

UNIVERSITÀ DEGLI STUDI DI PERUGIA
Facoltà di Scienze Matematiche, Fisiche e Naturali

CORSO DI LAUREA IN INFORMATICA



Tesi di Laurea

**Implementazione di un sistema mobile ed autonomo
per la ricerca di oggetti in base al colore**

Laureando

Simone Mariotti

Relatori

Prof. Marco Baiocchi

Dott. Emanuele Palazzetti

Anno Accademico 2013-2014

TODO: DEDICA

Indice

Introduzione	1
1 Visione Artificiale	2
2 Componenti hardware	7
2.1 UDOO Quad	7
2.2 Tank Kit	10
2.3 Sensori	10
2.3.1 Sensore di riflessività - QRD1114	10
2.3.2 Sensore di distanza a ultrasuoni - HC-SR04	13
3 Componenti software	17
3.1 OpenCV	17
3.2 ADK	18
3.3 ADK Toolkit	20
4 Implementazione	22
Conclusioni	23
Bibliografia	24

Introduzione

Obiettivi

Strumenti utilizzati

Capitolo 1

Visione Artificiale

La *visione artificiale* è l'unione dei procedimenti che permettono di creare un modello del mondo reale in tre dimensioni a partire da numerose immagini bidimensionali per cercare di emulare il processo biologico della vista. Negli umani, così come in molte altre specie, vedere non è solo scattare una fotografia bidimensionale mentale di un ambiente, è interpretare le informazioni ricevute attraverso la retina per analizzare e creare un modello 3D dell'ambiente circostante.

Un sistema di visione artificiale, per provare ad avvicinarsi alla capacità visiva e di interpretazione umana, ha bisogno di numerosi componenti di diversa natura: ottici, elettronici e meccanici per acquisire e memorizzare le immagini da elaborare. Il primo tentativo in tal senso è datato 1883 quando Paul Gottlieb Nipkow inventò il primo sistema in grado di trasformare o meglio trasdurre, informazioni visive in un segnale elettrico. Il sistema si basa su un disco con dei fori equidistanti e di egual raggio lungo una spirale che gira ad una velocità costante, l'immagine inquadrata viene indirizzata verso i fori da una lente, un sensore sul retro del disco trasforma le variazioni di

luminosità in segnali elettrici. Il ricevitore emetterà luce in base ad i segnali elettrici e attraverso un disco identico al primo e in rotazione sincrona proietterà un'immagine con tante righe quante sono i fori del disco. Questo sistema porterà alla creazione della TV meccanica nel 1925.

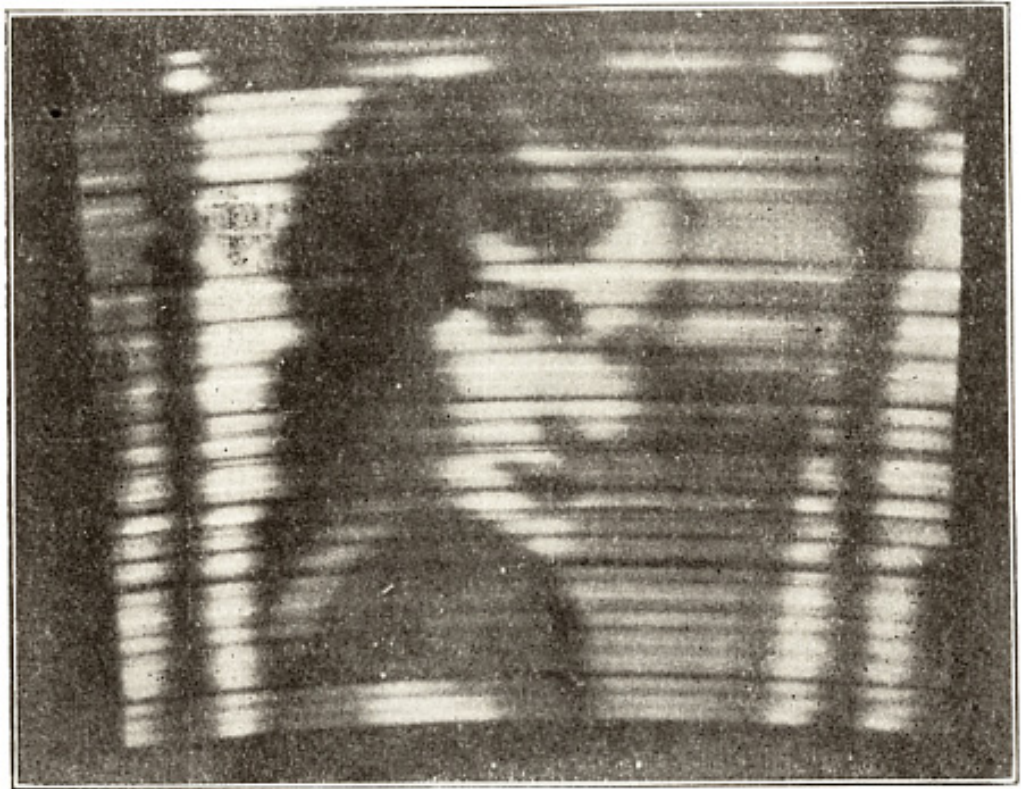


Figura 1.1: Immagine visibile su una TV meccanica del 1925

Da quel periodo si sono fatti enormi progressi grazie alla miniaturizzazione dell'elettronica. Gli scienziati per produrre un effetto sempre migliori si sono ispirati alla macchina migliore che esiste, il corpo umano: analizzando da vicino l'occhio e in particolare la retina si sono resi conto che era formata da minuscoli recettori sensibili alla luce, ognuno collegato con un filamento al cervello; l'insieme di questi filamenti è il nervo ottico. Si è così deciso di

creare dei micro sensori di luce e formarne un'enorme matrice. Il risultato di questi studi sono i sensori CCD¹ e CMOS², entrambi hanno come elemento base il fotodiodo che equivale ad un pixel. Il CCD è un sensore analogico³, ha bisogno di più energia ma offre in genere una qualità superiore un rumore minore ad un costo più elevato. Il CMOS è un sensore digitale che offre una

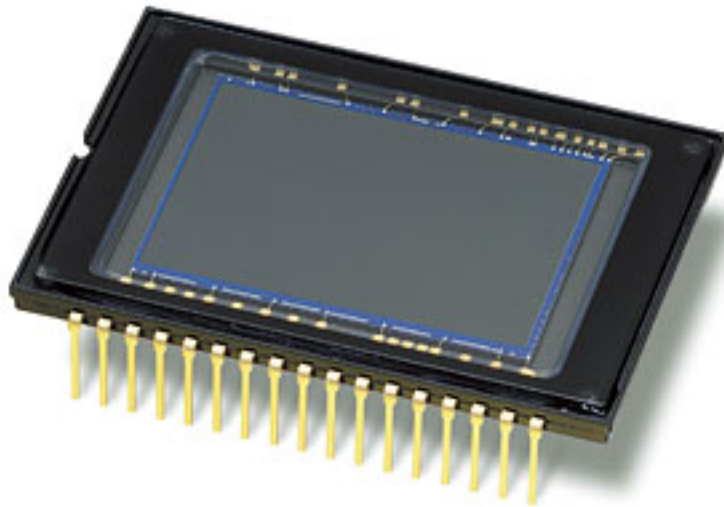


Figura 1.2: Un sensore CCD moderno

buona qualità di immagine e costo minore, per contro l'immagine presenta un forte rumore a causa della conversione analogico-digitale.

La visione artificiale ha tre utilizzi principali: ricognizione, identificazione, individuazione.

- **Ricognizione:** ricercare uno o più oggetti, scelti a priori, e organizzarli in insiemi generici o classi e mantenere informazioni riguardo il

¹Charge-Coupled Device

²Complementary Metal-Oxide-Semiconductor

³Il sensore più grande esistente è di 1,4 Gigapixel ed è montato sul telescopio Pan-STARRS sviluppato per l'individuazione di meteoriti in rotta di collisione con la Terra

loro posizionamento nella scena.

Esempio: ricercare in un'immagine o in un video tutte le persone, le macchine o gli animali.

- **Identificazione:** identificare una istanza specifica di una classe di oggetti.

Esempio: ricercare in un'immagine o in un video un volto, una macchina o un animale specifico..

- **Individuazione:** cercare una condizione specifica nell'immagine.

Esempio: cercare impurezze nelle immagini a raggi X di superfici o materiali.

Un altro ambito in cui si sta utilizzando la visione artificiale è la modellazione di ambienti 3D a partire da due o più immagini 2D; questo è stato reso possibile grazie all'enorme aumento della capacità di elaborazione grafica delle GPU.

Ci sono molti altri campi in cui la visione artificiale ha rivoluzionato il settore. Uno di questi è la medicina dove senza questa tecnica non esisterebbero radiografie, angiografie, tomografie e molte altre; in questo ambito la visione artificiale salva vite identificando anomalie e problemi come i tumori che l'occhio umano non potrebbe vedere se non in seguito a procedure molto invasive. Un'altro è il controllo di veicoli autonomi, sta crescendo ad un ritmo vertiginoso la quantità dei veicoli autonomi esistenti, sia per scopi militari sia per scopi civili: autovetture, droni, robot e carri per il rifornimento tutto quello che un tempo era naturalmente dipendente della guida umana ora può muoversi autonomamente. E' proprio in questo campo civile, o più



Figura 1.3: Drone della società statunitense Amazon. Consegnerà prodotti a domicilio in completa autonomia

precisamente in quello scientifico, che ricade il nostro lavoro di tesi su un robot autonomo per l'identificazione di colori.

Capitolo 2

Componenti hardware

2.1 UDOO Quad

UDOO è un progetto tutto italiano di una piattaforma hardware destinata alla generazione dei “makers”, cioè quelle persone che vogliono realizzare i propri progetti con le tecnologie a basso costo ad oggi disponibili. La scheda ha visto la luce dopo una sorprendente campagna di crowdfunding ¹ terminata l'8 Giugno 2013 con 4172 donazioni per un totale di \$641.612 a fronte di \$27.000 richiesti per iniziare la produzione. Per permettere l'utilizzo di librerie e applicazioni computazionalmente pesanti come openCV, PureData e altre UDOO monta un processore ARM Freescale i.MX6 Cortex-A9 Quad core 1GHz che supporta sia Android che Linux. Il tutto è completato da una GPU Vivante, 1GB di RAM DDR3, numerose porte di I/O come SATA, microfono, audio out, Ethernet, HDMI, USB, connettore per display LVDS con touch screen, connettore CSI per camera esterna e connettività bluetooth e Wi-Fi. La periferica di “boot” è una microSD il che permette

¹dall'inglese crowd, folla e funding, finanziamento. In italiano finanziamento collettivo.

un rapido passaggio da Linux a Android e viceversa. Quello che però rende veramente unica questa piattaforma, e che ne ha fatto la nostra scelta per questo progetto di tesi, è la presenza di un Arduino DUE completamente integrato nella stessa board. E' presente una CPU Atmel SAM3X8E ARM Cortex-M3² e 76 GPIO³, di cui 62 digitali e 14 digitali/analogici, disposti per essere perfettamente compatibili con la piedinatura dell'Arduino DUE e dell'Arduino UNO Rev.3.

La presenza di un Arduino DUE all'interno della board rende UDOO una scheda di prototipazione a tutti gli effetti e apre nuovi interessanti scenari e possibilità unendo la versatilità e semplicità di Arduino, la potenza di calcolo del Freescale i.MX6 e le numerose periferiche disponibili per Linux o Android.

Essendo una piattaforma open-source è possibile accedere alla shell del sistema operativo come root tramite la porta seriale integrata e modificare a piacimento la configurazione del sistema operativo. Arduino è collegato al Freescale i.MX6 tramite un bus interno e quindi viene rilevato come una normale periferica USB da Linux; su Android la comunicazione tra i due dispositivi avviene sullo stesso bus ma usa lo standard USB OTG⁴. L'interconnessione tra l'accessorio Arduino e l'applicazione Android è realizzata tramite l'ADK⁵ 2012, di cui parleremo più avanti in questo stesso capitolo, che permette l'integrazione delle più disparate periferiche a dispositivi

²la stessa di cui dispone l'Arduino DUE

³General Purpose Input/Output

⁴On-The -Go è una specifica che permettere di agire come host ad un qualsiasi dispositivo (tipicamente smartphone e tablet). A differenza dell'USB classico l'OTG è driver-less, cioè non necessita l'installazione di driver specifici per ogni dispositivo

⁵Android Development Kit

2.2 Tank Kit

Per dare la giusta stabilità e manovrabilità al robot si è deciso di usare una locomozione a cingoli che richiede solo due motori e permette di ruotare sul posto o comunque in spazi ristretti: la nostra scelta è stata il “Multi-Chassis - Tank Version”. Questa piattaforma, appositamente pensata per la realizzazione di robot multifunzione, si è rivelata la scelta perfetta in quanto possiede due potenti motori DC già forniti di riduttori 48:1 per affrontare terreni impervi e scoscesi, quattro ruote da 52 mm di diametro a cui sono applicati i due cingoli. E’ presente anche un alloggiamento per un servomotore standard che nella nostra applicazione non è stato usato. L’intelaiatura, di alluminio spesso 2,5 mm, presenta numerosi fori e asole per il montaggio di accessorie quali sensori, staffe e motori. Presenta inoltre un “doppio fondo” in cui sono alloggiati i motori DC e i riduttori e in cui è possibile sistemare altri componenti che non debbano essere facilmente accessibili.

2.3 Sensori

2.3.1 Sensore di riflessività - QRD1114

Avevamo la necessità di fornire al robot un modo per rilevare eventuali sconfinamenti dall’ambiente di test che fosse il più flessibile possibile. Abbiamo optato per il sensore di riflessività QRD1114 prodotto dalla Fairchild Semiconductor: questo sensore è costituito da un LED infrarosso e un fototransistor tarato sulla luce infrarossa e con filtro per la luce solare onde evitare disturbi. Il robot era stato pensato per lavorare su un tavolo o altra

superficie con spigoli netti: per rilevare l'imminente caduta in questo tipo di ambiente sarebbe stato sufficiente un sensore di distanza puntato verso terra. Con il sensore di riflessività abbiamo reso possibile l'utilizzo in terra o comunque in ambienti estesi delimitati da un recinto spesso circa 10cm realizzato con materiale a basse riflettività come del semplice cartoncino nero opaco. Il sensore non fa differenza tra il cartoncino nero o lo spazio a fianco di un tavolo, rileva semplicemente una riflessività vicina allo zero. Il sensore così come fornito dal produttore non è direttamente utilizzabile, per far sì che Arduino potesse acquisire dal sensore valori proporzionali alla riflessività del materiale in esame abbiamo dovuto realizzare un circuito elettronico di interfaccia.

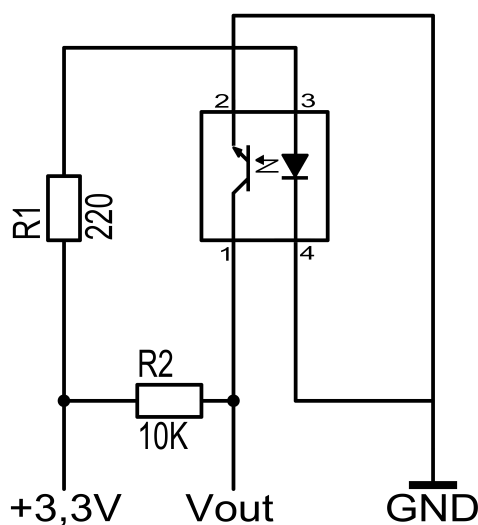


Figura 2.2: Circuito di interfaccia tra Arduino DUE e il sensore QRD1114

Il circuito alimenta il LED tramite una resistenza da 220Ω ($R1$) e collega

V_{CC} ⁶ al collettore del fototransistor tramite una resistenza da $10\text{ k}\Omega$ ($R2$) mentre l'emettitore è collegato a terra; il punto da cui prelevare il segnale (V_{OUT}) è tra $R2$ e il collettore. Il principio alla base del circuito è semplice: il LED è sempre acceso e illumina in modo diffuso parallelamente al fototransistor. Il fototransistor in assenza di luce o, nel nostro caso, in presenza di una bassa riflessività si trova in stato di interdizione; i pin di Arduino impostati come input sono in configurazione "alta impedenza", equivalenti ad un interruttore aperto dal punto di vista circuitale, quindi non c'è passaggio di corrente né tramite il transistor né tramite la resistenza $R2$ il che porta esattamente il valore di V_{CC} in ingresso ad Arduino. Quando il fototransistor è totalmente illuminato, cioè in presenza di alta riflessività, entra in stato di conduzione così che nel punto V_{OUT} si venga a trovare la massa. Ogni stato intermedio di illuminazione equivale ad una conduzione parziale del fotoresistore a cui corrisponde una tensione proporzionale alla riflessività sul pin V_{OUT} . Il sensore può essere utilizzato in modalità digitale o analogica semplicemente collegando il pin V_{OUT} ad un pin digitale o analogico e cambiando la configurazione relativa all'interno della programmazione di Arduino. La configurazione digitale si è rivelata inadatta all'applicazione in quanto l'intervallo di valori che rappresenta un oggetto riflette nelle immediate vicinanze del sensore, e quindi lo stato di normale funzionamento, è troppo stretto e anche un minimo sussulto manda il robot in allarme sconfinamento. La configurazione analogica invece ci permette di avere circa 1024 valori discreti dal trasduttore, abbiamo quindi impostato una soglia oltre la quale il robot va in allarme sconfinamento; è importante notare che questa libertà nello scegliere una soglia di allarme ci permette

⁶pari a 3,3 V nell'Arduino DUE

di effettuare una calibrazione affinata in base al materiale su cui si svolge il test per minimizzare la possibilità di falsi positivi.

2.3.2 Sensore di distanza a ultrasuoni - HC-SR04

Il robot aveva bisogno di conoscere la distanza degli oggetti che si trovavano di fronte ad esso, la scelta è ricaduta un sensore ad ultrasuoni, a discapito di uno ad infrarossi, per la totale assenza di disturbi dovuti all'illuminazione dell'ambiente di test. In particolare abbiamo scelto il sensore HC-SR04 che offre ottime prestazioni, è facile da integrare ed è di dimensioni contenute. Il sensore può rilevare oggetti distanti da 2 cm a 400 cm con una risoluzione di 0,3 cm entro un cono frontale con apertura di 60° , 30° per lato rispetto alla perpendicolare dal sensore.

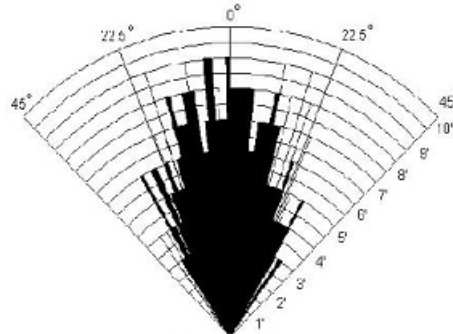


Figura 2.3: Risposta del sensore rispetto alla posizione dell'ostacolo

Il collegamento alla piattaforma di sviluppo è effettuato tramite quattro pin: $+5V$, *trig*, *echo* e *GND*. Il funzionamento è piuttosto semplice, una volta alimentato il sensore si invia sul pin *trig* un impulso di durata $10\mu s$, il sensore invia quindi un impulso ultrasonico e restituisce sul pin *echo* un segnale positivo di durata pari al tempo di percorrenza andata e ritorno del

segnale ultrasonico. Una volta ottenuto il tempo di percorrenza si utilizza la formula del moto rettilineo uniforme:

$$v = \frac{\Delta s}{\Delta t}$$

sapendo che la velocità del suono a 20 °C è approssimabile a 344 m/s allora la durata dell'impulso di *echo* sarà compreso tra 116.3 μ s e 23,26 ms rispettivamente per distanze percorse di 4 cm (2+2) e di 8 m (4+4).

Per ricavare la distanza (d) in centimetri a partire dalla durata dell'impulso notando che 344 m/s = 29,1 cm/ μ s e indicando con T_e la durata dell'impulso di echo usiamo:

$$d = \frac{T_e}{2 \cdot 29,1}$$

Per quanto riguarda il collegamento ad Arduino se la nostra scheda accettasse input a 5 V questo sensore non avrebbe bisogno di nessun accorgimento per essere collegato, purtroppo però, anche se la nostra UDOO e l'Arduino DUE in essa integrato possono fornire alimentazione a 5 V, non possono accettare input allo stesso voltaggio ma solo a 3,3 V.

Allo stesso modo del sensore QRD1114 ci serve un circuito di interfaccia che in questo caso sarà un semplice partitore di tensione il cui schema generico è riportato di seguito

Il partitore di tensione, a fronte di un noto valore di tensione di ingresso V_{IN} riesce a fornirci il valore di tensione desiderato in uscita V_{OUT} tramite due resistenze opportunamente dimensionate.

La formula che regola il partitore di tensione è la seguente

$$V_{OUT} = V_{IN} \cdot \frac{R_2}{R_1 + R_2}$$

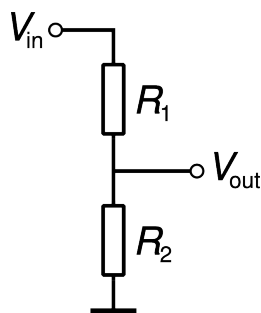


Figura 2.4: Schema generico del partitore di tensione

nel nostro caso $V_{IN} = 5\text{ V}$ e $V_{OUT} = 3,3\text{ V}$ quindi scegliendo arbitrariamente il valore di R_1 $1\text{ k}\Omega$ ricaviamo semplicemente il valore di R_2 tramite la formula inversa

$$R_2 = R_1 \cdot \frac{V_{OUT}}{V_{IN} - V_{OUT}}$$

che da come risultato $1941,48\ \Omega$; approssimando il valore appena calcolato al più vicino taglio standard di resistenze, cioè $R_2 = 2\text{ k}\Omega$, si ottiene $V_{OUT} = 3,\bar{3}$ che è assolutamente accettabile.

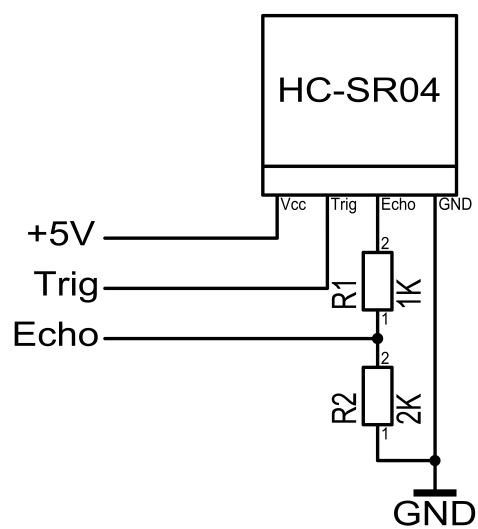


Figura 2.5: Circuito di interfaccia tra Arduino DUE e il sensore HC-SR04

Capitolo 3

Componenti software

3.1 OpenCV

Per dare al robot una visione dell'ambiente circostante ci serviva una libreria di visione artificiale. Le più famose disponibili per Java sono OpenCV e JavaCV; la nostra scelta è stata OpenCV per via della maggiore disponibilità di documentazione. OpenCV¹ è stata originariamente sviluppata da Intel che cercava di migliorare le prestazioni delle loro CPU con applicazioni computazionalmente pesanti come per esempio gli algoritmi di *ray tracing*² ed è ora disponibile sotto licenza BSD.

La libreria per poter funzionare su un dispositivo Android ha bisogno di un'applicazione di supporto chiamata OpenCV Manager che può essere scaricata dal Play Store o installata direttamente con il pacchetto apk.³ La

¹Open Source Computer Vision

²Tecnica di geometria ottica che analizza il percorso dei raggi di luce. In grafica 3D è un algoritmo di rendering che costruisce la scena facendo partire i raggi luminosi dalla camera (visuale del giocatore) invece che dalla sorgente luminosa.[1]

³Android Package, il tipo di file delle App Android simile al formato JAR

libreria viene caricata in modo asincrono all'avvio dell'App e fornisce ogni frame come una matrice dove ogni cella rappresenta un pixel; OpenCV usa il tipo di dato interno Mat per rappresentare le matrici. Tutte le operazioni effettuate su oggetti di tipo Mat usano algoritmi appositamente ottimizzati per le operazioni tra matrici.

3.2 ADK

In Android, a partire dalla versione 3.1, è supportato il protocollo Android Open Accessory (AOA), ogni accessorio sviluppato per Android deve supportare lo stesso protocollo.

A causa della limitata autonomia dei dispositivi su cui Android è maggiormente utilizzato (smartphone e tablet) l'AOA impone che l'accessorio funga da host e alimenti quindi il bus utilizzato

Per usare Arduino come un accessorio per Android e rispettare lo standard AOA esiste la libreria *adk.h* che permette di dialogare con un dispositivo Android dopo un'accurata configurazione. Android distingue diversi accessori sulla base di informazioni che ogni accessorio fornisce all'atto della connessione, queste informazioni sono:

- Vendor
- Name
- Longname
- Version
- Url

■ Serial

Il dispositivo Android accetterà la richiesta di connessione solo se le stringhe identificative in suo possesso combaciano perfettamente con quelle fornite dall'accessorio. Per ottenere effettivamente la connessione bisogna indicare ad Android quali sono gli accessori supportati e quale applicazioni è in grado di gestire un particolare accessorio. Questo si ottiene inserendo la successiva direttiva nel Manifest file dell'App

Codice 3.1: Porzione del Manifest file dell'App

```
...  
<meta-data  
    android:name="android.hardware.usb.action.  
                                USB_ACCESSORY_ATTACHED"  
    android:resource="@xml/usb_accessory_filter" />  
...
```

In questo modo alla connessione di qualsiasi accessorio sarà il sistema operativo a suggerire di aprire una certa app se l'accessorio connesso è presente nel file `usb_accessory_filter.xml` associato.

Nel nostro caso il file `usb_accessory_filter.xml` si presenta così

Codice 3.2: `usb_accessory_filter.xml`

```
<resources>  
    <usb-accessory  
        version="0.1.0"  
        model="Mobile-Tanker"  
        manufacturer="Simone Mariotti"/>  
</resources>
```

Le stesse identiche stringhe saranno impostate durante la fase di configurazione di Arduino in modo da permettere l'accoppiamento.

3.3 ADK Toolkit

L'ADK toolkit è una libreria che aggiunge un grado di astrazione all'ADK, l'invio e la ricezione dei messaggi sono semplificati così come l'inizializzazione della connessione.

Il toolkit si basa su due classi principali: `AdkManager` e `AdkMessage`.

`AdkManager` espone metodi per la gestione della connessione e per l'invio e la ricezioni di dati. `AdkMessage` rappresenta il messaggio ricevuto dall'accessorio ne suo formato nativo, cioè un array di byte; tramite dei metodi ausiliari, `getString()`, `getFloat()`, `getInt()`, è possibile ottenere una versione tipizzata del messaggio e naturalmente la versione “grezza” dei dati con `getBytes()` e `getByte()`.

Grazie a questa libreria l'uso dell'ADK, originariamente poco intuitivo, diventa efficiente ed elegante

Codice 3.3: Inizializzazione della connessione con l'accessorio

```
private AdkManager mAdkManager;

@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    mAdkManager = new AdkManager(this);
}
```



```
@Override  
protected void onResume() {  
    super.onResume();  
    mAdkManager.open();  
}
```

Codice 3.4: Lettura e scrittura dati

```
// Write  
adkManager.write("Ciao da Android!");  
  
// Read  
AdkMessage response = adkManager.read();  
System.out.println(response.getString());  
// Could outputs: "Ciao da Arduino!"
```

Capitolo 4

Implementazione

Conclusioni e sviluppi futuri

Bibliografia

- [1] Marco Ambroglio. Ray Tracing GPU-based - http://www.unibg.it/dati/corsi/38001/45691-lezione_06052011_optix.pdf

Elenco delle immagini

1.1	Immagine visibile su una TV meccanica del 1925	3
1.2	Un sensore CCD moderno	4
1.3	Drone della società statunitense Amazon. Consegnerà prodotti a domicilio in completa autonomia	6
2.1	Schema piedinatura UDOO	9
2.2	Circuito di interfaccia tra Arduino DUE e il sensore QRD1114	11
2.3	Risposta del sensore rispetto alla posizione dell'ostacolo . . .	13
2.4	Schema generico del partitore di tensione	15
2.5	Circuito di interfaccia tra Arduino DUE e il sensore HC-SR04	16

Appendice