quando non viene cancellato dal controller. Questa funzione semplifica e migliora l'efficienza del software di sistema eliminando la necessità di eseguire il polling del TCS34725.

ADDRESS	RESISTER NAME	R/W	REGISTER FUNCTION	RESET VALUE
--	COMMAND	W	Specifies register address	0x00
0x00	ENABLE	R/W	Enables states and interrupts	0x00
0x01	ATIME	R/W	RGBC time	0xFF
0x03	WTIME	R/W	Wait time	0xFF
0x04	AILTL	R/W	Clear interrupt low threshold low byte	0x00
0x05	AILTH	R/W	Clear interrupt low threshold high byte	0x00
0x06	AIHTL	R/W	Clear interrupt high threshold low byte	0x00
0x07	AIHTH	R/W	Clear interrupt high threshold high byte	0x00
0x0C	PERS	R/W	Interrupt persistence filter	0x00
0x0D	CONFIG	R/W	Configuration	0x00
0x0F	CONTROL	R/W	Control	0x00
0x12	ID	R	Device ID	ID
0x13	STATUS	R	Device status	0x00
0x14	CDATAL	R	Clear data low byte	0x00
0x15	CDATAH	R	Clear data high byte	0x00
0x16	RDATAL	R	Red data low byte	0x00
0x17	RDATAH	R	Red data high byte	0x00
0x18	GDATAL	R	Green data low byte	0x00
0x19	GDATAH	R	Green data high byte	0x00
0x1A	BDATAL	R	Blue data low byte	0x00
0x1B	BDATAH	R	Blue data high byte	0x00

Figura 4. Registri TCS34725

Il dispositivo è controllato attraverso registri dati e da un command register accessibili tramite interfaccia I2C e riportati nella fig. 4. Questi registri possono essere letti e modificati per determinare i risultati delle conversioni ADC.

I2C

Il bus I2C è un bus molto popolare e potente utilizzato per la comunicazione tra un master e uno o più dispositivi slave. La sua diffusione deriva soprattutto dal fatto che il master (nel nostro caso il Raspberry) può controllare più dispositivi periferici semplicemente attraverso due soli pin. Ogni dispositivo slave sul bus I2C ha un indirizzo, ovvero lo slave address (o device address) specifico per distinguersi da altri dispositivi che si trovano sullo stesso bus I2C. L'interfaccia fisica I2C, infatti, è costituita dalle linee serial clock (SCL) e serial data (SDA). Affinché una comunicazione tra un master e uno slave possa avvenire, innanzitutto lo slave deve essere indirizzato dal master ed inoltre il bus deve essere inattivo, cioè entrambe le linee SDA e SCL devono essere alte dopo una condizione di STOP [4].

Ecco riportati i passi che il master deve seguire per accedere al dispositivo slave.
Supponiamo che un master voglia inviare dati a uno slave:
• Il trasmettitore principale invia una condizione di START e si rivolge al ricevitore slave;
• Il trasmettitore principale invia i dati al ricevitore slave;
• Il trasmettitore master termina il trasferimento con una condizione di STOP.
Se un master desidera leggere dati da uno slave:
• Il ricevitore master invia una condizione di START e si rivolge al trasmettitore slave;
• Il ricevitore master invia il registro richiesto da leggere al trasmettitore slave;
• Il ricevitore master riceve i dati dal trasmettitore slave;
• Il master-ricevitore termina il trasferimento con una condizione di STOP.
Una transizione da high a low sulla linea SDA mentre SCL è elevato definisce una condizione di START. Una transizione da low ad high sulla linea SDA mentre SCL è alto definisce uno condizione di STOP. Una condizione REPETED START è simile a una condizione START e viene utilizzata al posto di uno STOP. Differisce dalla semplice condizione di START perché accade prima di una condizione di STOP (quando il bus non è inattivo). Questo è utile per quando il master desidera iniziare una nuova comunicazione, ma non desidera lasciare il bus inattivo con la condizione di STOP, e dunque perdere il controllo del bus.
Un bit di dati viene trasferito durante ciascun impulso di clock dell'SCL e ogni byte di dati è seguito da un bit di ACK mandato dal ricevitore. Qualsiasi numero di byte di dati può essere trasferito tra le condizioni START e STOP a condizione che i dati sulla linea SDA rimangano stabili durante la fase high del periodo di clock, poiché i cambiamenti nella linea di dati quando SCL è alto sono interpretati come comandi di controllo (START o STOP).
Prima che il ricevitore possa inviare un ACK, il trasmettitore deve rilasciare la linea SDA. Per inviare un bit ACK, il ricevitore deve abbattere la linea SDA durante la fase low del periodo di clock relativo a ACK / NACK in modo che la linea SDA sia stabile in low durante la fase high del periodo di clock relativo a ACK / NACK.
Quando la linea SDA rimane high durante il periodo di clock relativo a ACK / NACK, questa viene interpretata come NACK, inviato ad esempio quando il ricevitore non è in grado di ricevere o trasmettere perché sta eseguendo alcune funzioni, o riceve dati che non comprende, o non può ricevere più byte di dati.
La comunicazione, sostanzialmente avviene leggendo dai registri o scrivendo nei registri dello slave. Bisogna tenere presente che non tutti i dispositivi slave hanno registri. Alcuni dispositivi sono semplici e contengono solo 1 registro, che può essere scritto direttamente inviando i dati immediatamente dopo l'indirizzo slave, anziché indirizzare un registro.

Per scrivere sul bus I2C, il master invierà una condizione di avvio sul bus, l'indirizzo dello slave a 7 bit e un ultimo bit (il bit R / W) impostato su 0, che indica una scrittura. Dopo che lo slave ha inviato il bit di conferma, il master invierà quindi l'indirizzo del registro in cui desidera scrivere. Lo slave ritrasmetterà un ACK, il master inizierà a inviare i dati allo slave e terminerà la trasmissione con una condizione di STOP.
Nella fig. 5 viene riportato un esempio di scrittura di un solo byte di dati.

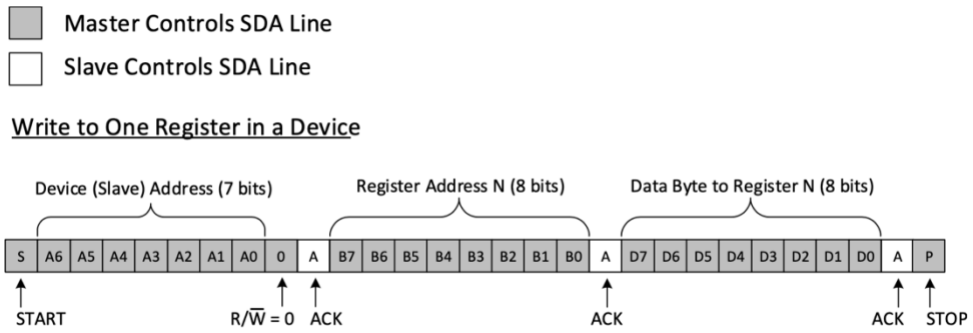

Figura 5. Esempio di scrittura di un byte di dati

Leggere da uno slave è molto simile alla scrittura, ma con alcuni passaggi aggiuntivi. Per leggere da uno slave, il master deve prima istruire lo slave da quale registro desidera leggere. Il master inizializza la trasmissione in modo simile alla scrittura, inviando lo slave address e il bit R / W uguale a 0, seguito dall'indirizzo del registro da cui desidera leggere. Ricevuto l'ACK, il master invierà nuovamente una condizione di START, seguita dallo slave address e il bit R/W impostato su 1 (che indica una lettura).

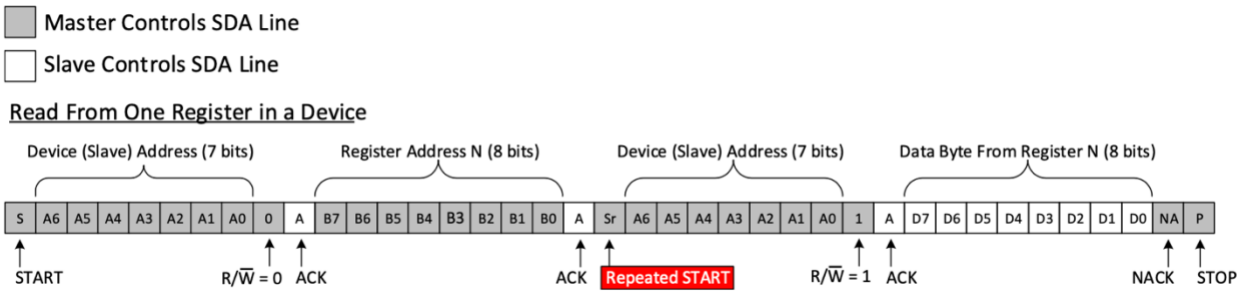

Figura 6. Esempio di lettura di un byte di dati

In seguito, il master continuerà a inviare gli impulsi di clock, ma rilascerà la linea SDA, in modo che lo slave possa trasmettere dati. Alla fine di ogni byte di dati, il master invierà un ACK allo slave. Una volta che il master ha ricevuto il numero di byte che si aspetta, invierà un NACK, segnalando allo slave di interrompere le comunicazioni e rilasciare il bus e una condizione di STOP.

BSC

Il controller Broadcom Serial Controller (BSC) è un master I2C che comprende otto registri a 32 bit [1]. All'interno della BCM esistono tre master BSC con indirizzo di base differente dipendente dal dispositivo. Per il Raspberry PI 4B i tre indirizzi di base sono i seguenti:

- BSC0: 0xFE20_5000;
- BSC1: 0xFE80_4000;
- BSC2: 0xFE80_5000;

Tutti i registri avranno lo stesso offset rispetto all'indirizzo base. La fig. 7 mostra gli offset con i relativi registri.

I2C Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	C	Control	32
0x4	S	Status	32
0x8	DLEN	Data Length	32
0xc	A	Slave Address	32
0x10	FIFO	Data FIFO	32
0x14	DIV	Clock Divider	32
0x18	DEL	Data Delay	32
0x1c	CLKT	Clock Stretch Timeout	32

Figura 7. Registri BSC

Il Control register è il registro principale che viene utilizzato per abilitare gli interrupt, abilitare il controller BSC (bit I2CEN) definire una lettura o una scrittura (bit READ), avviare un trasferimento (bit ST) e pulire la FIFO (bit CLEAR).

Lo Status register viene utilizzato per registrare lo stato dell'attività (bit TA che indica se il trasferimento è in corso, bit DONE che indica se il trasferimento è stato completato), gli aknowledgement (bit ERR), gli interrupts, e lo stato della FIFO per verificare ad esempio se è piena o vuota (read only bits RXF, TXE, RXD, TXD, RXR, TXW).

Il Data Length register definisce il numero di byte di dati da trasmettere o ricevere. La lettura del registro durante la trasmissione restituisce il numero di byte ancora da mandare.

Leggendo il campo DLEN quando TA = 0 e DONE = 0 restituisce l'ultimo valore scritto.

Lo Slave Address register specifica l'indirizzo dello slave nei 7 bit meno significativi del registro (field ADDR). Da notare che il master BSC supporta anche l'indirizzamento a 10 bit.
Il Data FIFO register è usato per accedere alla FIFO. Il campo DATA che comprende gli 8 bit meno significativi specifica i dati da trasmettere o ricevere. Le scritture di dati in una FIFO piena verranno ignorate e le letture da una FIFO vuota saranno invalide.
Il Clock Divider register viene utilizzato per definire la velocità di clock della periferica BSC.
Il Data Delay register fornisce un controllo accurato del punto di campionamento dei dati.
Il Clock Stretch Timeout register specifica il numero di cicli di clock da attendere dopo il fronte di salita di SCL prima di decidere che lo slave non risponde.

Configurazione

Descritte le specifiche del progetto e l'hardware necessario, sono riportate di seguito le fasi fondamentali per la realizzazione del sistema.
Il primo passo è quello di collegare tutte le componenti hardware utilizzate. Si è deciso di adottare una breadboard per questioni di praticità. Quindi è stato collegato il Raspberry, tramite un Ribbon Cable, ad un GPIO Extension Board e a sua volta quest'ultimo collegato alla breadboard. Ad essa sono stati attaccati i jumpers necessari a collegare il sensore TCS34725 e lo UART. In particolare il sensore comprende 7 pin, ovvero LED, INT, SDA, SCL, 3V3, GND e VIN. Il pin INT non è stato collegato perché la realizzazione di questo progetto non necessita di interrupts per il suo funzionamento. Gli altri pin invece sono stati collegati nel modo seguente:

- il pin LED è stato collegato al GPIO 17
- il pin SDA è stato collegato al GPIO 2, corrispondente alla linea SDA del BSC1
- il pin SCL è stato collegato al GPIO 3, corrispondente alla linea SCL del BSC1
- il pin 3v3 è stato collegato al pin 1, che serve da alimentazione a 3,3V
- il pin GND è stato collegato al pin 9 corrispondente al ground
- il pin VIN è stato collegato al pin 17, che serve da alimentazione a 3,3V

Per quanto riguarda lo UART è stato invece collegato nel seguente modo:

- il pin GND è stato collegato al pin 6 corrispondente al ground
- il pin TXD è stato collegato al GPIO 14, corrispondente alla linea TXD0
- il pin RXD è stato collegato al GPIO 15, corrispondente alla linea RXD0

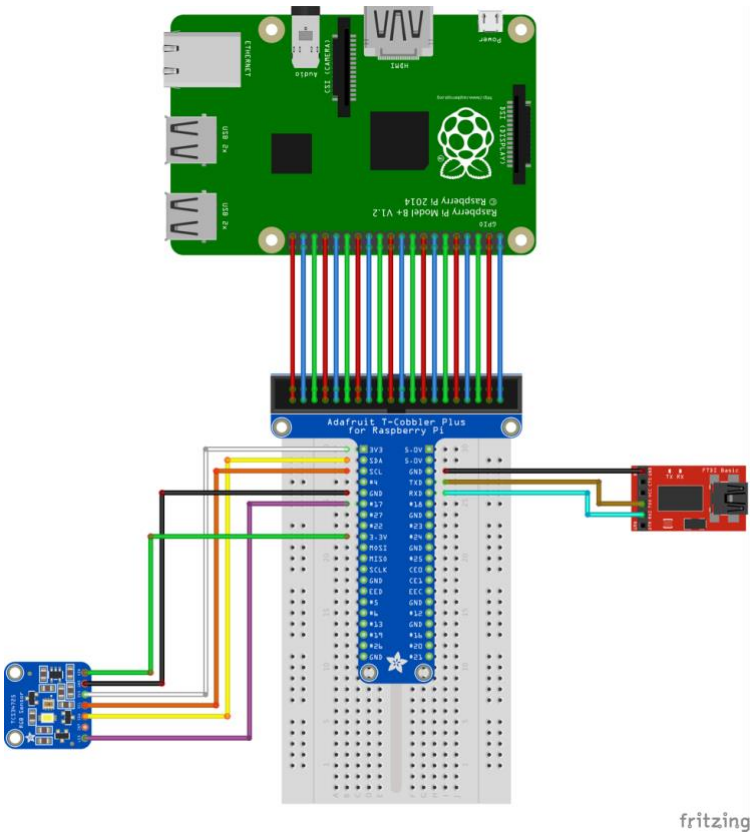

Figura 8. Modello del sistema

Per verificare la corretta configurazione, si può utilizzare un semplice tool presente nel sistema operativo Raspbian, ovvero i2cdetect [5]. Raspbian [6] è un sistema operativo gratuito basato su Debian, ottimizzato per l'hardware Raspberry Pi, raccomandato per l'uso normale su un Raspberry Pi.

Per potere utilizzare questo tool è necessario installare il sistema operativo su una microSD da usare come storage e lanciare il comando “sudo i2cdetect -y 1”. Per lanciare il comando senza dover collegare tastiera, mouse e schermo al Raspberry, si può anche utilizzare una connessione SSH. Per abilitare quest’ultima basta inserire all’interno della microSD un file vuoto chiamato ssh. Una volta collegato il Raspberry al router di casa, il DHCP gli fornirà un indirizzo IP, che sarà possibile visualizzare dalla pagina di configurazione del router. Basterà poi lanciare dal PC, connesso alla rete in cui è presente il Raspberry, il comando “ssh pi@192.168.1.xxx” e usare come password “raspberry”. In questo modo si ha accesso alla shell e sarà possibile lanciare il comando “sudo i2cdetect -y 1”.

Verrà mostrata una schermata simile alla seguente:

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Figura 9. Esecuzione di i2cdetect

Grazie a questo tool è possibile vedere tutti i dispositivi I2C collegati al Raspberry e il relativo indirizzo, che per il sensore TCS34725 è 0x29.
Una volta accertatosi che il sensore sia collegato correttamente, si può procedere con la configurazione dello UART. In particolare sono necessari due tool, Picocom e Minicom.

Software

Minicom

Minicom è un software di emulazione da terminale per sistemi operativi Unix-based. È comunemente usato quando si configura una console seriale remota [7].

Picocom

Picocom è, in linea di principio, molto simile al minicom. È stato progettato come uno strumento semplice, manuale, di configurazione del modem, di test e di debug [8].
In questo contesto, vengono utilizzati come strumento di comunicazione con la porta seriale.
Il comando utilizzato per avviare Picocom è :

- picocom --b 115200 /dev/ttyUSB0 --imap delbs -s "ascii-xfr -sv -l100 -c10" se si usa Linux
- picocom --b 115200 /dev/cu.usbserial-00000000 --imap delbs -s "ascii-xfr -sv -l100 -c10" se si usa macOS

Per utilizzare Minicom è invece necessario premere la combinazione Ctrl+A+S e inserire il path del file contenente le istruzioni da far eseguire al Raspberry. A tal proposito, per lo sviluppo di questo sistema, è stato utilizzato un ambiente interattivo di basso livello, ovvero pijFORTHos.

pijFORTHos

pijFORTHos è un interprete FORTH bare metal per il Raspberry Pi, basato su Jonesforth-ARM, perfetto per macchine con risorse limitate e incapaci di sostenere il peso di un sistema operativo [9].

Applicazione

L'applicazione realizzata ha lo scopo di riportare su un monitor i valori di una vasta gamma di colori e di rilevare questi ultimi attraverso il TCS34725 per evidenziare le differenze tra le intensità mostrate sul display e quelle individuate dal sensore. Entrando nel dettaglio, l'idea è quella di mostrare su una sezione del monitor i colori delle scale cromatiche di ogni componente RGB (quindi scala del rosso, scala del verde e scala del blu), in ordine di intensità crescente. Il sensore, puntato verso lo schermo, dovrà recepire i valori di intensità presentati e questi saranno riportati su dei grafici specifici. In particolare, saranno utilizzate delle tabelle di look-up (LUT).
Nella realizzazione del progetto saranno raffigurati sul display precisamente 3 LUT, una per ogni scala cromatica di colore; quindi, la prima mostrerà le differenze tra le intensità di rosso visualizzate sullo schermo e quelle rilevate dal TCS34725, la seconda mostrerà le differenze tra le intensità di verde e la terza quelle del blu. Un'altra specifica dell'applicazione consiste nel far riconoscere al sensore TCS34725 alcuni oggetti attraverso l'individuazione del colore che tali oggetti presentano, classificandoli in rosso, verde o blu. Il sistema comprenderà il conteggio degli oggetti dello stesso colore che dovrà rilevare e la visualizzazione sullo schermo delle occorrenze riscontrate.
Tutte le operazioni effettuate sul display, quindi la visualizzazione della finestra colorata, dei grafici, delle occorrenze degli oggetti dello stesso colore e di specifiche informazioni inerenti all'esecuzione dell'applicazione sono state possibili grazie alla gestione del display collegato all'interfaccia HDMI da parte del Raspberry.
Il display è visto come una matrice di pixel, gestiti tramite un array monodimensionale, il framebuffer, una regione di memoria condivisa da VC e CPU, che contiene valori di tipo colordepth. Un pixel è composto da due coordinate x e y. L'ascissa ha un valore compreso tra 0 e larghezza-1, mentre l'ordinata assume valore nell'intervallo tra 0 e altezza-1. Il pixel (0,0) è quello in alto a sinistra, il pixel (larghezza-1,0) è quello in alto a destra, il pixel (0,altezza-1) è quello in basso a sinistra e il pixel (larghezza-1, altezza-1) è quello in basso a destra. Ogni pixel è rappresentato da un indirizzo di memoria a 32 bit il cui contenuto indica il valore ARGB del colore del pixel. I tre byte meno significativi sono utilizzati per indicare le componenti rosso, verde e blu mentre il più significativo indica la trasparenza.

Codifica RGB

La codifica RGB [10] (Red, Green, Blue), elaborata nel 1931 dalla Commissione Internazionale dell'Illuminazione (CIE) consiste nel rappresentare lo spazio dei colori partendo da tre raggi monocromatici di colori: Rosso (di lunghezza d'onda uguale a 700,0 nm), Verde (di lunghezza d'onda uguale a 546,1 nm) e Blu (di lunghezza d'onda uguale a 435,8 nm). Questo spazio di colore corrisponde al modo in cui i colori sono solitamente codificati "informaticamente" o, più esattamente, al modo in cui gli schermi dei computer rappresentano i colori. Così, il modello RGB propone di codificare su un byte ogni componente di colore, il che corrisponde a 256 intensità di rosso, 256 intensità di verde e 256 intensità di blu, ossia 16.777.216 possibilità teoriche di colori diversi, cioè più di quelli distinguibili dall'occhio umano (circa 2 milioni). Tuttavia, questo valore non è che teorico dato che dipende fortemente dall'hardware di visualizzazione.

Tabelle di Look-Up (LUT)

Nell'ambito dell'elaborazione delle immagini, le tabelle di look-up, meglio note come LUT [11], sono lo strumento principale utilizzato per le trasformazioni puntuali omogenee, ovvero elaborazioni che dipendono solamente dai valori di intensità dei pixel e non dalla loro posizione. Alcune tipiche elaborazioni puntuali sono:

- miglioramento della luminosità, aggiungendo o sottraendo un valore costante a tutti i pixel dell'immagine (per compensare la sotto o la sovraesposizione);
- negativo dell'immagine (inversione della scala di grigi);
- correzione gamma;
- miglioramento del contrasto;
- elaborazione dell'istogramma;
- elaborazione di pseudocolori;
- bilanciamento del colore.

Questo tipo di trasformazioni sono di fatto dei mapping uno a uno e sono effettuate appunto con l'ausilio delle LUT, cioè array monodimensionali che mappano ogni valore di input con il corrispettivo valore di output. Le LUT possono dunque essere rappresentate come un grafico bidimensionale in cui sull'asse delle ascisse vengono riportati i valori di input, e sull'ordinata i relativi valori di output.

Programmazione FORTH

Il software è stato scritto su più file, in modo tale da avere una visione più strutturata e comprensibile di esso; ogni file contiene istruzioni e definizioni inerenti alla stessa funzionalità.
La prima parte della stesura del codice ha riguardato l'interfaccia I2C. In particolare, dovendo utilizzare un ambiente di "basso livello" come pijFORTHos, e non avendo a disposizione né un sistema operativo a supporto, né librerie già scritte da altri sviluppatori, si è resa necessaria una prima fase di scrittura delle words di base per potersi interfacciare ad I2C. Prima di tutto, è stato necessario settare l'ALT0 per i GPIO2 e GPIO3, quindi impostando i relativi bit del GPSEL0 a 100, per abilitare le linee SCL e SDA del BSC. A tal fine sono state consultate le seguenti tabelle:

29-27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO Pin 9 takes alternate function 3 011 = GPIO Pin 9 takes alternate function 4 010 = GPIO Pin 9 takes alternate function 5	R/W	0
26-24	FSEL8	FSEL8 - Function Select 8	R/W	0
23-21	FSEL7	FSEL7 - Function Select 7	R/W	0
20-18	FSEL6	FSEL6 - Function Select 6	R/W	0
17-15	FSEL5	FSEL5 - Function Select 5	R/W	0
14-12	FSEL4	FSEL4 - Function Select 4	R/W	0
11-9	FSEL3	FSEL3 - Function Select 3	R/W	0
8-6	FSEL2	FSEL2 - Function Select 2	R/W	0
5-3	FSEL1	FSEL1 - Function Select 1	R/W	0
2-0	FSEL0	FSEL0 - Function Select 0	R/W	0

Figura 10. GPIO Alternate Functions

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High	SDA0	SA5	<reserved>			
GPIO1	High	SCL0	SA4	<reserved>			
GPIO2	High	SDA1	SA3	<reserved>			
GPIO3	High	SCL1	SA2	<reserved>			
GPIO4	High	GPCLK0	SA1	<reserved>			ARM_TDI
GPIO5	High	GPCLK1	SA0	<reserved>			ARM_TDO
GPIO6	High	GPCLK2	SOE_N / SE	<reserved>			ARM_RTCK
GPIO7	High	SPI0_CE1_N	SWE_N / SRW_N	<reserved>			
GPIO8	High	SPI0_CE0_N	SD0	<reserved>			
GPIO9	Low	SPI0_MISO	SD1	<reserved>			
GPIO10	Low	SPI0_MOSI	SD2	<reserved>			
GPIO11	Low	SPI0_SCLK	SD3	<reserved>			
GPIO12	Low	PWM0	SD4	<reserved>			ARM_TMS
GPIO13	Low	PWM1	SD5	<reserved>			ARM_TCK
GPIO14	Low	TXD0	SD6	<reserved>			TXD1
GPIO15	Low	RXD0	SD7	<reserved>			RXD1
GPIO16	Low	<reserved>	SD8	<reserved>	CTS0	SPI1_CE2_N	CTS1
GPIO17	Low	<reserved>	SD9	<reserved>	RTS0	SPI1_CE1_N	RTS1
GPIO18	Low	PCM_CLK	SD10	<reserved>	BSCSL SDA / MOSI	SPI1_CE0_N	PWM0
GPIO19	Low	PCM_FS	SD11	<reserved>	BSCSL SCL / SCLK	SPI1_MISO	PWM1
GPIO20	Low	PCM_DIN	SD12	<reserved>	BSCSL / MISO	SPI1_MOSI	GPCLK0
GPIO21	Low	PCM_DOUT	SD13	<reserved>	BSCSL / CE_N	SPI1_SCLK	GPCLK1
GPIO22	Low	<reserved>	SD14	<reserved>	SD1_CLK	ARM_TRST	
GPIO23	Low	<reserved>	SD15	<reserved>	SD1_CMD	ARM_RTCK	
GPIO24	Low	<reserved>	SD16	<reserved>	SD1_DAT0	ARM_TDO	
GPIO25	Low	<reserved>	SD17	<reserved>	SD1_DAT1	ARM_TCK	
GPIO26	Low	<reserved>	<reserved>	<reserved>	SD1_DAT2	ARM_TDI	
GPIO27	Low	<reserved>	<reserved>	<reserved>	SD1_DAT3	ARM_TMS	
GPIO28	-	SDA0	SA5	PCM_CLK	<reserved>		
GPIO29	-	SCL0	SA4	PCM_FS	<reserved>		
GPIO30	Low	<reserved>	SA3	PCM_DIN	CTS0		CTS1
GPIO31	Low	<reserved>	SA2	PCM_DOUT	RTS0		RTS1
GPIO32	Low	GPCLK0	SA1	<reserved>	TXD0		TXD1
GPIO33	Low	<reserved>	SA0	<reserved>	RXD0		RXD1
GPIO34	High	GPCLK0	SOE_N / SE	<reserved>	<reserved>		
GPIO35	High	SPI0_CE1_N	SWE_N / SRW_N		<reserved>		
GPIO36	High	SPI0_CE0_N	SD0	TXD0	<reserved>		
GPIO37	Low	SPI0_MISO	SD1	RXD0	<reserved>		
GPIO38	Low	SPI0_MOSI	SD2	RTS0	<reserved>		
GPIO39	Low	SPI0_SCLK	SD3	CTS0	<reserved>		
GPIO40	Low	PWM0	SD4		<reserved>	SPI2_MISO	TXD1
	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5

Figura 11. Alternate Function assignments

Successivamente, una volta abilitato il bus I2C, sono state scritte le words WRITE e READ. Per potere definire queste due word, è stato creato un HAL (Hardware Abstraction Layer) che ha lo scopo di fornire una struttura di base su cui poter costruire un'applicazione più complessa. Una volta definite le words di base, il lavoro si è incentrato sulla stesura delle words specifiche per il sensore utilizzato. La periferica contiene 21 registri, e prevede l'utilizzo di questi per l'accensione, la configurazione e le comunicazioni con la stessa. In particolare, nel registro ENABLE sono stati settati il bit PON per attivare l'oscillatore interno e consentire il funzionamento dei timer e dei canali ADC, e il bit AEN per attivare i due canali ADC. Da precisare il fatto che è stato aggiunto un delay di 3 millisecondi tra l'evento di attivazione dell'oscillatore e l'attivazione dei canali ADC, poiché senza di esso le due operazioni eseguite immediatamente causavano un'errata configurazione del dispositivo.

	7	6	5	4	3	2	1	0	
ENABLE	Reserved			AIEN	WEN	Reserved	AEN	PON	Address 0x00

FIELD	BITS	DESCRIPTION
Reserved	7:5	Reserved. Write as 0.
AIEN	4	RGBC interrupt enable. When asserted, permits RGBC interrupts to be generated.
WEN	3	Wait enable. This bit activates the wait feature. Writing a 1 activates the wait timer. Writing a 0 disables the wait timer.
Reserved	2	Reserved. Write as 0.
AEN	1	RGBC enable. This bit actives the two-channel ADC. Writing a 1 activates the RGBC. Writing a 0 disables the RGBC.
PON [1, 2]	0	Power ON. This bit activates the internal oscillator to permit the timers and ADC channels to operate. Writing a 1 activates the oscillator. Writing a 0 disables the oscillator.

Figura 12. Enable register

In seguito, dopo un'attenta analisi della documentazione tecnica, ciò che è venuto fuori è che il sensore TCS34725 segue un Bayer pattern [12] molto simile al RGBW, e restituisce oltre alle componenti RGB, anche una componente di Clear (RGBC). Il segnale analogico catturato dal sensore viene trasformato in digitale da circuiti ADC (analog-to-digital converters); in particolare sono generati 16 bit per i dati relativi a ciascun canale. L'integrato è fornito anche di una circuiteria che riguarda la gestione degli interrupts, che però non sono stati presi in considerazione in quanto non necessari per l'applicazione in questione. Il dispositivo dispone di otto registri a 8 bit (fig. 13), due per ogni canale, che al momento della rilevazione del colore, vengono settati con il valore dell'intensità della relativa componente di colore in questione.

REGISTER	ADDRESS	BITS	DESCRIPTION
CDATA	0x14	7:0	Clear data low byte
CDATAH	0x15	7:0	Clear data high byte
RDATA	0x16	7:0	Red data low byte
RDATAH	0x17	7:0	Red data high byte
GDATA	0x18	7:0	Green data low byte
GDATAH	0x19	7:0	Green data high byte
BDATA	0x1A	7:0	Blue data low byte
BDATAH	0x1B	7:0	Blue data high byte

Figura 13. Channel Data Register

Per poterne leggere il contenuto, il sensore richiede che sia effettuata una read da 2 bytes sul registro dedicato ai bit meno significativi di ogni canale. Questi dati devono però essere interpretati correttamente per poter ricavare le componenti RGB; a tal scopo è stato utilizzato un algoritmo di normalizzazione dei dati.

Algoritmo di normalizzazione

L'algoritmo di normalizzazione adottato è stato scelto dopo una serie di problematiche riscontrate per l'interpretazione dei valori contenuti nei Channel Data Registers. Dopo un'intensa e dettagliata consultazione con librerie open-source [13] [14] e una lunga attività di testing, è stato utilizzato l'algoritmo in seguito presentato in quanto è stato ritenuto il più preciso e più consono tra quelli analizzati. La procedura è la seguente:

- Viene controllato se il valore di clear è 0. In tal caso gli altri canali assumeranno valore 0 senza ulteriori elaborazioni.
- Le componenti rosse, verdi e blu sono state moltiplicate per 255.
- Il valore dei canali è stato diviso per il valore contenuto nel registro di clear
- È stato calcolato il minimo tra i valori dei canali rosso, verde e blu ricavati e memorizzato in una variabile MIN.
- Infine, a ciascuna delle componenti RGB ricavate è stata sottratta la variabile MIN, ricavata al passo precedente.

Per effettuare una rilevazione, sono state create delle words per mostrare sullo schermo il colore da far riscontrare al sensore, e il colore rilevato. Per avere una visione più chiara della procedura è di seguito riportata una figura rappresentativa dell'output del display. Nella parte bassa dello schermo, sono presenti le tre tabelle di look-up su cui dovrebbero essere localizzati i punti (non riportati in figura) indicanti i livelli di intensità rilevati dal sensore; nella parte alta invece si possono osservare due grandi finestre colorate: quella a sinistra indica il valore di intensità di input, quindi il colore che l'integrato dovrebbe rilevare, mentre nella finestra a destra è visualizzato il livello di intensità che il dispositivo ha effettivamente riscontrato.

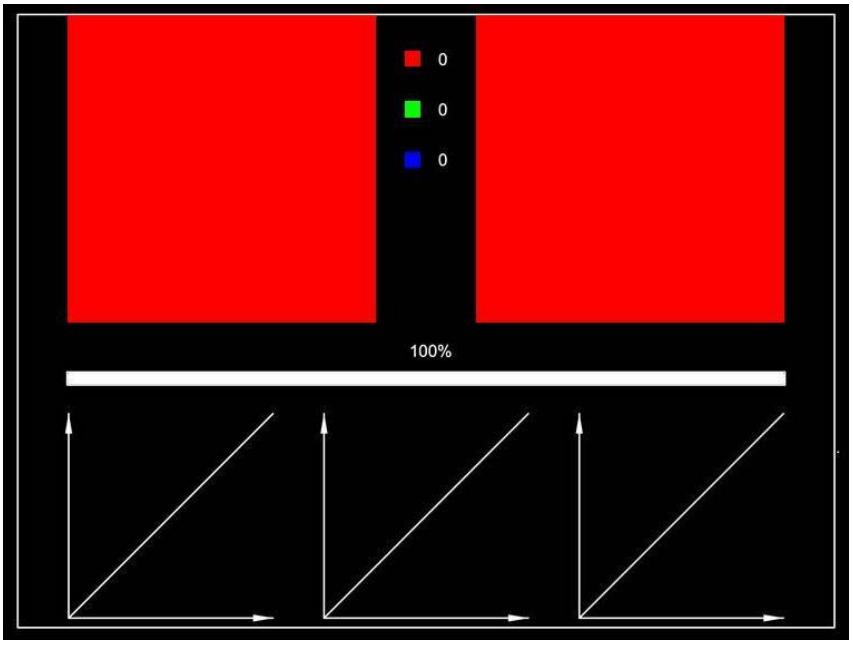

Figura 14. Mockup dell'output

La rilevazione è effettuata tramite due word, la prima (CONFIGURATION) serve ad inizializzare il sensore, mentre la seconda (START) avvia di fatto la rilevazione. Oltre alla creazione delle LUT, l'applicazione produce anche la visualizzazione sul terminale del mapping tra il valore esadecimale delle componenti in input e quelle in output. La figura seguente mostra un esempio:

RED	
30	20
38	2C
40	3A
48	48
50	57
58	62
60	6A
68	72
70	78
78	7F
80	83
88	88
90	8D
98	91
A0	95

GREEN	
30	33
38	3B
40	41
48	46
50	4A
58	4D
60	50
68	53
70	56
78	58
80	5A
88	5B
90	5C
98	5D
A0	5E

BLUE	
30	10
38	20
40	2A
48	30
50	38
58	49
60	5D
68	6B
70	76
78	80
80	88
88	8F
90	96
98	9B
A0	9B

Figura 15. Esempio output terminale

Per quando riguarda la specifica che prevede il conteggio delle occorrenze degli oggetti che presentano lo stesso colore, è stata semplicemente utilizzata la finestra a sinistra contenente il colore da rilevare, e il conteggio è stato implementato attraverso un controllo effettuato sulla massima componente tra le tre rilevate e, quindi, l'aggiornamento della relativa variabile rappresentante il contatore. Per visualizzare lo stato delle variabili è mostrata a schermo una legenda contenente i colori riconosciuti e i valori dei contatori associati (fig. 15). I numeri stampati per rappresentare questi ultimi sono stati realizzati pixel per pixel con delle words, una specifica per ogni numero. Allo stesso modo è stato ottenuto il simbolo "%" impiegato nella barra di completamento dell'applicazione situata al centro del display.

Conclusioni

Lo sviluppo dell'applicazione ha comportato una procedura di interpretazione delle caratteristiche funzionali del sensore tutt'altro che banale, in quanto la documentazione tecnica a disposizione non è risultata esaustiva come ci si aspettava a causa delle poche informazioni contenute. Dunque, sono state necessarie delle ricerche online e dei test direttamente sull'hardware. Inoltre, non è stato facile interpretare i valori restituiti dal dispositivo principalmente a causa di due motivi:

- l'output del sensore è molto influenzato dalla distanza dal colore rilevato, quindi dallo schermo, e quindi si discosta abbastanza in esperimenti ripetuti con distanza variabile. È stata scelta a tal proposito una distanza fissa, di 3 centimetri circa.
- i valori contenuti nei CHANNEL_DATA_REGISTERS necessitano di una normalizzazione prima di potere essere rappresentativi della corretta componente RGB. Di fatto, sulla documentazione non è riportata alcuna indicazione sull'operazione in questione e l'algoritmo scelto è stato frutto di consultazione di librerie online open-source.

Tuttavia, le difficoltà incontrate nel corso dello sviluppo dell'applicazione sono state superate, tutte le specifiche sono state correttamente rispettate e tutte le funzionalità sono state implementate come previsto dal progetto. Sono stati ottimizzati i risultati ottenuti dal sensore di colore, nel limite delle possibilità del sistema.

Possibili migliorie

Ciò non esclude il fatto che il sistema potrebbe sicuramente essere migliorato ad esempio trovando un metodo più efficiente e più preciso per la normalizzazione dei dati restituiti dal sensore o anche l'utilizzo di un dispositivo più sensibile e più appropriato allo scopo. Imposto ciò, sarebbe anche possibile e sensata un'implementazione della reale calibrazione dello schermo, operazione risultata infattibile nel sistema sviluppato a causa dell'imprecisione dell'output del sensore di colore.

Bibliografia

[1] Broadcom Corporation, 06 February 2012. [Online]. Available: https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf.

[2] Future Technology Devices International Limited, «FT232R usb uart ic datasheet» [Online]. Available: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

[3] TAOS, August 2012. [Online]. Available: http://www.adafruit.com/datasheets/TCS34725.pdf.

[4] J. Valdez e J. Becker, «Understanding the I2C Bus,» Texas Instruments, June 2015. [Online]. Available: http://www.ti.com/lit/pdf/slva704.

[5] «i2cdetect(8) - Linux man page» [Online]. Available: https://linux.die.net/man/8/i2cdetect.

[6] «Raspbian,» [Online]. Available: https://www.raspberrypi.org/documentation/raspbian/.

[7] «minicom(1) - Linux man page» [Online]. Available: https://linux.die.net/man/1/minicom.

[8] «picocom(8) - Linux man page» [Online]. Available: https://linux.die.net/man/8/picocom.

[9] «PijFORTHos forth reference» [Online]. Available: https://github.com/organix/pijFORTHos.

[10] «La codifica RGB,» [Online]. Available: https://it.ccm.net/contents/697-la-codifica-rgb-rvb.

[11] R. C. Gonzalez e R. E. Woods, «Using Lookup Table» in *Digital Image Processing*, Pearson, 2017.

[12] «Bayer filter» [Online]. Available: https://en.wikipedia.org/wiki/Bayer_filter.

[13] «TCS34725» [Online]. Available: https://github.com/hideakitai/TCS34725.

[14] «TCS34725 Color Sensor» [Online]. Available: https://www.waveshare.com/w/upload/5/53/Infrared-Temperature-Sensor-Code.7z.